Université de Montréal

Optimisation des horaires des agents et du routage des appels dans les centres d'appels

par
Wyean Chan

Département d'informatique et de recherche opérationnelle
Faculté des arts et des sciences

Thèse présentée à la Faculté des études supérieures et postdoctorales
en vue de l'obtention du grade de Philosophiæ Doctor (Ph.D.)
en informatique

Septembre, 2013

Résumé

Nous étudions la gestion de centres d'appels multi-compétences, ayant plusieurs types d'appels et groupes d'agents. Un centre d'appels est un système de files d'attente très complexe, où il faut généralement utiliser un simulateur pour évaluer ses performances.

Tout d'abord, nous développons un simulateur de centres d'appels basé sur la simulation d'une chaîne de Markov en temps continu (CMTC), qui est plus rapide que la simulation conventionnelle par événements discrets. À l'aide d'une méthode d'uniformisation de la CMTC, le simulateur simule la chaîne de Markov en temps discret imbriquée de la CMTC. Nous proposons des stratégies pour utiliser efficacement ce simulateur dans l'optimisation de l'affectation des agents. En particulier, nous étudions l'utilisation des variables aléatoires communes.

Deuxièmement, nous optimisons les horaires des agents sur plusieurs périodes en proposant un algorithme basé sur des coupes de sous-gradients et la simulation. Ce problème est généralement trop grand pour être optimisé par la programmation en nombres entiers. Alors, nous relaxons l'intégralité des variables et nous proposons des méthodes pour arrondir les solutions. Nous présentons une recherche locale pour améliorer la solution finale.

Ensuite, nous étudions l'optimisation du routage des appels aux agents. Nous proposons une nouvelle politique de routage basé sur des poids, les temps d'attente des appels, et les temps d'inoccupation des agents ou le nombre d'agents libres. Nous développons un algorithme génétique modifié pour optimiser les paramètres de routage. Au lieu d'effectuer des mutations ou des croisements, cet algorithme optimise les paramètres des lois de probabilité qui génèrent la population de solutions.

Par la suite, nous développons un algorithme d'affectation des agents basé sur l'agrégation, la théorie des files d'attente et la probabilité de délai. Cet algorithme heuristique est rapide, car il n'emploie pas la simulation. La contrainte sur le niveau de service est convertie en une contrainte sur la probabilité de délai. Par après, nous proposons une variante d'un modèle de CMTC basé sur le temps d'attente du client à la tête de la file. Et finalement,

nous présentons une extension d'un algorithme de coupe pour l'optimisation stochastique avec recours de l'affectation des agents dans un centre d'appels multi-compétences.

Mots clés: centre de contacts, optimisation stochastique, affectation, planification, quarts de travail, heuristique, chaîne de Markov, simulation.

ABSTRACT

We study the management of multi-skill call centers, with multiple call types and agent groups. A call center is a very complex queueing system, and we generally need to use simulation in order to evaluate its performances.

First, we develop a call center simulator based on the simulation of a continuous-time Markov chain (CTMC) that is faster than traditional discrete-event simulation. Using an uniformization method, this simulator simulates the embedded discrete-time Markov chain of the CTMC. We propose strategies to use this simulator efficiently within a staffing optimization algorithm. In particular, we study the use of common random numbers.

Secondly, we propose an algorithm, based on subgradient cuts and simulation, to optimize the shift scheduling problem. Since this problem is usually too big to be solved as an integer programming problem, we relax the integer variables and we propose methods to round the solutions. We also present a local search to improve the final solution.

Next, we study the call routing optimization problem. We propose a new routing policy based on weights, call waiting times, and agent idle times or the number of idle agents. We develop a modified genetic algorithm to optimize all the routing parameters. Instead of doing mutations and crossovers, this algorithm refines the parametric distributions used to generate the population of solutions.

We also develop a staffing algorithm based on aggregation, queueing theory and delay probability. This heuristic algorithm is fast, because it does not use simulation. The service level constraint is converted into a delay probability constraint. Moreover, we propose a variant of a CTMC model based on the waiting time of the customer at the head of the queue. Finally, we design an extension of a cutting-plane algorithm to optimize the stochastic version with recourse of the staffing problem for multi-skill call centers.

Keywords: contact center, stochastic optimization, staffing, scheduling, work shift, heuristic, Markov chain, simulation.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES ALGORITHMES

LISTE DES SIGLES

AGM Algorithme génétique modifié

AWT *Acceptable waiting time*, ou temps d'attente acceptable, utilisé pour mesurer le niveau de service

BP Politique de routage basé sur des poids

CMTC Chaîne de Markov en temps continu

CMTD Chaîne de Markov en temps discret

CP *Cutting-plane*, algorithme d'optimisation par coupes linéaires

DG Méthode de descente du gradient

E Politique de routage à priorités égales

ED Type de simulation par événements discrets

FCFS *First come, first served*, ou premier arrivé, premier servi

IP *Integer programming*, ou programmation à nombres entiers

LISF *Longest-idle-server-first*, ou le premier agent inactif avant

LP *Linear programming*, ou programmation linéaire

LG$c\mu$ Adaptation linéaire de la règle de routage $c\mu$ généralisée

P Politique de routage par priorités

PD Politique de routage par priorités et temps de délai

PDS Politique de routage par priorités, délais et seuils d'agents libres

PDY Programmation dynamique

PS Politique de routage par priorités et seuils d'agents libres

QN Méthode d'optimisation quasi-Newton

SCP *Scheduling cutting-plane*, version CP pour la planification des horaires

SIPP *Stationary independent period by period*, une méthode de planification stationnaire indépendante par période

SL *Service level*, ou le niveau de service

TS *Two-stage* ou *two-step*, ou algorithme à deux étapes

VAC Variables aléatoires communes

VAI Variables aléatoires indépendantes

À tous ceux qui vivent
pour atteindre leurs rêves,

et

À tous ceux qui luttent
pour un monde meilleur,
plus juste et équitable.

REMERCIEMENTS

Le cheminement menant à l'aboutissement de cette thèse de doctorat fut une longue aventure très enrichissante. Tout d'abord, je tiens à remercier Pierre L'Ecuyer, mon directeur de thèse, de m'avoir offert la chance de poursuivre mes études au doctorat, et de m'avoir soutenu autant sur le plan financier que sur le développement pédagogique. J'ai eu aussi la chance de voyager dans plusieurs régions du monde lors des conférences.

Je remercie également le Conseil de recherches en sciences naturelles et en génie du Canada (CRSNG), le Fonds de recherche du Québec – Nature et technologies (FQRNT) et Bell Canada pour leur support financier grâce à la bourse BMP innovation. Je remercie Naoufel Thabet et Mohamed Nassim de Bell Canada.

Je remercie mes collaborateurs Athanassios Avramidis, Eric Buist, Michel Gendreau, Ger Koole et Ornella Pisacane. Puis, je tiens à saluer Richard Simard et les nombreux gens que j'ai côtoyés au laboratoire de simulation et d'optimisation du DIRO durant toutes ces longues années.

Finalement, je tiens à remercier ma famille : mes parents et mes sœurs.

CHAPITRE 1

INTRODUCTION

Un *centre d'appels* peut être défini comme étant une organisation dont l'activité économique principale repose sur l'interaction avec les clients par téléphone [103]. Les éléments formant un centre d'appels se résument à l'ensemble du personnel, les ordinateurs, les équipements de communication et de routage, les meubles de bureau et le ou les immeubles de travail. Ces centres représentent une industrie importante et en forte croissance depuis les dernières années. Ils permettent, par exemple, de fournir de l'information ou du support aux clients, de payer des factures ou d'accroître les ventes d'une compagnie. Certains centres d'appels sont mêmes indispensables ; il serait difficile d'imaginer un organisme gouvernemental, une institution financière ou les services d'urgence 911 sans service téléphonique.

Le Bureau of Labor Statistics [36] classe les *agents au service à la clientèle* au 7e rang dans la liste des occupations les plus nombreuses aux États-Unies en 2010. Cet organisme estime qu'il y avait environ 2.3 millions d'agents aux États-Unis en 2008 dont 23% travaillaient dans les secteurs financiers et d'assurances, et 15% dans le secteur de l'administration et de service de support [35]. La majorité de ces agents travaillent dans des centres d'appels, mais les données incluent aussi les agents qui interagissent directement devant les clients. L'étude prévoit une augmentation du nombre d'emplois de 18% pour atteindre 2.7 millions en 2018. Un autre rapport estime qu'il y avait 2.15 millions d'agents

en mai 2010 avec des salaires horaires moyen de 15.76 $US et médian de 14.64$US, et un salaire annuel moyen de 32 780 $US [34]. Le coût salarial annuel des agents est alors estimé à 70.3 milliards $US en 2010 aux États-Unis. Comparé aux données de mai 2007 (avant la crise financière de 2008-09), il y avait 2.2 millions d'agents et un salaire annuel moyen de 31 040 $US pour un coût salarial total de 68.1 milliards $US [33]. Étant donné que les centres d'appels dépensent généralement 60% à 70% de leur budget sur le coût salarial [58], il est important d'optimiser la gestion de la main-d'œuvre.

Dans le domaine des sciences de la gestion, plusieurs études sur l'industrie de service (banques, restaurants, ...) observent un lien entre la satisfaction des clients, leur loyauté envers l'entreprise et les profits générés [73, 76, 110, 123]. La loyauté d'un client est généralement plus élevée lorsque celui-ci reçoit un bon service. Ceci permet d'augmenter les profits, car un client loyal a une meilleure chance d'effectuer des achats subséquents ou de recommander l'entreprise à de nouveaux clients potentiels (ce qui est du même coup de la publicité gratuite). Puis, la satisfaction des employés permet d'augmenter la productivité et l'efficacité au travail. Les centres d'appels ont souvent un haut taux de rotation du personnel (plusieurs ont même un taux de rotation annuel supérieur à 50% [58]). Ceci signifie des coûts et des temps de formations substantiels, sans parler de l'expérience perdue par les démissionnaires. Pour bien gérer un centre d'appels, il faut assurer une bonne qualité de service (nous donnerons des exemples de mesures de qualité à la section 1.2) aux clients ainsi qu'aux employés, et ce, tout en minimisant les coûts d'opérations.

La gestion des centres d'appels s'est grandement complexifiée avec les développements technologiques en télécommunication. Par exemple, les équipements de plus en plus sophistiqués permettent une vaste possibilité de routage des appels. Il n'est pas rare qu'une grande entreprise relocalise son centre de service dans un autre pays avec une main-d'œuvre à bon marché, en Inde par exemple. Il est aussi possible d'opérer de grands centres d'appels virtuels en reliant plusieurs centres situés dans différentes régions (Toronto et Montréal par exemple) ou même jusqu'aux résidences des agents désirant travailler à la maison. Certains centres, appelés *centres de contacts*, permettent aux clients d'interagir à travers d'autres médiums de communication tels que le clavardage et les courriels ou courriers électroniques. Accompagnant la croissance de cette industrie, il y a l'émergence des

entreprises spécialisées dans les prestations de services (ou "*outsourcers*"). Une compagnie pourrait préférer externaliser une partie ou la totalité de son volume d'appels à ces entreprises de sous-traitance. La croissance de l'industrie des centres d'appels modernes apporte une grande variété de problèmes encore peu étudiés.

1.1 Fonctionnement d'un centre d'appels

Gans et al. [58] donnent une bonne description du fonctionnement d'un centre d'appels et des différentes étapes auxquelles un appel doit passer avant d'être répondu par un agent.

L'interaction entre une entreprise et un client s'effectue par le médium d'un *appel*. Un appel peut être originaire d'un client vers l'entreprise (*appel entrant*) ou d'un agent vers un client (*appel sortant*). Les appels sortants sont souvent associés au travail de télémarketing. Un centre d'appels qui traite à la fois les appels entrants et sortants est appelé un centre *mixte* ou "*blend*". Durant les périodes où le volume d'appels entrants est faible, les agents pourraient être sollicités à produire des ventes en appelant des clients potentiels. Les appels sont catégorisés par *type*. Un type d'appel peut être défini selon le service demandé, le produit en vente, la région géographique de l'appel, la langue de communication, etc. Nous utilisons le terme *centre d'appels multi-compétences* (ou "*multi-skills*") pour désigner un centre qui traite plusieurs types d'appels, où chaque type requiert une compétence ou habileté spécifique. Dans cette thèse, nous nous concentrons sur les centres d'appels multi-compétences avec uniquement des appels entrants.

L'employé qui interagit avec le client au téléphone s'appelle un *agent* ou *représentant au service à la clientèle*. Un agent possède un ensemble d'habiletés pour répondre à certains types d'appels. Certaines habiletés peuvent être acquises en suivant des périodes de formation offertes par l'entreprise. Le salaire d'un agent dépend généralement du nombre d'habiletés qu'il possède, de son quart de travail, de son expérience, etc. Pour faciliter la gestion, les agents sont souvent divisés en groupes. Un groupe pouvant servir beaucoup de types d'appels est appelé *généraliste* et un groupe pouvant servir peu, *spécialiste*. En réalité, un client pourrait demander plusieurs services et être répondu par différents agents, mais les problèmes de l'industrie supposent habituellement qu'un client ne demande qu'un

seul type de service par appel. De plus, nous supposons qu'un agent ne peut servir qu'un seul client à la fois, mais si le service préemptif est permis, l'agent peut interrompre momentanément le service d'un client afin de répondre à un autre appel (jugé généralement plus important). Le service préemptif est beaucoup plus approprié dans un centre de contacts où un agent doit répondre à la fois aux appels et aux courriels. Il peut interrompre la rédaction d'un courriel à l'arrivée d'un nouvel appel.

Un centre d'appels loue des lignes téléphoniques d'une compagnie de télécommunication à un coût fixe. Chaque appel dans le système occupe une ligne. Lorsque toutes les lignes sont occupées, tout nouvel appel sera *bloqué* et recevra un signal occupé. Déterminer le nombre optimal de lignes à louer est aussi un problème d'optimisation, mais nous supposons habituellement dans nos problèmes que les centres d'appels disposent d'une quantité suffisante de lignes téléphoniques.

Un appel arrive au centre par un commutateur appelé *Private Automatic Branch eXchange* (PABX ou PBX) qui est connecté aux lignes téléphoniques. Un centre d'appels possède généralement un système de traitement vocal automatisé, nommé *Interactive Voice Response* (IVR). L'IVR est un équipement important qui permet d'identifier le type de l'appel et l'information du client, ainsi que d'offrir des services automatisés pour les tâches simples. Par exemple, les données statistiques pour l'année 1999 d'une banque en Israël montre qu'environ 65% des appels étaient résolus dans l'IVR, sans l'assistance d'un agent [26]. Gans et al. [58] rapportent que la proportion des appels qui se terminent dans l'IVR est autour de 80% pour plusieurs centres d'appels bancaires.

Lorsqu'un client doit parler à un agent, l'IVR l'envoie à un *routeur* appelé *Automatic Call Distributor* (ACD) qui gère les files d'attente et distribue les appels aux agents. Les politiques de routage sont implémentées dans l'ACD. Lorsqu'un nouvel appel arrive, l'ACD choisit un agent libre pour le répondre, sinon il place l'appel dans une file d'attente. Quand un agent termine un appel, l'ACD cherche le prochain appel en attente à se faire servir, en fonction des habiletés de cet agent. Dans les centres d'appels où les groupes partagent plusieurs habiletés communes, il existe un grand nombre de configurations de routage possibles. Par exemple, il y a le routage par priorités où chaque groupe d'agents possède une liste de priorités des types d'appels à servir et une liste similaire pour chaque

type d'appel. Il peut avoir une différence substantielle sur la qualité de service entre un bon routage et un mauvais routage. L'ACD s'occupe également de recueillir les données statistiques telles que les temps d'arrivée des appels, les temps d'attente, les durées de service, le nombre d'abandons, etc. Cependant, les données sont souvent agrégées par période (de 15 ou 30 minutes par exemple), ce qui complique l'analyse statistique.

Lorsqu'un client en attente devient impatient, il *abandonne* la file et le centre d'appels. Les clients abandonnés ou bloqués peuvent augmenter artificiellement le volume d'appels en *recomposant* au centre. De plus, les clients bloqués ou ayant abandonné représentent une perte de revenu potentiel. Il y a aussi le problème d'efficacité du service qui est de répondre adéquatement aux demandes du client. Un client peut *rappeler* lorsqu'il ne reçoit pas un service satisfaisant, ce qui contribue aussi à l'augmentation du volume d'appels. Dans la réalité, certains agents ou groupes répondent avec plus de succès que d'autres pour un type d'appel. Le problème à considérer dans ce contexte est : "Un client devrait être servi spécialement par un agent *plus performant* (taux de succès élevé) après combien de rappels ?"

La figure 1.1, inspirée de Gans et al. [58], résume les acheminements possibles des appels. Dans la modélisation des problèmes de cette thèse, nous ignorons l'IVR et nous considérons uniquement les processus d'arrivée des appels qui entrent dans l'ACD, c'est-à-dire les clients qui demandent une interaction avec un agent.

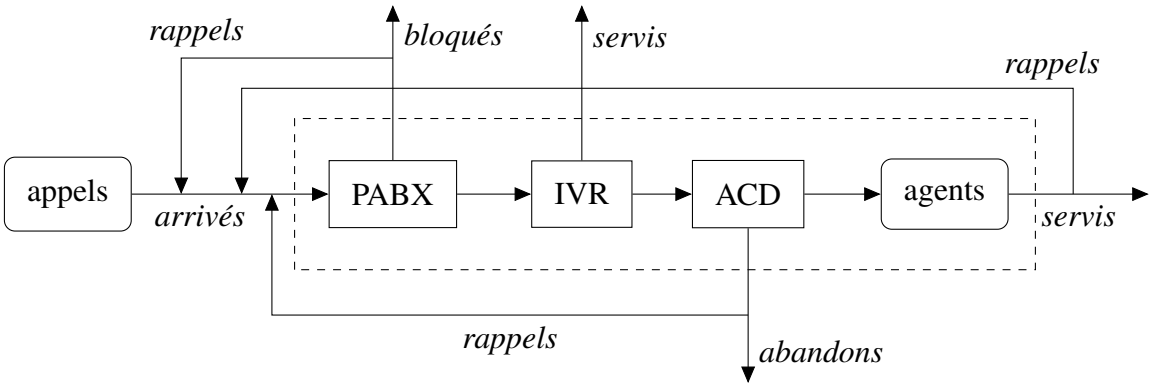


Figure 1.1 – Acheminement des appels.

1.2 Mesures de performance

Les mesures de performance permettent d'évaluer la qualité de service et l'efficacité d'un centre d'appels. Étant donné le lien observé entre la satisfaction des clients et les revenus engendrés [73, 76], il est important que le gestionnaire d'un centre d'appels maintient un minimum de qualité de service tout en limitant les coûts d'opérations.

Il existe plusieurs mesures de performance, dont le *niveau de service* ou "*service level*" (SL) qui est une des mesures les plus utilisées dans l'industrie. Cette mesure est aussi appelée le "*telephone service factor*" (TSF). La formule du SL n'est pas unique, mais elle se résume par : *la proportion des appels répondus dans un délai d'attente d'au plus* τ *secondes*, où τ est un paramètre. Nous donnerons une définition plus précise et quelques variantes à la section 2.3, page 20.

En général, la contrainte sur le SL est d'atteindre une proportion $x\%$ pour un seuil de délai τ donné. Ce seuil est appelé le *temps d'attente acceptable* ou "*acceptable waiting time*" (AWT). Voici un exemple typique : la règle du 80/20 signifie que 80% des appels ne doivent pas avoir attendu plus de 20 secondes. Jusqu'à récemment, le Conseil de la radiodiffusion et des télécommunications canadiennes (CRTC) [44] pénalisait fortement les compagnies de télécommunication lorsque les SL agrégés par mois de certains types d'appels étaient inférieurs à une proportion ciblée $x\%$.

Du point de vue de l'optimisation, la définition du SL nous encourage à répondre en priorité aux appels ayant attendu égal ou moins de τ, parce que servir ceux ayant attendu plus de τ ne peut pas améliorer la mesure. Mais ceci n'est pas équitable pour les clients qui sont arrivés avant. Un gestionnaire de centre d'appels surveille généralement plusieurs mesures de qualité de service même s'il n'impose pas nécessairement de contrainte précise. Nous présenterons d'autres mesures de performance telles que le temps d'attente moyen et la fraction d'abandons dans la section 2.3.

Tel que mentionné auparavant, la satisfaction et le bien-être des agents influencent la performance d'un centre d'appels. Un gestionnaire mesure souvent la productivité du centre d'appels selon le *taux d'occupation* des agents, calculé par : *le temps total passé à répondre aux appels divisé par le temps total au poste de travail*. Une définition plus

précise se trouve à la section 2.3, page 20. Pour un centre d'appels avec un fort volume d'appels et beaucoup d'agents, il est souvent possible d'avoir une très bonne qualité de service avec un taux d'occupation supérieur à 90%. Par contre, un taux d'occupation trop élevé nuira à l'efficacité, car les agents seront épuisés. Lorsque le volume est très faible (durant la nuit par exemple), il faudra plus d'agents, par rapport au volume d'appels, pour obtenir le même SL. Il est généralement difficile d'avoir un taux d'occupation élevé tout en ayant une bonne qualité de service dans un petit centre d'appels, à cause de la variance stochastique plus importante. D'autres facteurs [79, 120] qui influencent la satisfaction des agents sont : pouvoir servir en fonction de leurs préférences de types d'appels, avoir la capacité et les outils pour résoudre les problèmes des clients, le barème sur la durée de service accordée par appel (qui est parfois trop court), l'écoute de contrôle des conversations entre les agents et les clients, etc.

1.3 Gestion d'un centre d'appels

La gestion d'un centre d'appels est un exercice complexe qui comprend plusieurs prises de décisions à différentes étapes dans le temps. Plus l'écart de temps entre la prise de décision et son application est court, plus ce type de décision est fréquent.

1. Création du centre : Cette tâche appartient à la direction de l'entreprise. La direction doit déterminer l'objectif du centre d'appels, l'emplacement du centre, les équipements à acheter ou à louer, les types de service à offrir, déterminer les groupes d'agents et leurs habiletés, choisir la politique de routage, etc. Cette étape demande souvent beaucoup de temps et d'argent, et les décisions changent peu après l'ouverture du centre d'appels.
2. À long terme : Le gestionnaire d'un centre d'appels doit calculer des prévisions grossières des volumes d'appels quelques mois à l'avance afin de procéder à l'embauche et à la formation des agents.
3. À moyen terme : Le gestionnaire doit créer les horaires hebdomadaires ou journaliers en fonction des agents disponibles et des prévisions révisées plus précises.

4. À court terme : Des recours sont parfois possibles durant la journée ou la veille, pour corriger les erreurs de prévisions. Les cas d'absentéisme ou de congé de maladie sont fréquents également. Si un gestionnaire croit que le volume sera plus fort, il peut faire rentrer des agents sur appel ou annuler des séances de formation afin de répondre au surplus d'appels. Dans le cas où le volume serait plus faible que prévu, un gestionnaire pourrait organiser des périodes de formation ou offrir des congés sans solde aux agents. Ceci correspond à un problème stochastique avec recours.

5. En temps réel : Des prises de décisions se font en temps réel par le routeur. Lors de l'arrivée d'un appel, le routeur doit décider d'envoyer cet appel à un agent libre capable de le servir ou de placer cet appel dans une file d'attente. Quand un agent devient libre, le routeur doit choisir entre lui affecter un appel en attente ou de garder l'agent inoccupé. Le routage est un problème beaucoup plus complexe pour les centres d'appels multi-compétences, car il peut y avoir un grand nombre de choix de routages possibles.

La prévision du volume d'appels est un problème important et difficile en soi. Plusieurs facteurs peuvent varier le volume d'appels. Le nombre d'appels peut dépendre du jour de le semaine, du mois, de la saison, des journées de fêtes civiles, d'une nouvelle campagne de publicité, du lancement d'un nouveau produit, d'une panne de service ou même de la pluie. Généralement, les centres d'appels observent une corrélation positive entre les périodes d'une même journée. Ceci justifie les options de recours qui permettent au gestionnaire du centre de réagir dès qu'il s'aperçoit que le volume d'appels du matin est supérieur ou inférieur au prévision.

La gestion de la main-d'œuvre est un autre problème difficile qui dépend de la prévision du volume d'appels. Lors de la création des horaires, le gestionnaire doit respecter les contraintes sur les qualités de service, sur les quarts de travail, le nombre d'agents disponibles, etc. Puisque les qualités de service sont des variables aléatoires, les problèmes sont optimisés en fonction de leurs espérances mathématiques. Ces espérances sont estimées à l'aide d'algorithmes d'approximation ou par la simulation.

Dans certains centres, un gestionnaire doit assurer un niveau d'équité entre les clients et aussi entre les agents. L'équité entre les clients vise à ce que tous les types d'appels reçoivent un SL similaire. Par exemple, supposons que la contrainte du SL est 80/20 (80% des appels répondus en 20 secondes ou moins) et que le SL est 70% pour un matin en particulier. Il arrive que les critères d'équité du centre ne permettent pas d'augmenter le SL de 80% à 90% en après-midi pour compenser la sous-performance du matin (en supposant des volumes égaux), parce que les clients qui appellent en après-midi seraient fortement favorisés. D'autre part, par souci d'équité, le gestionnaire préférera avoir des taux d'occupation similaires entre les groupes d'agents. Dans les centres de télémarketing où un boni est souvent attribué au nombre de ventes, chaque agent voudrait avoir la même chance de recevoir le boni, d'où le besoin d'équilibrer les taux d'occupation. À notre connaissance, il existe très peu d'études dans la littérature portant sur l'optimisation des centres d'appels avec des contraintes d'équité.

L'absentéisme est une source d'incertitude souvent ignorée dans les modèles de centres d'appels. Le Bureau of Labor Statistics [32] montre que les absences non planifiées chez les travailleurs à temps plein dans l'industrie du service représentaient 3.6% des effectifs en 2010. Certains centres d'appels ont des taux d'absentéisme encore plus élevés, allant parfois jusqu'à 20% ! L'approche généralement utilisée dans la pratique est de résoudre de prime abord le problème d'affectation sans considérer l'absentéisme, puis de multiplier la solution par un facteur pour tenir compte des absences [58].

La direction d'un centre d'appels pourrait faire affaire avec une entreprise de sous-traitance. Elle devra alors décider du volume d'appels à externaliser à la tierce entreprise. L'entreprise de sous-traitance limite souvent ses risques contre les fluctuations du volume d'appels en limitant le nombre d'appels qu'elle peut recevoir dans son contrat. Un centre d'appels de sous-traitance peut également faire affaire avec plusieurs compagnies à la fois. Leurs agents sont formés pour servir différents types de clients.

L'optimisation des politiques de routage est souvent un problème négligé par rapport à celui de la prévision du volume d'appels ou de la planification des horaires des agents. Un bon routage permet d'améliorer ou équilibrer les performances d'un centre d'appels lors d'imprévus modérés tels que l'absentéisme des agents ou une mauvaise prévision du vo-

lume d'appels. De plus, la majorité des dépenses d'un centre d'appels est souvent attribuée aux salaires des agents. Choisir une bonne politique de routage permettrait d'économiser sur les coûts de recours (ou changements d'horaires). Bien entendu, une mauvaise prévision ou planification des agents aura un impact plus important. Une grave pénurie d'agents causera de mauvaises qualités de service même si le routage est optimal.

Dans la pratique, la politique de routage est rarement changée une fois qu'elle est implémentée. En regardant la documentation des routeurs (Cisco [42, 43], Nortel [107] et Avaya [15, 16]), les routeurs actuels sont souvent limités aux politiques déjà implémentées, et créer ou modifier une politique peut s'avérer très complexe, voire impossible dans certains cas. Cependant, il n'est pas rare qu'un gestionnaire modifie les paramètres de la politique de routage à la main et de manière ad hoc. Il est alors important d'étudier et de chercher des politiques de routage plus efficaces qui sont faciles à implémenter.

1.4 Plan de la thèse

Dans cette thèse, nous nous intéressons principalement aux centres d'appels multi-compétences avec strictement des appels entrants, parce que ce type de problème est le plus fréquent dans l'industrie. Tout d'abord, nous présentons le modèle général du centre d'appels multi-compétences qui nous intéresse au chapitre 2. Nous définissons certaines notations mathématiques qui seront utilisées au cours de cette thèse. Nous présentons également quelques mesures de performance et politiques de routage populaires que nous retrouvons dans la pratique. Nous présentons une revue de la littérature au chapitre 3, en particulier sur les recherches portant sur l'optimisation des agents et du routage dans les centres d'appels. Le reste de cette thèse est divisé comme suit.

Si tous les processus stochastiques d'un centre d'appels sont markoviens, alors nous pouvons l'approximer par une chaîne de Markov en temps continu (CMTC). Au chapitre 4, nous présentons notre article [30] sur l'amélioration de l'optimisation des agents à l'aide de la simulation de la chaîne de Markov en temps discret (CMTD) imbriquée de la CMTC. Ce simulateur CMTD est plus rapide qu'un simulateur traditionnel par événements discrets (ED), parce qu'il ne fait qu'incrémenter et décrémenter des compteurs de l'état de la

CMTC. Nous utilisons la méthode d'uniformisation d'une CMTC afin de pouvoir générer les instances de transition de la CMTC indépendamment de la CMTD imbriquée. L'uniformisation simplifie beaucoup la simulation, mais elle ajoute des transitions fictives qui peuvent ralentir la simulation.

Nous étudions l'application de ce simulateur CMTD pour améliorer les performances de l'algorithme d'affectation des agents de Ceẓik et L'Ecuyer [37]. Puisque l'algorithme utilise la simulation pour estimer les coupes basées sur les sous-gradients, le processus d'optimisation est bruité. Pour réduire ce bruit, nous synchronisons les simulations à l'aide des variables aléatoires communes (VAC). Cependant, il n'est pas évident de bien appliquer les VAC avec l'algorithme d'optimisation, parce que la synchronisation dépend du nombre d'agents. Mais nous ne connaissons pas à l'avance les solutions que l'algorithme visitera ! Nous étudions et comparons plusieurs manières de synchroniser avec les VAC. Une des meilleures méthodes est d'utiliser les mêmes VAC par estimation de sous-gradient. Cette méthode donne un bon compromis entre la réduction du bruit et la minimisation de transitions fictives. Dans les exemples numériques, nous considérons des centres d'appels markoviens et non markoviens. Pour les exemples non markoviens, l'optimisation débute avec le simulateur CMTD, puis se termine avec le simulateur ED. Les résultats montrent une amélioration de la performance d'optimisation dans tous les cas.

Au chapitre 5, nous présentons notre article [17] sur la planification des quarts de travail des agents sur plusieurs périodes. Notre algorithme est une adaptation et une généralisation des algorithmes de coupes linéaires et de simulation d'Atlason et al. [12] et de Ceẓik et L'Ecuyer [37]. Nous incluons le mécanisme de transfert temporaire d'agents de Bhulai et al. [23] qui permet de transférer temporairement des agents vers un groupe requérant moins d'habiletés.

Ce problème de planification constitue un grand problème d'optimisation à nombres entiers tel qu'il faut généralement optimiser une version relaxée du problème. Nous étudions deux méthodes pour arrondir les variables relaxées. Nous avons une méthode simple qui arrondit en fonction d'un seuil fractionnaire, et une autre méthode, plus complexe, qui fixe itérativement chaque variable relaxée à une valeur entière. La deuxième méthode s'assure que la solution arrondie soit réalisable pour le problème relaxé. Cependant, nous

découvrons que la méthode simple par seuil est préférable. Il n'est pas nécessaire de bien arrondir les solutions intermédiaires, car d'autres coupes s'ajouteront au cours de l'optimisation. La deuxième méthode d'arrondissement risque même d'augmenter les erreurs d'optimisation et d'éliminer la solution optimale ! Une autre difficulté est l'arrondissement des variables de transfert. Les transferts en chaîne impliquent qu'une mauvaise valeur d'une variable de transfert peut se répercuter sur plusieurs variables. Nous développons un algorithme pour assurer la cohérence entre les variables de transfert et le vecteur d'agents. Les résultats numériques montrent que notre algorithme est significativement meilleur que l'approche conventionnelle à deux étapes.

Au chapitre 6, nous étudions exclusivement le problème d'optimisation du routage entre les appels et les agents. Ce chapitre se base sur notre article [38] qui est en deuxième tour de révision. Traditionnellement, les routeurs étaient implantés avec des politiques de routage simples, fixes ou peu flexibles. Les routeurs modernes sont de plus en plus sophistiqués, et il est maintenant possible d'implanter des politiques de routage plus complexes, à l'aide de scriptes par exemple.

La contribution principale de ce chapitre est une nouvelle politique de routage basé sur des poids, les temps d'attente des clients et les temps d'inoccupation des agents ou le nombre d'agents libres. Nous définissons un poids pour chaque paire de type d'appel et de groupe d'agents pouvant répondre à ce type. Un poids, défini par une fonction linéaire, représente le niveau de priorité de la paire. Les paramètres (la constante et les coefficients) continus permettent de contrôler dynamiquement ces priorités en fonction de l'état du centre d'appels. En calculant les poids à partir de données des appels et des agents simultanément, notre politique sert à la fois pour le routage d'un nouvel appel vers un agent inoccupé, et pour le routage d'un agent qui se libère vers un appel en attente. Cette double utilité de la même fonction de poids constitue une différence majeure par rapport à la majorité des autres politiques retrouvées en pratique ou dans la littérature. Le routage peut aussi imposer des temps de délai à l'aide de poids négatifs, où un agent demeure libre même s'il y a des appels en attente auxquels il peut répondre. Cette politique de routage avec des poids peut approximer d'autres politiques comme le routage avec des listes de priorités.

De plus, nous proposons une adaptation linéaire de la règle $c\mu$ généralisée pour des fonctions de type boîte noire, qui ne sont pas nécessairement convexes. Nous étudions aussi plusieurs routages basés sur la pratique, comme le routage avec listes de priorités, temps de délai et seuils d'agents libres.

Une différence importante avec la littérature est que nous désirons optimiser le routage pour des modèles de centres d'appels plus réalistes. Ainsi, nous considérons des fonctions objectifs de type boîte noire et tout centre d'appels pouvant être approximé par la simulation. Cette approche très générale donne des problèmes de routage beaucoup plus difficiles qui ne sont pas résolus dans la littérature. Notre recherche représente vraisemblablement une première étude sur l'optimisation du routage de centres d'appels réalistes.

Nous proposons un algorithme génétique modifié (AGM) pour l'optimisation de tous les paramètres des politiques de routage que nous considérons dans ce chapitre. L'AGM est basé sur les *algorithmes d'estimation de distribution* [94, 106] et de la méthode d'optimisation *entropie croisée* [25, 49, 116]. Au lieu d'effectuer des mutations ou des croisements, l'AGM optimise les paramètres des lois de probabilité qui génèrent la population de solutions. Il est très flexible, et il peut optimiser les variables entières, continues et les problèmes combinatoires.

Dans la section numérique, nous comparons les performances optimisées de notre politique avec celles des politiques tirées de la pratique et la littérature. Les résultats numériques montrent que notre politique réussit en général aussi bien ou mieux que la meilleure des autres politiques.

Tous les algorithmes d'optimisation étudiés jusqu'à ce point de la thèse utilisent la simulation qui peut être très coûteuse en temps d'exécution. Au chapitre 7, nous présentons un algorithme heuristique rapide pour optimiser l'affectation des agents *sans l'aide d'un simulateur* d'un centre d'appels. Notre méthode se base sur les observations de Wallace et Whitt [132] qui supposent des durées de service indépendantes des types d'appels et des groupes d'agents. Ils observent qu'un centre d'appels, où chaque agent a 2 ou 3 habiletés bien choisies, peut performer presque aussi bien qu'un centre d'appels où tous les agents ont toutes les habiletés. Nous disons qu'un tel centre d'appels est *efficace*. Comme la probabilité de délai est un peu plus facile à approximer que le SL, nous traduisons les contraintes

sur les SL par des contraintes sur les probabilités de délai. Nous utilisons la théorie des files d'attente pour générer les scénarios du problème stochastique. Les exemples numériques montrent de bonnes performances d'optimisation pour les centres d'appels "efficaces" et une bonne satisfaction des mesures de performance agrégées.

Les deux derniers chapitres présentent principalement des idées pour des futurs projets de recherche. Au chapitre 8, nous présentons une variante du modèle de Koole et al. [90] pour l'approximation d'une CMTC basée sur le temps d'attente du client à la tête de la file d'attente. Contrairement aux modèles de CMTC traditionnels basés sur les longueurs des files d'attente, notre modèle permet de mieux approximer le routage basé sur les temps d'attente des appels et d'estimer la fonction de répartition du temps d'attente quand il y a plusieurs types d'appels. Ce modèle permet aussi d'optimiser le routage en fonction des temps de délai, à l'aide de la programmation dynamique [89]. Nous proposons des modifications simples au modèle de Koole et al. [90], telles que les transitions d'incrémentation de la CMTC incluent toujours les taux d'arrivée. Les résultats numériques montrent que notre variante est généralement plus précise que le modèle original pour un système de file $M/M/n$.

Finalement, au chapitre 9, nous étudions le problème d'affectation avec des taux d'arrivée stochastiques et des moyens de recours pour les centres d'appels multi-compétences. Nous supposons qu'un gestionnaire doit planifier les agents à l'avance (disons quelques semaines) en se basant sur des prévisions probabilistes. À l'approche de la journée planifiée, les prévisions sont révisées et elles deviennent plus précises. Le gestionnaire peut alors corriger son effectif d'agents selon ces révisions, en retirant ou en ajoutant des agents avec des coûts de pénalités. Nous proposons des idées pour adapter l'algorithme de coupes et de simulation de Ceşik et L'Ecuyer [37] pour optimiser ce type de problème. Nous étudions aussi une version du problème avec une contrainte par chance, où les seuils de SL doivent être atteints avec une certaine probabilité.

CHAPITRE 2

MODÉLISATION D'UN CENTRE D'APPELS

Nous décrivons le modèle général d'un centre d'appels multi-compétences que nous étudions dans cette thèse, accompagné de quelques notations mathématiques qui seront utilisées par la suite. Nous définissons également quelques mesures de performance et politiques de routage couramment utilisées dans l'industrie.

2.1 Description du modèle

Nous considérons le modèle de centres d'appels avec uniquement des appels entrants. Supposons un centre d'appels avec K types d'appels et I groupes d'agents. Chaque appel est associé à un type de service $k \in \{1,\dots,K\}$ et chaque agent appartient à un groupe $i \in \{1,\dots,I\}$. Chaque type d'appel requiert une habileté distincte et l'ensemble des habiletés du groupe i est défini par $\mathcal{S}_i \subseteq \{1,\dots,K\}$. Similairement, nous définissons l'ensemble des groupes pouvant servir le type d'appel k par $\mathcal{T}_k \subseteq \{1,\dots,I\}$. La figure 2.1 donne des exemples de modèles canoniques de centres d'appels. Nous supposons que le service est *non préemptif*, c'est-à-dire qu'un agent ne peut pas interrompre la conversation avec un client afin de répondre à un autre appel. Nous supposons aussi qu'un client ne demande qu'un seul type de service et qu'il n'effectue aucun rappel même s'il a abandonné ou a été

bloqué. Nous présumons que la quantité de lignes téléphoniques et les capacités des files d'attente sont suffisamment grandes telles qu'aucun appel ne sera bloqué.

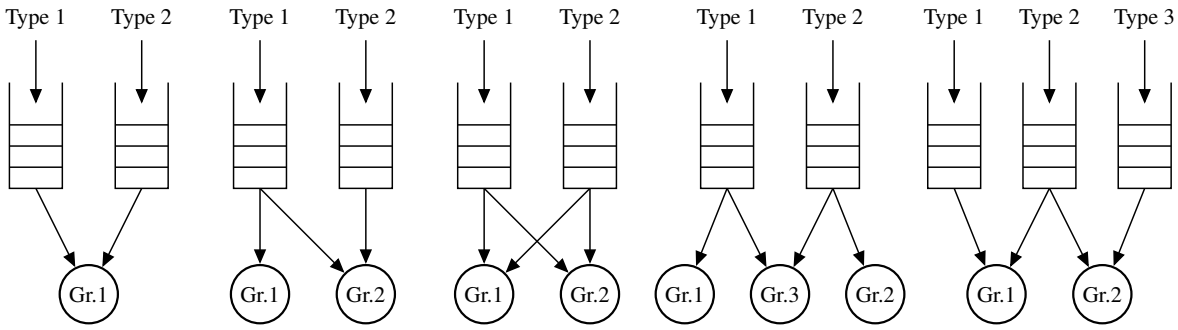


Figure 2.1 – Quelques modèles canoniques de centres d'appels : les modèles V, N, X, M et W. Chaque modèle a 2 ou 3 types d'appels et 1 à 3 groupes d'agents. Les arcs définissent les habiletés des groupes d'agents.

Une journée est divisée en P périodes d'une durée constante (par exemple 15 minutes). Le processus d'arrivée du type d'appel k dans une journée est divisé en P processus d'arrivée, soit un pour chaque période. Les arrivées sont généralement modélisées par un processus de Poisson avec un taux $\lambda_{k,p}$ constant sur chaque période p. Brown et al. [26] ont étudié les données d'un centre d'appels d'une banque, et leurs tests n'ont pas rejeté l'hypothèse d'un processus de Poisson constant par période. La durée de service d'un appel de type k par un agent du groupe i est souvent considérée une variable aléatoire de loi exponentielle de moyenne $1/\mu_{k,i}$. Des études récentes suggèrent plutôt une variable log-normale (voir la section 3.1, page 27). Dans le cas où la durée de service dépend uniquement du type de l'appel, le taux de service est donné par μ_k. La durée de patience (temps avant d'abandonner) d'un client de type k est aussi modélisée selon une loi exponentielle de moyenne $1/\nu_k$. Lorsqu'un client de type k entre dans la file d'attente, il y a une probabilité η_k qu'il abandonne immédiatement, sinon il abandonnera la file lorsque son temps de patience sera écoulé.

Dans la planification des horaires, le problème est de déterminer le nombre d'agents à affecter par groupe et par quart de travail. Les quarts de travail sont souvent prédéfinis, car ils doivent respecter des règles ou des conventions de travail telles que le nombre

de pauses et les périodes de repas. En supposant S quarts de travail possibles, la matrice binaire $\bar{\mathbf{A}}$ de taille $P \times S$ définit la couverture des périodes pour chaque quart de travail avec l'élément $\bar{a}(p,s) = 1$ si le quart de travail s couvre la période p, sinon $\bar{a}(p,s) = 0$. Le vecteur de décision est $\mathbf{x} = (x_{1,1}, \ldots, x_{1,S}, \ldots, x_{I,1}, \ldots, x_{I,S})^{\mathrm{T}}$, et l'élément $x_{i,s}$ représente le nombre d'agents du groupe i affectés au quart de travail s. La matrice $\mathbf{A}$ de taille $IP \times IS$ est définie par I fois la matrice $\bar{\mathbf{A}}$ sur la diagonale et 0 ailleurs. Le vecteur $\mathbf{y} = \mathbf{A}\mathbf{x} = (y_{1,1}, \ldots, y_{1,P}, \ldots, y_{I,1}, \ldots, y_{I,P})^{\mathrm{T}}$ contient le nombre d'agents par groupe et par période.

Pour certains problèmes, le modèle est simplifié pour ne considérer qu'une seule période $P = 1$ et nous cachons l'indice p des notations. Pour le problème d'affectation des agents pour une seule période, le vecteur de décision est $\mathbf{y}$.

2.2 Politiques de routage

Une composante importante du modèle d'un centre d'appels est le routage. Lorsqu'un nouvel appel arrive, le routeur cherche un agent libre capable de servir cet appel. S'il n'en existe aucun, l'appel est placé dans une file d'attente. Quand un agent termine un appel et devient libre, le routeur cherche le prochain appel en attente capable d'être servi par cet agent. Les règles de routage sont déterminées par la politique de routage du centre d'appels. Nous disons qu'une politique de routage est *préservatrice de travail* ou "*work-conserving*" si un agent doit être occupé tant qu'il y a des appels en attente dont il peut servir. Toutes les politiques que nous considérons dans cette thèse respectent les règles d'équité de base suivantes :

1. Les appels du même type sont toujours servis selon la règle du *premier arrivé, premier servi* ou "*first-come, first-served*" (FCFS).

2. Les agents appartenant au même groupe répartissent le travail en choisissant *le premier agent inactif avant*. Cette règle est aussi appelée "*longest-idle-server-first*" (LISF) [7] et "*longest-available-agent*" (LAA) [42]. L'agent libre qui a terminé son dernier appel le plus tôt est le prochain agent de ce groupe à travailler.

Le rôle des politiques de routage est de déterminer les priorités entre les types d'appels et les groupes d'agents. Certaines politiques sont simples et faciles à implémenter, alors que d'autres sont complexes et difficiles, par exemple une politique de contrôle obtenue par la programmation dynamique. Le choix de la politique de routage est important, car le routage influence grandement les mesures de performance.

Nous décrivons quelques politiques de routage populaires que nous retrouvons dans l'industrie. Dans le chapitre 6, nous introduisons une nouvelle politique de routage basé sur des poids, les temps d'attente des appels et les temps d'inactivité des agents.

2.2.1 Routage à priorités égales (E)

Cette politique de routage est la plus rudimentaire. Il n'y a aucun paramètre de routage. Tous les types d'appels et tous les groupes d'agents ont la même priorité. Les appels sont servis par ordre du *premier arrivé, premier servi* peu importe le type, et les agents travaillent selon *le premier agent inactif avant* peu importe le groupe. La seule condition est que l'agent doit posséder l'habileté pour servir l'appel.

2.2.2 Routage par priorités (P)

Cette politique est aussi appelée "*overflow routing*" [107] et "*priority routing*" [132]. Chaque type d'appel et chaque groupe d'agents possède une liste de priorités qui détermine l'ordre de visite des groupes d'agents ou des files d'attente. La liste de priorités du groupe i, aussi appelée *liste groupe-vers-type*, est $\mathcal{L}_i = (\mathcal{L}_i^{(1)}, \ldots, \mathcal{L}_i^{(m_i)})$ où m_i est le nombre de *niveaux* ou *rangs de priorités* et $\mathcal{L}_i^{(1)}, \ldots, \mathcal{L}_i^{(m_i)}$ représentent une partition de $\mathcal{S}_i$. C'est-à-dire $\mathcal{L}_i^{(a)} \cap \mathcal{L}_i^{(b)} = \emptyset$ pour $a \neq b$ et $\cup_{a=1}^{m_i} \mathcal{L}_i^{(a)} = \mathcal{S}_i$. Quand un agent devient libre, il vérifie premièrement s'il y a un appel en attente parmi les files d'attente correspondant aux types dans $\mathcal{L}_i^{(1)}$. S'il y en existe au moins un, il sert l'appel ayant attendu le plus longtemps (FCFS). Autrement, l'agent poursuit en vérifiant les files d'attente correspondant aux types d'appels dans $\mathcal{L}_i^{(2)}$, ainsi de suite. L'agent demeure libre s'il ne trouve aucun appel en attente.

Le processus est similaire pour le choix d'un agent lors de l'arrivée d'un appel. Nous

définissons la liste de priorités du type k, aussi appelée *liste type-vers-groupe*, par $\mathcal{G}_k = (\mathcal{G}_k^{(1)}, \ldots, \mathcal{G}_k^{(n_k)})$ où n_k est le nombre de niveaux de priorités, et $\mathcal{G}_k^{(1)}, \ldots, \mathcal{G}_k^{(n_k)}$ forment une partition de $\mathcal{T}_k$. Lorsqu'un appel du type k arrive, le routeur va chercher un agent libre parmi les groupes dans $\mathcal{G}_k^{(1)}$ en suivant la règle LISF. S'il n'y a aucun agent libre, il poursuit avec les groupes dans $\mathcal{G}_k^{(2)}$, ainsi de suite. S'il ne trouve aucun agent libre après avoir parcouru $\mathcal{G}_k$, l'appel est placé dans la file d'attente k.

Cette politique est préservatrice de travail. Elle est souvent utilisée dans la pratique quand les agents ont des habiletés primaires et secondaires. Les types d'appels correspondant aux habiletés primaires sont alors placés sur haute priorité. Nous obtenons la politique de routage E lorsque $\mathcal{L}_i^{(1)} = \mathcal{S}_i$ et $\mathcal{G}_k^{(1)} = \mathcal{T}_k$ pour tout i et tout k.

2.2.3 Routage par priorités et temps de délai (PD)

Une extension à la politique de routage par priorités est l'ajout de temps de délai. Nous définissons l'ensemble des habiletés du centre d'appels par $\mathcal{H} = \{(k,i) : k \in \mathcal{S}_i, \forall i\}$. Il y a un paramètre de délai $d_{k,i} \geq 0$ pour chaque paire $(k,i) \in \mathcal{H}$. La nouvelle condition de délai s'ajoute à la politique de routage P : un appel du type k doit avoir attendu au moins un temps $d_{k,i}$ avant d'être servi par un agent du groupe i. Ceci implique qu'un appel de type k ne peut jamais être servi immédiatement à l'arrivée par un agent du groupe i si $d_{k,i} > 0$. Si tous les $d_{k,i} = 0$, nous retrouvons la politique de routage par priorités P. Cette politique n'est pas préservatrice de travail à cause des délais.

Dans l'industrie, le paramètre $d_{k,i}$ est généralement inférieur au AWT que nous dénotons par τ_k, car si $d_{k,i} > \tau_k$ alors le groupe i ne peut jamais donner un bon SL au type d'appel k. Cette politique peut être utilisée pour favoriser les habiletés primaires des agents.

Les temps de délai sont fixes dans cette politique. Bien entendu, le routage serait meilleur si les paramètres de délais étaient dynamiques, en fonction de l'état du centre d'appels, mais la politique serait plus difficile à optimiser. Nous présentons une nouvelle politique de routage basé sur des poids qui permet d'avoir des temps de délai dynamiques dans le chapitre 6.

2.3 Mesures de performance

Nous décrivons plus en détails les mesures de performance introduites à la section 1.2. Dans la réalité, les mesures de performance sont calculées à partir des données observées, à la fin d'une période ou d'une journée. Dans un problème d'optimisation, l'approche conventionnelle est d'optimiser en fonction des espérances des fonctions de mesures de performance. Notons qu'en pratique, les équipements de centres d'appels calculent les données statistiques à partir d'observations réelles et non des espérances.

Il n'y a pas de formule précise ou standard pour les mesures de performance. Il existe souvent différentes implémentations ou manières de compter les observations. Par exemple, faut-il compter les appels ayant abandonné quasi instantanément ? Lorsqu'une journée est divisée en plusieurs périodes, à quelle période doit-on comptabiliser un appel dont le séjour dans le système chevauche sur plusieurs périodes ? Le simulateur [31] utilisé dans nos expériences comptabilise les appels selon leur période d'arrivée. L'avantage est que le nombre de périodes demeure P, mais le désavantage est que les mesures de performance ne sont pas nécessairement disponibles immédiatement à la fin d'une période. Une autre possibilité est de comptabiliser les appels à leur sortie du centre. L'avantage est que les mesures de performance sont disponibles dès la fin de chaque période. Par contre, il peut avoir plus de P périodes dans une journée si des appels sont répondus après la fermeture.

Nous présentons dans cette section les mesures de performance utilisées dans cette thèse ou jugées importantes. Les formules des mesures de performance implémentées dans le simulateur sont disponibles dans le guide de ce dernier [29].

Une mesure de performance populaire est le *niveau de service* ou "*service level*" (SL) qui est définie par la proportion des appels répondus dans un délai d'attente d'au plus τ, appelé le *temps d'attente acceptable* ou "*acceptable waiting time*" (AWT). Soient $X_{\mathrm{B}}(\tau)$ le nombre d'appels répondus dans un délai inférieur ou égal à τ, N le nombre total d'appels et $A_{\mathrm{B}}(\tau)$ le nombre d'abandons ayant attendu au plus τ. La formule du SL considérée dans cette thèse est :

$$f_{\mathrm{S}}(\tau) = \frac{\mathbb{E}[X_{\mathrm{B}}(\tau)]}{\mathbb{E}[N - A_{\mathrm{B}}(\tau)]}, \tag{2.1}$$

où $\mathbb{E}$ est l'opérateur de l'espérance mathématique. Le numérateur et le dénominateur, sans

les opérateurs $\mathbb{E}$, ont été choisis par Bell Canada et le Conseil de la radiodiffusion et des télécommunications canadiennes (CRTC) [44]. Remarquons que nous ne savons pas comment calculer analytiquement les espérances dans $f_{\mathrm{S}}(\tau)$. Dans cette thèse, nous approximons $f_{\mathrm{S}}(\tau)$ par la simulation en estimant les espérances par des moyennes d'échantillons, et nous appelons cette fonction approximative $\hat{f}_{\mathrm{S}}(\tau)$. Cette remarque s'applique également pour les autres mesures de performance présentées dans cette section.

Notons que les problèmes d'optimisation des agents étudiés dans les chapitres 4, 5, 7 et 9 ne sont assujettis qu'aux contraintes sur les niveaux de service avec des τ constants. Dans ces chapitres, nous utiliserons les notations $g(\mathbf{y})$ et $\hat{g}(\mathbf{y})$, qui varient en fonction du vecteur d'agents $\mathbf{y}$, pour représenter $f_{\mathrm{S}}(\tau)$ et $\hat{f}_{\mathrm{S}}(\tau)$, respectivement.

Une contrainte typique sur le SL est, par exemple, que 80% des appels soient répondus dans un délai de $\tau = 20$ secondes. Étant donné que les contraintes sur le niveau de service sont souvent calculées sur une longue période de plusieurs journées ou semaines, la fonction d'un ratio d'espérances est plus appropriée pour estimer le niveau de service d'un client moyen. Si nous prenions $\mathbb{E}[X_{\mathrm{B}}(\tau)/(N - A_{\mathrm{B}}(\tau))]$, c'est-à-dire l'espérance du ratio, alors les appels durant les jours où le volume est faible auraient une pondération plus grande. La variance de cet estimateur serait aussi plus grande. Cependant, la distribution du ratio serait appropriée si nous voulons estimer la distribution du SL pour une journée.

Voici deux définitions alternatives du SL. Notons que ces alternatives ne sont pas utilisées dans les expérimentations numériques de cette thèse. La première alternative considère que les appels abandonnés ayant attendu moins ou égal à τ ont reçu un "bon service" :

$$f_{\mathrm{S2}}(\tau) = \frac{\mathbb{E}[X_{\mathrm{B}}(\tau) + A_{\mathrm{B}}(\tau)]}{\mathbb{E}[N]}. \tag{2.2}$$

Une autre définition considère tous les abandons comme des appels mal servis :

$$f_{\mathrm{S3}}(\tau) = \frac{\mathbb{E}[X_{\mathrm{B}}(\tau)]}{\mathbb{E}[N]}. \tag{2.3}$$

Ces trois définitions du SL sont disponibles dans le routeur Cisco [43]. Cependant, leur règle d'attribution de la période d'un appel est différente. Un appel est associé à : (1)

sa période de réponse s'il est répondu dans un délai inférieur ou égal à τ, (2) sa période d'abandon s'il abandonne dans un délai d'au plus τ, ou (3) la période où son temps d'attente atteint τ (peu importe qu'il soit répondu ou non plus tard). Jouini et al. [83] présentent d'autres définitions du SL.

Étant donné que le SL ne considère pas les durées d'attente des appels ayant attendu plus de τ, une politique optimale n'a aucun intérêt à répondre à ces appels. Pour éviter cette solution draconienne, un gestionnaire de centre d'appels surveille ou impose des contraintes (explicitement ou subjectivement) sur d'autres mesures de performance.

Une mesure importante est la *proportion d'abandons* qui est définie comme :

$$f_{\mathrm{A}} = \frac{\mathbb{E}[A]}{\mathbb{E}[N]}, \tag{2.4}$$

où A est le nombre total d'abandons. Les abandons signifient généralement la perte de clients potentiels.

La *proportion de délai* mesure la fraction des appels qui n'ont pas été répondus immédiatement à leur arrivée :

$$f_{\mathrm{D}} = 1 - \frac{\mathbb{E}[X_{\mathrm{B}}(0)]}{\mathbb{E}[N]}. \tag{2.5}$$

Notons que cette mesure inclut les appels bloqués.

Une mesure souvent surveillée est le *temps d'attente moyen* à long terme ou "*average waiting time*" :

$$f_{\mathrm{W}} = \frac{\mathbb{E}[W]}{\mathbb{E}[N]}, \tag{2.6}$$

où W est le temps d'attente d'un appel (servi ou abandonné). Une variante est le *temps de réponse moyen* ou "*average speed of answer*" (ASA) qui mesure le temps d'attente moyen des appels servis. Soient W_{S} le temps d'attente d'un appel servi et X le nombre total d'appels servis, le temps de réponse moyen est défini par :

$$f_{\mathrm{ASA}} = \frac{\mathbb{E}[W_{\mathrm{S}}]}{\mathbb{E}[X]}. \tag{2.7}$$

Comme mentionnée ci-haut, l'information mesurée par le SL se concentre sur les appels répondus "rapidement" avec peu de délai. Le SL indique que $x\%$ des appels ont at-

tendu moins ou égal à τ, mais il ne donne aucune information sur les temps d'attente du $(100-x)\%$ des appels qui ont attendu plus que τ. Ces délais d'attente pourraient aussi bien être de 30 secondes ou 30 minutes ! La mesure du temps d'attente moyen considère tous les délais des appels. Par contre, un inconvénient majeur à utiliser une moyenne est que différents échantillons d'observations peuvent donner la même moyenne. Supposons que $\tau = 20$ (tous les nombres sont en secondes), les échantillons des temps d'attente $\{10,10,10,10,160\}$ et $\{40,40,40,40,40\}$ donnent des niveaux de service de 80% et 0% respectivement, mais les temps d'attente moyens sont égaux à 40.

Koole [88] propose une mesure de performance appelée le *temps d'excès moyen* ou "*average excess time*" (AET) qui calcule la moyenne du temps d'attente dépassant τ :

$$f_{\mathrm{AET}}(\tau) = \mathbb{E}\left[\sum_{j=1}^{N} \max(W_j - \tau, 0)\right], \tag{2.8}$$

où W_j est le temps d'attente du j-ième appel. Cette mesure est un mélange du SL et du temps d'attente moyen. Elle calcule le temps d'attente moyen dépassant τ. Pour les deux exemples d'échantillons précédents, les AET sont 28 et 20, respectivement. Selon cette mesure, l'échantillon avec un SL de 0% est meilleur que celui avec 80% ! Malencontreusement, il n'y a pas de mesure de qualité de service parfaite. Un gestionnaire surveille généralement plusieurs mesures de performance à la fois. Les choix des mesures dépendent des besoins et des objectifs du centre d'appels.

Nous avons présenté les formules de manière générale dans cette section en considérant tous les types d'appels et toutes les périodes. Mais dépendamment du type du problème, il peut être nécessaire de calculer la qualité de service par type d'appel ou par période, ou les deux à la fois. Nous ajoutons les indices k et p pour spécifier le type d'appel k et la période p, par exemple $f_{\mathrm{S},k,p}$ et $\tau_{k,p}$. L'omission des indices k ou p est évidente selon le contexte du problème.

Il n'y a pas seulement la qualité de service des clients qui est importante. L'efficacité d'un centre d'appels est souvent mesurée par le taux d'occupation des agents. Soient y_i le nombre d'agents dans le groupe i, T le temps d'horizon couvert par la mesure et $G_i(t) \leq y_i$ le nombre d'agents du groupe i occupés au temps $0 \leq t \leq T$. Le *taux d'occupation* est

défini par la proportion moyenne d'agents occupés durant la période de longueur T :

$$f_{\mathrm{O},i} = \frac{1}{y_i T}\mathbb{E}\left[\int_0^T G_i(t)dt\right]. \tag{2.9}$$

Lorsque l'effectif d'agents varie en fonction du temps, le taux d'occupation est calculé par la formule suivante :

$$f_{\mathrm{O2},i} = \frac{1}{\int_0^T y_i(t)dt}\mathbb{E}\left[\int_0^T G_i(t)dt\right], \tag{2.10}$$

où $y_i(t)$ est le nombre d'agents dans le groupe i au temps t.

Un taux d'occupation qui est bas signifie généralement que le centre d'appels emploie trop d'agents. En théorie, il est possible d'offrir une bonne qualité de service aux clients tout en ayant un taux d'occupation presque à 100% dans un régime à trafic intense (ou "*heavy traffic*"). Ceci s'explique par le phénomène d'*économie d'échelle* : un grand centre d'appels (fort volume d'appels et beaucoup d'agents) nécessite un plus petit ratio volume/agents pour avoir la même performance qu'un petit centre d'appels. Autrement dit, un grand centre d'appels opère plus efficacement qu'un petit centre d'appels. Par contre, dans la réalité, un taux d'occupation élevé augmente le stress et la fatigue chez l'agent. Un agent fatigué aura tendance à étirer les durées de service ou à allonger les temps de post-traitement à la fin des appels. Ces comportements ne réduisent pas le niveau d'occupation, car les agents demeurent occupés, mais ils diminuent l'efficacité. Notons que, dans cette thèse, nous ne modélisons pas les temps de post-traitement des appels dans nos problèmes. Cependant, il est facile d'intégrer cet aspect en ajoutant un temps de post-traitement aléatoire entre la fin de service et avant que l'agent devient libre. Un gestionnaire vise habituellement un taux d'occupation entre 85% et 95%. Il y a des exceptions, comme les services d'urgence, où suffisamment d'agents doivent être disponibles pour répondre aux appels avec pratiquement aucun délai.

Dans la réalité, les agents servent les appels à des vitesses différentes. Il y a plusieurs facteurs possibles tels l'expérience et le nombre de types d'appels à servir. Les agents spécialistes ont tendance à servir plus rapidement que les agents généralistes. (Les agents doivent souvent respecter un barème sur la durée de service pour chaque type d'appel.) Un

gestionnaire pourrait être motivé à maximiser le *taux d'appariement* (ou "*match rate*") qui mesure la proportion des appels du type k servis par leur groupe d'agents de préférence :

$$f_{\mathrm{M},k} = \frac{\mathbb{E}[S_{k,\hat{i}_k}]}{\mathbb{E}\left[\sum_{i=1}^{I} S_{k,i}\right]}, \tag{2.11}$$

où $\hat{i}_k$ est le groupe de préférence du type d'appel k et $S_{k,i}$ est le nombre d'appels de type k servis par le groupe i. En pratique, un gestionnaire attribue souvent un groupe de préférence par type d'appel, mais rien n'empêche d'avoir plus qu'un groupe. Le dénominateur pourrait être le nombre d'arrivées au lieu du nombre total d'appels servis. Le taux d'appariement est souvent relié à la mesure de *résolution au premier appel* (ou "*first call resolution*"), définie par la proportion des clients ayant obtenu un service satisfaisant dès leur premier appel. Nos problèmes et ceux présents dans la littérature supposent ordinairement que les clients ne rappellent pas le centre d'appels (même si abandonnés ou bloqués). Or, dans la vraie vie, les rappels peuvent représenter une portion significative du volume d'appels. Un gestionnaire serait alors motivé à augmenter le taux d'appariement aux agents les plus compétents pour chaque type d'appel afin de réduire le nombre de rappels des clients.

CHAPITRE 3

REVUE DE LA LITTÉRATURE

Nous présentons dans ce chapitre une revue de la littérature sur les centres d'appels. Akşin et al. [2] et Gans et al. [58] font un survol des études de recherche sur les centres d'appels, et présentent une longue liste de publications scientifiques. Bien que ces articles couvrent une grande variété de sujets, nous porterons principalement notre attention sur les problèmes de gestion de la main-d'œuvre et de routage pour les centres d'appels multi-compétences.

3.1 Modélisation et prévision des volumes d'appels

Il y a deux étapes importantes avant l'optimisation : la modélisation du centre d'appels et les prévisions des volumes d'appels. Cependant, nous ne donnerons qu'un bref résumé puisque ce ne sont pas les sujets principaux de cette thèse.

En général, l'industrie suppose que les processus d'arrivée des appels sont Poisson et utilise de simples techniques de régression ou des séries chronologiques afin de produire les prévisions des volumes. Une pratique courante est la méthode "*top-down*" qui consiste à répartir la prévision agrégée des appels à travers les périodes de la journée proportionnellement aux distributions historiques des appels [58]. Par exemple, si nous disposons

seulement de la prévision pour la journée complète (et non pour chaque période), alors les taux d'arrivées par période sont répartis selon les proportions historiques moyennes des appels pour chaque période (ex. : 3% des appels pour la période 1, 5% dans la période 2, etc.).

Gans et al. [58] mentionnent que le nombre de recherches est limité dans ce domaine, mais quelques articles ont été publiés récemment. À ce jour, les modèles de prévision par régression standard sont difficiles à battre au-delà d'un horizon de 2 semaines. Cependant, il existe de nombreux modèles pouvant donner de meilleures prévisions à court terme. Plusieurs études ont examiné les méthodes de séries chronologiques, en particulier les méthodes auto-régressives comme ARIMA, sur des données de centres d'appels réels [4, 40, 118, 125]. Ces modèles examinent les effets de différents facteurs sur les volumes d'appels tels que les jours de la semaine, les mois, les saisons, les journées de fêtes, les campagnes de publicité, etc. Tych et al. [129] examinent les modèles basés sur les régressions harmoniques. Soyer et Tarimcilar [121], Weinberg et al. [133] et Gans et al. [59] proposent des méthodes basées sur des modèles d'analyses bayésiennes. Taylor [125, 126] étudie des méthodes de lissage exponentielle. Aldor-Noiman et al. [3] et Ibrahim et L'Ecuyer [80] proposent des modèles à effets mixtes. Channouf et L'Ecuyer [39] modélisent la dépendance des volumes d'appels entre les périodes par une copule normale.

Il est rare que le taux d'arrivée des appels soit constant durant toute la journée. Un centre d'appels est un système non stationnaire tel que l'intensité des volumes d'appels et les nombres d'agents varient en fonction du temps, et les files d'attente se vident à la fermeture du centre. La pratique courante divise une journée en plusieurs périodes (de 30 minutes par exemple) et estime chaque période comme étant un processus Poisson homogène où le taux d'arrivée est constant par période. Brown et al. [26] ont étudié les données d'un centre d'appels d'une banque et ils ne rejettent pas l'hypothèse nulle d'un processus Poisson homogène par période. Cependant, Green et Kolesar [67] observent qu'il serait dangereux d'utiliser cette méthode d'approximation pour estimer les mesures de performance quand les variations de volumes sont significatives. Néanmoins, les méthodes de planification courantes utilisent les formules d'approximation d'Erlang, qui suppose un système markovien stationnaire, à cause de leur simplicité et rapidité de calcul. Un centre

d'appels est cependant loin d'être un système stationnaire. Par exemple, la période d'ouverture d'un centre a souvent un niveau de service plus élevé que celui estimé par les formules d'Erlang, car la file d'attente est vide au départ. De plus, les formules d'Erlang ne peuvent évaluer que pour un centre avec un seul type d'appel et un groupe d'agents. Plus récemment, Gans et al. [59] intègrent le processus de prévision avec l'optimisation stochastique des agents.

Certaines études ont examiné différents processus d'arrivée, car des observations de données réelles indiquaient que la variance du nombre d'appels était trop élevée pour être un processus Poisson homogène. Plusieurs proposent un processus Poisson doublement stochastique, en particulier un processus Poisson-gamma [19, 50, 82, 135]. Ce modèle d'arrivée suppose un processus Poisson homogène par période avec un taux d'arrivée aléatoire. Dans le cas d'un processus Poisson-gamma, le taux d'arrivée est une variable aléatoire de loi gamma. Par exemple, le taux d'arrivée pourrait avoir la forme suivante : $\Lambda(t) = W\lambda(t)$ où $\lambda(t)$ est constant par période et W est une variable aléatoire gamma. Avramidis et al. [19] observent une forte corrélation positive sur les volumes entre les périodes d'une même journée. Un fort volume d'appels durant l'avant-midi suggère vraisemblablement un fort volume en après-midi, et dans le cas opposé également. La variable W, de moyenne 1, représenterait le facteur d'achalandage (ou "*busyness factor*") de la journée. Cette variable pourrait dépendre des facteurs mentionnés plus tôt tels que le jour de la semaine, le mois, etc.

D'autre part, des études d'analyses statistiques récentes observent que le temps de service d'un appel ne suit pas une loi exponentielle telle que présumé dans le système markovien $M/M/n$, sur lequel se basent les formules d'Erlang. La fonction de densité d'une variable exponentielle est strictement décroissante, ainsi une plus forte probabilité d'avoir un temps de service proche de 0. Ce n'est généralement pas le cas. Brown et al. [26] et Deslauriers [50] proposent plutôt une loi log-normale.

3.2 Évaluation des mesures de performance d'un centre d'appels

Le gestionnaire doit planifier le nombre d'agents de manière à respecter certaines qualités de service. Les mesures de performance sont des fonctions complexes, et les algorithmes d'optimisation doivent recourir à des méthodes d'approximation ou la simulation. Le domaine des files d'attente dispose d'une énorme quantité de résultats scientifiques, en particulier pour les systèmes avec un type de client et un type de serveur, voir Wolff [138].

3.2.1 Formules d'approximation d'Erlang

Plusieurs gestionnaires utilisent encore les formules d'Erlang dans leur procédure de planification à cause de la simplicité de leur implémentation. Les formules d'Erlang [45, 111] approximent les performances d'un processus markovien stationnaire $M/M/n$ (processus stochastiques exponentiels et n serveurs homogènes). Soient λ le taux d'arrivée, μ le taux de service et $\rho = \lambda/\mu$ la *charge de trafic*.

La formule d'Erlang B calcule la probabilité de blocage quand il n'y a aucune file d'attente (c'est-à-dire qu'un client abandonne immédiatement s'il ne trouve aucun agent libre à son arrivée) :

$$B(\rho, n) = \frac{\rho^n/n!}{\sum_{j=0}^{n}(\rho^j/j!)}, \qquad \text{avec } n > \rho. \tag{3.1}$$

La formule d'Erlang C considère un système $M/M/n$ sans abandon avec une file d'attente de capacité infinie. La probabilité de délai est :

$$C(\rho, n) = \frac{nB(\rho, n)}{n - \rho(1 - B(\rho, n))}, \tag{3.2}$$

et le SL est :

$$\mathbb{P}[W \leq \tau] = 1 - C(\rho, n)e^{\tau(\lambda - n\mu)}, \qquad \text{avec } t \geq 0 \text{ et } n > \rho, \tag{3.3}$$

où W est le temps d'attente d'un client et τ est l'AWT. À cause de la propriété PASTA ("*Poisson Arrivals See Time Averages*"), voir Wolff [137], la probabilité de blocage (ou de

délai) correspond exactement à la proportion d'appels bloqués (ou ayant subit un délai).

La formule d'Erlang A, aussi appelée la formule de Palm [111], évalue le SL d'un système $M/M/s+M$ où le temps de patience d'un client est une variable exponentielle de moyenne $1/\nu$. La formule proposée par Palm est plus complexe à calculer, et elle contient une somme d'une alternance de coefficients 1 et -1, ce qui rend le calcul numériquement instable. Nous présentons la formule dérivée par Richard Simard dans l'article de Deslauriers et al. [51] qui est numériquement stable. La probabilité que le temps d'attente W d'un nouvel appel soit plus grand que τ quand il voit j appels déjà en attente est :

$$p_j(\mu,\tau) = \mathbb{P}[W > \tau | X = n+j] = \xi^{\phi} \sum_{z=0}^{j} \frac{(\phi)_z (1-\xi)^z}{z!}, \tag{3.4}$$

où $\phi = n\mu/\nu$, $\xi = e^{-\nu\tau}$ et les symboles de Pochhammer : $(\phi)_0 = 1$ et $(\phi)_z = (\phi)(\phi+1)\cdots(\phi+z-1)$ pour $z \geq 1$. Avec la propriété PASTA, l'équation (3.4) et une capacité de file d'attente C, la probabilité que le temps d'attente W d'un appel soit au plus τ est :

$$\mathbb{P}[W \leq \tau] = 1 - \sum_{j=0}^{C-1} p_j(\mu,\tau)\pi_{n+j} + \pi_{n+C}, \tag{3.5}$$

où π_x est la probabilité stationnaire d'avoir x appels dans la CMTC associée au système $M/M/n+M$. Les probabilités stationnaires π_x sont obtenues en résolvant un système d'équations linéaires, qu'on retrouve dans plusieurs ouvrages classiques dont Hillier et Lieberman [77] et Ross [115]. Une variante de la formule (3.5) serait de compter les abandons immédiats comme des appels ayant reçu de mauvais service, et de remplacer le terme $+\pi_{n+C}$ par $-\pi_{n+C}$. Notons que la formule (3.5) peut tout de même avoir des erreurs numériques, car les valeurs des π_x deviennent très petites quand le nombre d'états est très grand.

Garnett et al. [61] présentent une approximation de la formule d'Erlang A pour un régime à trafic intense (taux d'arrivée et nombre d'agents très grands). Leur formule d'approximation est tractable et elle se calcule numériquement mieux que la formule (3.5) dans ce régime à trafic intense. Par contre, leur approximation est moins précise quand les paramètres du système $M/M/n+M$ sont petits. En général, nous suggérons d'utiliser

l'approximation de Garnett et al. [61] lorsque $n \geq 500$ (une valeur conservatrice).

3.2.2 Autres modèles d'approximation

Les formules classiques s'appliquent difficilement aux modèles de centres d'appels modernes avec plusieurs types d'appels et groupes d'agents, car le routage influence significativement les mesures de performance. De plus, le nombre d'états dans la chaîne de Markov grandit exponentiellement en fonction du nombre de types d'appels et du nombre de groupes d'agents. L'analyse exacte de la chaîne de Markov devient rapidement impraticable. Quelques approximations ont toutefois été proposées pour le modèle N (2 types d'appels et 2 groupes d'agents) par Stanford et Grassmann [122], Shumsky [119], Deslauriers et al. [51], et Koole et al. [90]. Le deuxième article présente un algorithme de décomposition pour accélérer le temps de calcul en fractionnant la chaîne de Markov. Le troisième article ajoute des transitions dans la chaîne de Markov pour mesurer le temps d'attente de l'appel à la tête de la file, ce qui permet d'avoir une politique de routage avec des temps de délai. Le quatrième considère un type d'appel entrant et un type d'appel sortant.

À ce jour, il n'existe aucune méthode d'approximation efficace pour les centres d'appels modernes avec plus que 2 ou 3 types d'appels. Cependant, il existe plusieurs méthodes d'approximation relativement précises pour le cas particulier où il n'y a aucune file d'attente avec une politique de routage par débordement (ou "*overflow routing*"). La politique de débordement est un cas spécial de la politique par priorités, mais sans priorité égale. Ces méthodes se basent sur la décomposition exponentielle [92, 93], la décomposition hyper-exponentielle [54], les méthodes d'équivalence aléatoire ou "*equivalent random methods*" (ERM) [45] et la méthode de Hayward-Fredericks (une extension de ERM) [55, 138]. Étant donné qu'il n'y a pas de file d'attente, ces approximations mesurent la probabilité de blocage des appels. Nous observons souvent une vague corrélation entre le SL et la probabilité de blocage, mais il n'est pas évident de déterminer la mesure de l'un à partir l'autre. Par ailleurs, Avramidis et al. [18] se basent sur la décomposition exponentielle et approximent un centre d'appels avec un routage particulier. Les appels suivent une politique de routage par débordement, et s'ils ne trouvent d'agents libres à leur arrivée, alors ils aboutissent dans une file d'attente qui est servie uniquement par le dernier groupe d'agents de la sé-

quence de débordement. Chaque type d'appel peut suivre une séquence de débordement différente, ce qui rend l'approximation non triviale. Malgré que ces méthodes d'approximation ont souvent des erreurs significatives, elles sont plus rapides que la simulation et peuvent se montrer utiles dans un algorithme d'optimisation, par exemple pour comparer les solutions dans un voisinage [18, 109].

Pour obtenir de bonnes estimations, il est pratiquement nécessaire d'utiliser un logiciel de simulation. Évidemment, il faut avoir au préalable une bonne modélisation du centre d'appels. En général, les gestionnaires achètent des simulateurs commerciaux tels que Arena [114]. Ces logiciels facilitent l'utilisation à l'aide d'interfaces graphiques, mais ils sont lents et s'intègrent difficilement dans les programmes d'optimisation. Il est possible d'implémenter un simulateur sans trop de difficulté pour des modèles très simples, mais la complexité augmente rapidement. Dans notre cas, nous utilisons la librairie de simulation ContactCenters [31] en Java.

3.3 Méthodes de planification des agents

La gestion des agents dans un centre d'appels était auparavant un domaine de recherche peu actif, mais plusieurs études sont apparues depuis la dernière décennie. Nous nous concentrerons sur les principales méthodes de planification proposées dans la littérature. Ces méthodes sont des heuristiques et il y a encore place à beaucoup d'améliorations dans ce domaine.

Plusieurs problèmes de planification étudiés sont assujettis aux contraintes sur les niveaux de service $f_S(\tau)$, définis à la section 2.3, page 20, puisque ce sont des mesures de performance les plus couramment utilisées en pratique. Les temps d'attente acceptables τ et τ_p étant constants dans ces problèmes, nous redéfinissons $f_S(\tau)$ par $g(\mathbf{y})$ qui varie en fonction du vecteur d'agents $\mathbf{y}$.

3.3.1 Centre avec un type d'appel et un groupe d'agents

Méthode de planification stationnaire indépendante par période (SIPP)

La méthode de planification traditionnelle, et encore couramment utilisée dans l'industrie des centres d'appels, est la méthode stationnaire indépendante par période ou "*stationary independent period by period*" (SIPP) telle que nommée par Green et al. [68]. Le problème classique considère un centre avec un seul type d'appel et un groupe d'agents. L'objectif est de minimiser le nombre d'agents à affecter à chaque période tout en respectant les SL. Soient P le nombre de périodes, y_p le nombre d'agents à la période p, le vecteur de décision $\mathbf{y} = (y_1, \ldots, y_P)^{\mathrm{T}}$, $g_p(\mathbf{y})$ le SL à la période p avec l'objectif correspondant l_p. Le problème est :

$$\begin{aligned} \min \quad & \textstyle\sum_{p=1}^{P} y_p \\ \text{sujet à :} \quad & g_p(\mathbf{y}) \geq l_p, \quad p = 1, \ldots, P, \\ & \mathbf{y} \geq 0 \text{ et entier.} \end{aligned}$$

La fonction $g_p(\mathbf{y})$ représente la performance d'un système non stationnaire, puisque l'état du système dépend des périodes précédentes. L'approche SIPP suppose que les périodes sont indépendantes et elle divise le problème principal en P sous-problèmes. Elle suppose aussi que chaque période est un processus markovien stationnaire $M/M/n$. La fonction $g_p(\mathbf{y})$ est estimée par la formule d'Erlang C ou Erlang A. Le taux d'arrivée utilisé dans la formule d'Erlang est le taux d'arrivée moyen de la période.

Cette méthode peut être étendue pour optimiser les quarts de travail. Ceci représente la méthode traditionnelle de planification à deux étapes. La première étape consiste à trouver le nombre d'agents requis par période, disons $\hat{y}_p$. La deuxième étape est le problème classique de planification originairement formulé par Dantzig [47] :

$$\begin{aligned} \min \quad & \textstyle\sum_{s=1}^{S} c_s x_s \\ \text{sujet à :} \quad & \mathbf{A}\mathbf{x} \geq \hat{\mathbf{y}}, \\ & \mathbf{x} \geq 0 \text{ et entier,} \end{aligned}$$

où S représente le nombre de quarts de travail, x_s est le nombre d'agents affectés au quart de travail s avec un coût de c_s par agent, $\mathbf{x} = (x_1, \ldots, x_S)^{\mathrm{T}}$, $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_P)^{\mathrm{T}}$, et $\mathbf{A}$ est

une matrice binaire, de taille $P \times S$, telle que $a(p,s) = 1$ si le quart de travail s couvre la période p et $a(p,s) = 0$ sinon. Mentionnons que l'article a été publié en 1954, bien avant le développement des techniques d'optimisation à nombres entiers, et l'auteur propose d'optimiser par des variables continues et d'arrondir les solutions fractionnaires. Diverses études de cas ont été publiées [5, 27, 78, 87].

À moins d'avoir des horaires très flexibles, il est fort probable que la solution ne coïncide pas exactement avec le nombre d'agents requis par période, et qu'il y ait des surplus d'employés. Keith [85] ajoute des variables de surplus et de pénurie d'employés pour chaque période. Ces variables permettent de mieux répartir le niveau de surplus entre les périodes. Un gestionnaire peut offrir des congés impayés lorsqu'il y a plusieurs périodes avec surplus d'employés, ou demander de faire du temps supplémentaire quand il y a une courte pénurie. L'auteur résout le problème en deux étapes : (1) optimiser la version continue du problème, puis (2) utiliser une méthode heuristique, qui ajoute ou retire un employé à la fois, pour obtenir une solution entière.

Thompson [128] ajoute des contraintes sur les SL, par période et agrégé sur toutes les périodes, au problème de planification de Dantzig [47]. Il suppose que le SL agrégé est plus important que ceux par période, par exemple une cible de 80% pour la semaine et une cible de 50% pour chaque période. La méthode proposée requiert l'évaluation, faite au préalable, des SL pour différents nombres d'employés, afin d'incorporer l'impact de chaque agent additionnel au modèle mathématique. Les périodes sont considérées indépendantes les unes des autres. L'auteur optimise le problème par une méthode de recuit simulé.

Amélioration de la méthode SIPP

Dans la réalité, l'intensité du volume d'appels varie continuellement dans le temps. Green et al. [68] présentent certaines difficultés rencontrées par la méthode SIPP. Ils observent que les SL d'une solution obtenue par cette méthode fluctuent énormément lorsque l'intensité des arrivées change rapidement avec une forte amplitude, et que les périodes sont courtes. Dans ces conditions, il y a souvent un effet de décalage entre le nombre d'arrivées et les SL. Ces erreurs peuvent alors donner des périodes avec des SL insuffisants. La figure 3.1 montre un exemple typique de l'intensité du volume d'appels dans une journée.

Le volume est généralement plus bas le matin et le soir, et il fluctue durant la journée.

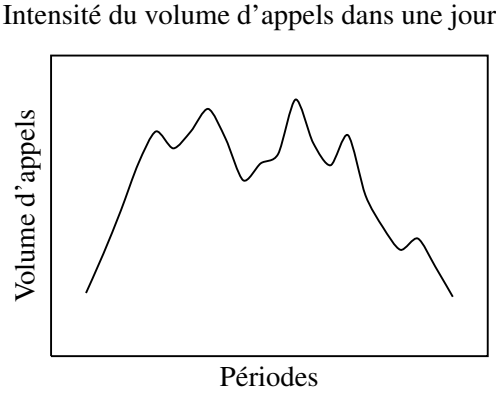


Figure 3.1 – Exemple d'une courbe du volume d'appels dans une journée.

Green et al. [68] proposent 3 variantes de la méthode SIPP :

- **SIPP Max :** Soit $[t_p, t_{p+1}]$ l'intervalle de temps de la période p. Cette méthode utilise le taux maximal : $\lambda = \max\{\lambda(t) : t \in [t_p, t_{p+1}]\}$ au lieu du taux moyen de la période. Elle améliore les SL, mais peut causer une sur-affectation d'agents. Par contre, la même difficulté que SIPP survient lorsque les périodes sont courtes et les taux varient avec de grandes amplitudes.
- **SIPP Mix :** Pour éviter le surplus d'agents de SIPP Max, cette variante utilise SIPP classique pour les périodes avec un taux d'arrivée décroissant et SIPP Max lorsque le taux est croissant. Par contre, les expériences numériques montrent que SIPP Mix n'est pas meilleure que SIPP Max en général.
- **SIPP Lag :** Cette variante s'attaque directement à l'effet de décalage en décalant les taux d'arrivée. Elle estime le temps de décalage L. Pour calculer le nombre d'agents nécessaires à la période p définie sur $[t_p, t_{p+1}]$, le taux d'arrivée λ est calculé sur l'intervalle $[t_p - L, t_{p+1} - L]$. Les expériences numériques montrent que cette méthode est généralement meilleure que SIPP classique. Nous pouvons mixer les méthodes et obtenir SIPP Lag Max qui est plus conservateur.

Les méthodes SIPP sont rapides, mais elles ne sont utiles que pour les centres avec un seul groupe d'agents et il faudrait utiliser la simulation pour corriger les solutions. La méthode SIPP peut être trop contraignante, car elle impose une contrainte sur le SL pour

chaque période. Dans certains centres d'appels, les véritables contraintes sont sur les SL agrégés sur un horizon d'une journée, de semaines ou même des mois. Il est possible qu'il n'y a aucune contrainte en réalité pour les périodes individuelles de la journée.

Planification sans contraite par période basée sur SIPP

Saltzman et Mehrotra [117] étudient le problème de planification basé sur un vrai centre d'appels de Expedia.com. Les contraintes sont sur les temps d'attente moyens et les abandons mesurés sur un horizon d'une semaine. La méthode SIPP donne des solutions trop coûteuses, car elle optimise de manière à satisfaire les contraintes à chaque période, alors qu'il n'est pas nécessaire dans ce problème. Les auteurs proposent l'algorithme "*Weekly Staffing Plan Routine*" (WSPR) qui se base sur SIPP.

Les contraintes sur la qualité de service sont exprimées par des coûts de pénalités. Soient P le nombre de périodes, S le nombre de quarts de travail, h_s le nombre d'heures de travail dans le quart s, c_s le coût par heure du quart s, x_s le nombre d'agents affectés au quart s, $\mathbb{E}[N_{\mathrm{A}}]$ le nombre moyen d'abandons, C_{A} le coût par abandon, $\mathbb{E}[N_{\mathrm{W}}]$ le nombre moyen d'appels en attente par période et C_{W} le coût par appel qui subit un délai. Le problème est :

$$\begin{aligned} \min \quad & f_1(\mathbf{x}) = \textstyle\sum_{s=1}^{S} h_s c_s x_s + C_{\mathrm{A}}\mathbb{E}[N_{\mathrm{A}}] + PC_{\mathrm{W}}\mathbb{E}[N_{\mathrm{W}}] \\ \text{sujet à :} \quad & \mathbf{x} \geq 0 \text{ et entier.} \end{aligned} \tag{W}$$

L'algorithme se résume comme suit. Dans l'initialisation, la méthode SIPP est utilisée pour trouver le nombre d'agents requis pour chaque période, disons $\mathbf{b} = (b_1, \ldots, b_P)^{\mathrm{T}}$. La méthode consiste à affecter les quarts de travail en fonction du nombre total d'heures payées T. Pour un nombre d'heures $T \in [T_{\min}, T_{\max}]$, elle résout le problème suivant pour trouver les horaires $\mathbf{x}$:

$$\begin{aligned} \min \quad & \textstyle\sum_{p=1}^{P} |z_p - b_p| \\ \text{sujet à :} \quad & \mathbf{A}\mathbf{x} = \mathbf{z}, \\ & \textstyle\sum_{s=1}^{S} h_s x_s \leq T, \\ & \textstyle\sum_{s=v+1}^{S} x_s \leq T^{\mathrm{q}}_{\max}, \\ & \mathbf{x} \geq 0 \text{ et entier.} \end{aligned} \tag{W(T)}$$

La définition de la matrice **A** est identique à celle des sections précédentes. Les auteurs étudient également l'avantage d'avoir des agents à temps partiel (demi-journée de travail). Toutefois, les conventions de travail exigent souvent de donner préférence aux employés à temps plein. La contrainte $T^{\mathrm{q}}_{\max}$ représente le maximum d'heures de travail pour les agents à temps partiel. Les quarts de travail à temps plein sont numérotés de 1 à v, et ceux à temps partiel ont les indices $v+1$ à S. Contrairement à la méthode SIPP, les horaires **x** obtenus par ce sous-problème ne sont pas contraints à couvrir les nombres d'agents requis **b**. Soit $\mathbf{x}(T)$ la solution optimale de $(\mathrm{W}(T))$, les espérances $\mathbb{E}[N_{\mathrm{A}}]$ et $\mathbb{E}[N_{\mathrm{W}}]$ sont calculées à l'aide de la simulation afin d'estimer le coût $f_1(\mathbf{x}(T))$. La solution optimale $\mathbf{x}^*$ du problème (W) est celle qui possède le coût minimal pour f_1 parmi les solutions $\mathbf{x}(T_{\min})$ à $\mathbf{x}(T_{\max})$. Par la suite, l'algorithme applique une recherche heuristique utilisant une méthode tabou. Puisque les réductions de coûts sont faibles et la recherche locale demande un temps d'exécution considérable, l'algorithme de recherche n'est pas détaillé par les auteurs.

Dans leur étude de cas, leur problème est défini sur un horizon du lundi au vendredi de 6:00 à 21:00. Les expériences numériques basées sur des données réelles montrent que la méthode WSPR trouve une solution significativement moins coûteuse que SIPP avec 30% moins d'agents. Avec des choix de 49 quarts de travail différents (26 temps pleins et 23 temps partiels) pour 300 périodes, SIPP planifie beaucoup trop d'agents, alors que WSPR reste proche du nombre d'agents requis.

Optimisation par simulation et programmation linéaire

Atlason et al. [12] présentent un algorithme qui optimise la planification des agents en une seule étape à l'aide de la simulation et la programmation linéaire. La simulation permet d'éviter les erreurs des approximations stationnaires en simulant les journées entières, mais augmente significativement le temps d'optimisation. Leur algorithme se base sur la méthode de coupes par sous-gradients de Kelley Jr. [86] pour les problèmes avec une fonction objectif et un domaine de solutions convexes.

Le problème consiste à minimiser le coût des agents sous contrainte de respecter des

seuils minimaux l et l_p pour les périodes $p = 1, \ldots, P$. Le problème à résoudre est :

$$\begin{aligned} \min \quad & \textstyle\sum_{s=1}^{S} c_s x_s \\ \text{sujet à :} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{y}, \\ & g_p(\mathbf{y}) \geq l_p, \quad p = 1, \ldots, P, \\ & g(\mathbf{y}) \geq l, \\ & \mathbf{x}, \mathbf{y} \geq 0 \text{ et entiers.} \end{aligned} \tag{P0}$$

Le problème original (P0) contient des termes non linéaires, ce qui rend l'optimisation difficile. L'idée de cette méthode consiste à remplacer ces contraintes difficiles par un ensemble de contraintes linéaires afin d'utiliser la programmation linéaire.

Cette approche se base sur la propriété de concavité de la fonction g pour une file d'attente $M/M/n$ sans abandon [81]. Dans le cas avec abandon, la fonction g a une courbe en forme d'un "S" étiré (g est concave à partir d'un certain nombre d'agents), voir la figure 3.2. Par contre, il n'est pas sûr que g soit concave dans le cas d'un centre d'appels multi-périodes. Notons que $g(\mathbf{y})$ est une fonction discrète définie sur le domaine des entiers naturels $\mathbb{N}^P$. Nous pouvons dire que g est concave si l'ensemble de points $\{(\mathbf{y}, g(\mathbf{y}))\}$ forme une enveloppe convexe sur tout $\mathbf{y} \in \mathbb{N}^P$. Si g est concave, alors il existe au moins un sous-gradient (il peut en exister une infinité puisque g est une fonction discrète) à tout point.

Supposons que $g(\bar{\mathbf{y}}) < l$ pour une solution $\bar{\mathbf{y}}$, alors par la propriété de concavité et le fait que g est non décroissante :

$$g(\bar{\mathbf{y}}) + \mathbf{q}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \bar{\mathbf{y}}) \geq g(\mathbf{y}),$$

où $\mathbf{q}(\bar{\mathbf{y}})$ représente un *sous-gradient* de g au point $\bar{\mathbf{y}}$. Nous utilisons le terme sous-gradient, car la fonction g est discrète et non différentiable. Nous cherchons une solution $\mathbf{y}$ telle que :

$$g(\bar{\mathbf{y}}) + \mathbf{q}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \bar{\mathbf{y}}) \geq g(\mathbf{y}) \geq l,$$

et nous obtenons l'équation linéaire :

$$\mathbf{q}(\bar{\mathbf{y}})^{\mathrm{T}}\mathbf{y} \geq \left(l - g(\bar{\mathbf{y}}) + \mathbf{q}(\bar{\mathbf{y}})^{\mathrm{T}}\bar{\mathbf{y}}\right). \tag{3.6}$$

Les auteurs utilisent la méthode des différences finies de longueur 1 pour estimer le sous-gradient $\mathbf{q}$, où $q_i(\bar{\mathbf{y}}) = g(\bar{\mathbf{y}} + \mathbf{e}_i) - g(\bar{\mathbf{y}})$ et $\mathbf{e}_i$ est le vecteur unitaire avec 1 en position i. Observons que tous les termes à part $\mathbf{y}$ dans l'équation 3.6 sont connus. La figure 3.2 montre un exemple simple à une dimension d'une génération de coupe par un sous-gradient. Dans un cas à plusieurs dimensions, le sous-gradient $\mathbf{q}$ est un hyperplan.

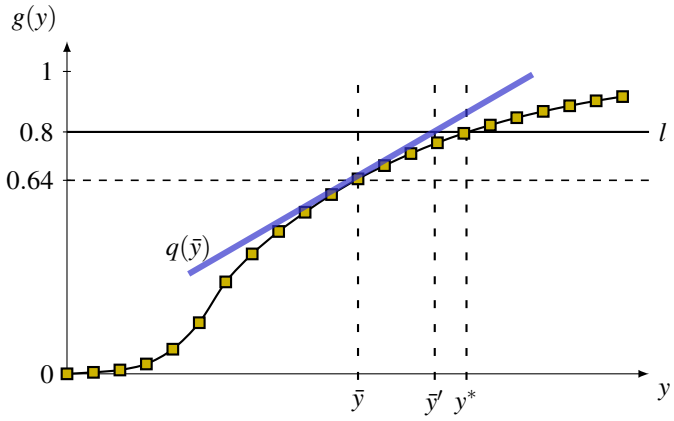


Figure 3.2 – Exemple d'une génération de coupe par sous-gradient où la solution courante est $\bar{y}$, la solution optimale est y^* et la nouvelle contrainte est $y \geq \bar{y}'$.

Puisque nous ne connaissons pas exactement la fonction g, il faut l'approximer à l'aide de la simulation, disons par $\hat{g}$. Nous remplaçons g par $\hat{g}$ dans (P0) et appelons ce **problème (P1)**. Cependant, au lieu de résoudre directement (P1), cette méthode résout itérativement un problème relaxé, disons (P2), où les contraintes sur $\hat{g}$ sont remplacées par un ensemble de coupes linéaires basées sur les sous-gradients. Les problèmes linéaires sont plus faciles à optimiser que les problèmes non linéaires, en particulier si les variables entières sont

relaxées en variables continues. Soit le problème (P2) à l'itération v :

$$\begin{aligned} \min \quad & \textstyle\sum_{s=1}^{S} c_s x_s \\ \text{sujet à :} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{y}, \\ & \mathbf{B}^{(v)}\mathbf{y} \geq \mathbf{b}^{(v)}, \\ & \mathbf{x}, \mathbf{y} \geq 0 \text{ et entiers}, \end{aligned} \tag{P2$^{(v)}$}$$

où la paire $(\mathbf{B}^{(v)}, \mathbf{b}^{(v)})$ représente un ensemble de contraintes. Voici un résumé de l'algorithme :

Étape 1 Initialiser $(\mathbf{B}^{(1)}, \mathbf{b}^{(1)})$ vide ou avec une coupe initiale et $v = 1$.

Étape 2 Résoudre le problème linéaire (P2$^{(v)}$) pour obtenir une solution $(\mathbf{x}^{(v)}, \mathbf{y}^{(v)})$.

Étape 3 Simuler la solution. Terminer l'algorithme si la solution est réalisable pour (P1).

Étape 4 Estimer le sous-gradient $\mathbf{q}(\mathbf{y}^{(v)})$ et ajouter la coupe linéaire à $(\mathbf{B}^{(v)}, \mathbf{b}^{(v)})$, si le présumé sous-gradient passe un test de concavité heuristique (pour vérifier que le sous-gradient est partout supérieur à g).

Étape 5 $(\mathbf{B}^{(v+1)}, \mathbf{b}^{(v+1)}) = (\mathbf{B}^{(v)}, \mathbf{b}^{(v)})$ et $v = v + 1$. Retourner à l'étape 2.

Étant donné que cette méthode utilise la simulation, elle résout une version approximée par une moyenne d'échantillons (ou "*sample average approximation*") du problème. Les bruits de simulation lors des estimations des sous-gradients sont réduits en ré-utilisant les mêmes échantillons tout au long de l'optimisation. Ceci correspond à la technique de réduction de variance par variables aléatoires communes (VAC).

Cette méthode converge vers la solution optimale du problème original (P0) si : (1) g est une fonction concave (et non décroissante), et (2) les simulations sont suffisamment longues. Les difficultés sont : (1) il est difficile de vérifier si g est concave, (2) il faut faire $P + 1$ simulations à chaque itération et (3) il faut de longues simulations pour avoir des estimations sans bruit.

3.3.2 Centre avec plusieurs types d'appels et groupes d'agents

Le problème de planification devient beaucoup plus complexe lorsqu'il y a plusieurs types d'appels et groupes d'agents qui partagent des habiletés en commun. Dans cette section, nous supposons K types d'appels, I groupes d'agents, et $\mathcal{S}_i \subseteq \{1,2,\ldots,K\}$ est l'ensemble des types d'appels pouvant être servis par le groupe i. Il pourrait avoir jusqu'à $2^K - 1$ groupes différents, ou même plus s'ils ont des routages différents. De plus, le routage a beaucoup plus d'influence sur la solution dans ce type de problème que dans celui avec un seul type d'appel. Il n'y a que la simulation qui peut évaluer précisément la performance de ces centres d'appels, et la plupart des méthodes que nous présentons ici utiliseront la simulation à un certain moment. Nous commençons par les méthodes qui optimisent pour une seule période, puis celles pour plusieurs périodes.

Optimisation par simulation et programmation linéaire

Cezik et L'Ecuyer [37] présentent une adaptation et des extensions à l'algorithme de coupes par sous-gradients de Atlason et al. [12] pour les centres d'appels multi-compétences. Ils optimisent l'affectation des agents pour une période :

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{I} c_i y_i \\ \text{sujet à :} \quad & g_k(\mathbf{y}) \geq l_k, \quad k = 1,\ldots,K, \\ & g(\mathbf{y}) \geq l, \\ & \mathbf{y} \geq 0 \text{ et entiers}, \end{aligned}$$

où y_i représente le nombre d'agents dans le groupe i, c_i est le coût respectif par agent et $\mathbf{y} = (y_1,\ldots,y_I)^{\mathrm{T}}$. Les contraintes sur les SL peuvent être définies sur les types d'appels et sur l'ensemble des appels.

Contrairement à Atlason et al. [12], les auteurs décident de ne pas tester la concavité de g pour sauver du temps d'exécution, mais ils proposent des méthodes heuristiques afin d'éviter le problème de non-concavité. La méthode se base sur l'hypothèse que la fonction g a une courbe en "S" similaire à la figure 3.2 pour un système $M/M/n+M$ avec abandon. La fonction g est souvent non concave lorsque g est proche de 0 et générer une coupe dans

cette région risque d'éliminer la solution optimale.

Afin d'éviter les régions avec faibles niveaux de service, l'algorithme initialise les contraintes pour couvrir une proportion $\alpha \geq 0$ de la charge du trafic d'appels. L'initialisation est évidente dans le cas d'un centre avec un type d'appel et un groupe : la contrainte est $y \geq \alpha\lambda/\mu$. Cette contrainte est plus complexe lorsqu'il y a plusieurs types d'appels et groupes d'agents. Si le temps de service ne dépend que du type d'appel (et non de l'agent), il est possible de modéliser ce sous-problème comme un problème de flot dans un réseau et de générer les contraintes à l'aide de l'algorithme *Max-Flow Min-Cut* pour identifier les charges d'appels non couvertes. Dans ce réseau, le flot représente la charge des appels, les capacités des arcs sont les nombres d'agents, et les chemins du flot dépend des habiletés des agents. La figure 3.3 montre l'exemple d'un graphe où $\rho_k = \lambda_k/\mu_k$. Les avantages d'utiliser un graphe sont qu'il n'y a aucune nouvelle variable à ajouter et qu'il n'est pas nécessaire d'énumérer toutes les contraintes. Les inconvénients sont qu'il faut résoudre le problème de flot pour chaque solution intermédiaire et que les temps de service doivent être indépendants des agents.

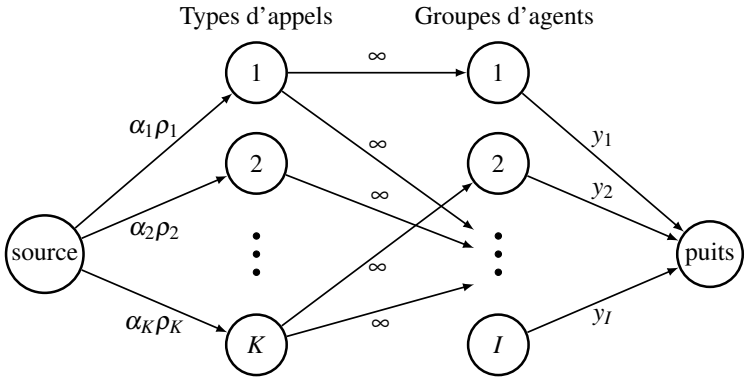


Figure 3.3 – Modèle d'un graphe à résoudre dans le problème du flot maximal, où la capacité supérieure de chaque arc est indiquée ; la capacité inférieure étant zéro.

Une autre façon de satisfaire la couverture des charges d'appels est de diviser les activités des agents d'un même groupe. Soient $w_{k,i}$ le nombre d'agents du groupe i servant les appels du type k et $\mu_{k,i}^{-1}$ le temps de service moyen d'un appel du type k par un agent du

groupe i, les contraintes sont :

$$\sum_{i \in \mathcal{T}_k} w_{k,i}\mu_{k,i} \geq \alpha_k \lambda_k, \quad k = 1, \ldots, K,$$
$$\sum_{k \in \mathcal{S}_i} w_{k,i} = y_i, \quad i = 1, \ldots, I,$$
$$w_{k,i} \geq 0,$$

où $\mathcal{T}_k = \{i : k \in \mathcal{S}_i, \forall i\}$ est l'ensemble des groupes pouvant servir le type d'appel k. Dans ce cas-ci, le paramètre α_k représente plutôt la fraction du volume d'appels du type k à couvrir.

Les avantages de cette approche sont que le nombre de contraintes est limité à exactement $I + K$, et que ces contraintes sont incluses dès l'initialisation. De plus, le taux de service $\mu_{k,i}$ peut dépendre du type d'appel et du groupe d'agents. L'inconvénient est l'ajout de $\sum_{i=1}^{I} |\mathcal{S}_i|$ nouvelles variables continues au problème.

La non-concavité peut être causée par le bruit de la simulation, notamment si les simulations sont courtes. Au lieu d'augmenter la simulation (et le temps d'exécution du même coup), les auteurs proposent d'estimer les sous-gradients par différences finies avec une distance de pas supérieure à 1.

Pour de grands centres d'appels, le temps d'optimisation pour résoudre le problème avec variables $\mathbf{y}$ entières s'avère souvent trop long. Les auteurs proposent de relaxer les variables $\mathbf{y}$ en variables continues et de les arrondir au plafond. Puis corriger la solution finale par une simple recherche locale qui élimine les agents en surplus par ordre décroissant des coûts.

Cette méthode donne de bons résultats dans les expériences numériques. Elle a l'avantage de pouvoir optimiser directement les centres d'appels complexes, car toute la complexité du modèle stochastique est cachée dans la simulation. Les inconvénients sont similaires à ceux d'Atlason et al. [12] : il est possible que g ne soit pas concave et les simulations demandent un temps d'exécution considérable.

Optimisation par une recherche aléatoire aidée par des approximations

Avramidis et al. [18] présentent un algorithme d'optimisation pour le même problème

que Ceẓik et L'Ecuyer [37]. Les auteurs proposent une recherche aléatoire aidée par des approximations analytiques. Leurs approximations se basent sur la décomposition exponentielle de Koole et Talim [93], mais ils ajoutent des files d'attente aux modèles afin de calculer les niveaux de service. Leurs modèles d'approximation supposent une politique de routage par débordement particulière où chaque type d'appel possède une liste ordonnée des groupes d'agents à visiter, et seulement le dernier groupe de chaque liste peut servir les appels en attente. Cette politique de routage n'est pas réaliste, mais les expériences numériques montrent que ces approximations peuvent être utiles pour l'optimisation.

L'algorithme de recherche débute par une procédure d'initialisation qui répartit les agents entre les groupes spécialistes (peu d'habiletés) et les généralistes selon une proportion donnée en paramètre. La recherche aléatoire, la méthode principale, est composée d'une procédure qui retire des agents et d'une procédure qui transfère des agents d'un groupe vers un autre groupe moins coûteux. Étant donné que la taille du voisinage est souvent immense, les solutions candidates sont choisies aléatoirement. Cette recherche pourrait utiliser la simulation pour de très petits centres d'appels (par exemple deux types d'appels), mais elle devient rapidement impraticable pour des centres plus grands.

Finalement, la solution de la recherche aléatoire est corrigée par une recherche locale déterministe qui ajoute ou retire des agents à l'aide de la simulation.

Les expériences numériques montrent que cet algorithme fonctionne légèrement mieux que Ceẓik et L'Ecuyer [37] lorsque les temps d'optimisation sont courts, car l'algorithme de recherche est peu affecté par le bruit d'optimisation (et de simulation). Cependant, la méthode de coupes par sous-gradients semble mieux performer quand le budget de temps alloué est long.

Optimisation par la relaxation lagrangienne

Pot et al. [109] présentent un algorithme simple de recherche par voisinage pour un

centre d'appels avec seulement la contrainte du SL agrégé sur tous les appels :

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{I} c_i y_i \\ \text{sujet à :} \quad & g(\mathbf{y}) \geq l, \\ & \mathbf{y} \geq 0 \text{ et entiers.} \end{aligned}$$

Ils appliquent la relaxation lagrangienne sur l'unique contrainte, et le problème devient :

$$\min_{\mathbf{y} \in \mathbb{N}_+^I} \left[\mathbf{c}^{\mathrm{T}} \mathbf{y} - \beta (g(\mathbf{y}) - l) \right],$$

où $\beta \geq 0$ est le multiplicateur de Lagrange. Leur méthode optimise le dual lagrangien :

$$\max_{\beta \geq 0} \min_{\mathbf{y} \in \mathbb{N}_+^I} \left[\mathbf{c}^{\mathrm{T}} \mathbf{y} - \beta (g(\mathbf{y}) - l) \right].$$

Afin de réduire la taille des voisinages, ils divisent les solutions selon le nombre total d'agents n et le problème devient :

$$\min_{n \in \mathbb{N}_+} \max_{\beta \geq 0} \min_{\mathbf{y}} \left[\mathbf{c}^{\mathrm{T}} \mathbf{y} - \beta (g(\mathbf{y}) - l) : \mathbf{y} \in \mathbb{N}_+^I, \sum_{i=1}^{I} y_i = n \right].$$

Voici les étapes d'optimisation :

1. Le nombre n est optimisé par la méthode du nombre d'or.
2. Pour n fixé, le multiplicateur β est optimisé par une méthode de bissection.
3. Pour n et β fixés, $\mathbf{y}$ est optimisé par une recherche locale qui transfère les agents d'un groupe vers un autre.

Afin de réduire le temps d'exécution, les auteurs estiment la probabilité de blocage $b(\mathbf{y})$ à l'aide d'une méthode de décomposition hyper-exponentielle [54], et ils approximent le SL avec $g(\mathbf{y}) = 1 - b(\mathbf{y})$. Ils évaluent la valeur de la solution intermédiaire $\mathbf{y}$ avec la simulation après l'étape 3. Des expériences numériques montrent que cette méthode a des performances comparables à la méthode de coupes de Cezik et L'Ecuyer [37] pour des petits centres d'appels avec une contrainte sur le SL agrégé seulement. L'avantage est la ra-

pidité grâce aux approximations et à la simplicité de l'algorithme. Le désavantage principal est que l'algorithme ne peut considérer qu'une seule contrainte du SL.

3.3.2.1 Optimisation des quarts de travail avec transfert d'agents

Bhulai et al. [23] présentent une méthode pour optimiser la planification des quarts de travail par la méthode SIPP. La méthode SIPP commence par trouver le nombre d'agents requis $\bar{y}_i$ pour chaque groupe i et chaque période p. Chaque période est optimisée comme étant un système stationnaire indépendant à l'aide des méthodes telles que celles de Pot et al. [109] ou Ceẓik et L'Ecuyer [37] par exemple. La deuxième étape de SIPP consiste à affecter les agents aux quarts de travail de manière à satisfaire le nombre minimal d'agents requis par période. Pour un centre avec un seul groupe d'agents, SIPP trouve une solution souvent trop chère quand les quarts de travail sont peu flexibles. Une difficulté additionnelle s'ajoute lorsque le centre a plusieurs groupes d'agents qui partagent des habiletés en commun, car il peut exister plusieurs solutions. Étant donné que les périodes sont optimisées indépendamment, le nombre d'agents par groupe peut varier beaucoup entre les périodes. Avec ces deux difficultés, SIPP risque de trouver une solution beaucoup trop chère.

Les auteurs proposent d'ajouter au modèle la possibilité de transférer des agents du groupe i avec les habiletés $\mathcal{S}_i$ vers un groupe j avec les habiletés $\mathcal{S}_j$ si $\mathcal{S}_j \subset \mathcal{S}_i$. Par exemple, supposons qu'il y a 3 types d'appels et 7 groupes d'agents avec les habiletés suivantes : {1}, {2}, {3}, {1,2}, {1,3}, {2,3} et {1,2,3} respectivement. Le gestionnaire peut transférer des agents du groupe 7 vers tous les autres groupes. Un agent du groupe 4 peut être transféré vers le groupe 1 ou 2. Soient $\bar{y}_{i,p}$ le nombre d'agents du groupe i requis à la période p, $a_{p,s} = 1$ si le quart de travail s couvre la période p, et $a_{p,s} = 0$ sinon, et $x_{i,s}$ le nombre d'agents du groupe i affectés au quart de travail s. La solution $\mathbf{x}$ de l'étape 2 de SIPP doit maximiser le flot au puits ($\sum_{i=1}^{I} \bar{y}_{i,p}$) pour le graphe présenté à la figure 3.4, et il y a un graphe pour chaque période. Les sommets représentent les groupes d'agents, identifiés par leurs habiletés. Les capacités des arcs sont indiquées dans la figure, et elles sont infinies pour les arcs de transfert entre les groupes.

Soit la variable de transfert $z_{i,j,p}$ qui représente le nombre d'agents du groupe i transférés au groupe j durant la période p. Pour un centre avec K types d'appels et P périodes,

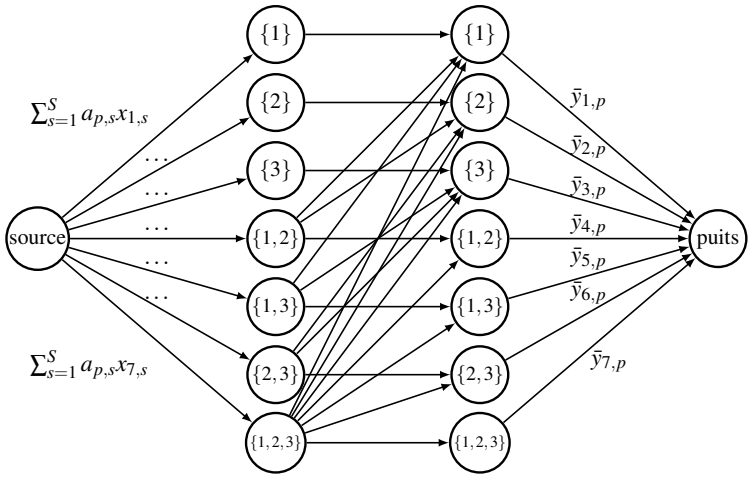


Figure 3.4 – Exemple d'un graphe avec transfert d'agents pour la période p.

il y a au maximum $2^K - 1$ groupes d'agents, et il y a au maximum $P\sum_{n=1}^{K}\binom{K}{n}\sum_{j=1}^{n}\binom{n}{j} = P\sum_{n=1}^{K}\binom{K}{n}(2^n - 1) = P(3^K - 2^K)$ variables de transfert. Heureusement, le nombre de variables z peut être réduit à un maximum de $P\sum_{n=2}^{K}\binom{K}{n}n$ si les transferts peuvent être exécutés en chaîne, voir la figure 3.5. La variable $z_{i,j,p}$ existe si et seulement si $\mathcal{S}_j$ est le plus grand sous-ensemble strict de $\mathcal{S}_i$, c'est-à-dire $\mathcal{S}_j \subset \mathcal{S}_i$ et il n'existe aucun k tel que $\mathcal{S}_j \subset \mathcal{S}_k \subset \mathcal{S}_i$. Le nombre de variables de transfert est réduit de $65P$ à $9P$ dans notre exemple.

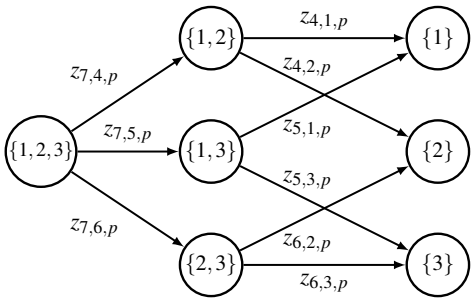


Figure 3.5 – Exemple d'un modèle de transferts en chaîne des agents pour réduire le nombre de variables de transfert.

Les variables de transfert sont ajoutées au problème à l'étape 2 de SIPP :

$$\begin{aligned} \min \quad & \sum_{i=1}^{I}\sum_{s=1}^{S} c_{i,s}x_{i,s} \\ \text{sujet à :} \quad & \sum_{s=1}^{S} \bar{a}_{p,s}x_{i,s} + \sum_{j\in\mathcal{Z}_i^+} z_{j,i,p} - \sum_{j\in\mathcal{Z}_i^-} z_{i,j,p} \geq \bar{y}_{i,p}, \quad i=1,\ldots,I, p=1,\ldots,P, \\ & \mathbf{x},\mathbf{z} \geq 0 \text{ et entier,} \end{aligned}$$

où $c_{i,s}$ est le coût d'affecter un agent du groupe i au quart de travail s, $\mathcal{Z}_i^+ = \{j : \exists\, z_{j,i,p}\}$ est l'ensemble des groupes qui peuvent transférer vers le groupe i, et $\mathcal{Z}_i^- = \{j : \exists\, z_{i,j,p}\}$ est l'ensemble des groupes vers lesquels les agents du groupe i peuvent transférer.

Les expériences numériques montrent que le modèle de transfert d'agents améliore les solutions de la méthode SIPP. Un danger potentiel est le fait que les périodes sont optimisées indépendamment. Nous donnons un exemple d'une solution sous-optimale, tiré de notre article [17], au chapitre 5. Un autre inconvénient est qu'il faut souvent relaxer les problèmes à nombres entiers, et résoudre avec des variables continues, à l'exception de petits centres d'appels. L'arrondissement des variables $z_{i,j,p}$ représente une difficulté supplémentaire, car la décision peut se répercuter sur plusieurs groupes. Nous présentons également un exemple au chapitre 5. De plus, les variables de transfert augmentent la complexité des solutions. Par exemple, les voisinages deviennent beaucoup plus grands dans les algorithmes de recherche.

Optimisation des groupes d'agents et du routage

Wallace et Whitt [132] considèrent le problème où les groupes d'agents et le routage ne sont pas fixés par le modèle comme dans les articles précédents, car ils seront choisis par l'algorithme d'optimisation. Si tous les agents ont le même coût et que le service ne dépend que de l'appel (et non de l'agent), alors il serait en général optimal d'affecter uniquement des généralistes pouvant servir tous les appels. En supposant des durées de service indépendantes des types d'appels et des groupes, les auteurs présentent des observations numériques qui montrent qu'un centre d'appels, dont les agents ont seulement deux habiletés, peut avoir des performances presque aussi bonnes que si les agents avaient toutes les

habiletés.

Leur modèle utilise une politique de routage par priorités en fonction de l'habileté primaire et de l'habileté secondaire de chaque groupe d'agents. Un agent servira en priorité les appels correspondant à son habileté primaire. Ils supposent également que tous les agents ont le même coût. Le problème consiste à minimiser le nombre d'agents tout en respectant des seuils minimaux de SL.

L'algorithme utilise la simulation et il se résume comme suit. Dans l'initialisation, le nombre total d'agents n est déterminé à l'aide de la formule d'Erlang avec tous les appels agrégés et les agents 100% généralistes. Les n agents sont répartis proportionnellement parmi les $K(K-1)$ groupes (toutes les permutations de taille 2 parmi K types) selon le nombre requis par chaque type d'appel. Des agents sont ajoutés à la solution si elle n'est pas réalisable selon la simulation. L'étape d'optimisation contient deux procédures : (1) retirer un agent lorsque la solution est réalisable, et (2) transférer un agent pour rendre une solution réalisable.

L'optimisation repose essentiellement sur l'idée qu'un centre d'appels avec seulement des agents ayant deux habiletés (et souvent moins coûteux) peut performer presque aussi bien que si ces agents étaient 100% généralistes. La performance de cet algorithme pour les problèmes numériques dans Avramidis et al. [18] montre qu'il y a une réduction substantielle du coût si le routage et les groupes d'agents n'étaient pas fixés d'avance. Cependant, cette méthode donne des solutions trop coûteuses pour les problèmes originaux (avec les groupes d'agents pré-déterminés et des coûts différents pour les agents). Cet algorithme serait efficace pour les centres d'appels en phase de création, mais moins utile pour les centres déjà en opération.

D'autre part, limiter le nombre d'habiletés à 2 ou 3 par agents ne serait probablement pas une bonne idée dans un exemple (extrême) où tous les types d'appels ont de très faibles volumes, tels que chaque type nécessite moins qu'un agent. De plus, il n'est pas toujours possible de bien répartir les ensembles d'habiletés dans la réalité. Prenons par exemple un centre d'appels qui doit servir ses clients avec des choix de langues différentes. Il n'est pas réaliste de demander à un agent d'apprendre une nouvelle langue, et il serait plus productif de restreindre les agents à servir avec la ou les langues qu'ils maîtrisent.

3.3.3 Planification par optimisation stochastique

Optimisation stochastique par les modèles fluides

Certaines études récentes utilisent les modèles fluides en raison de leur simplicité et leur rapidité pour approximer les centres d'appels. Un modèle fluide [101] représente généralement un système à trafic intense avec des paramètres asymptotiques comme les taux d'arrivée et le nombre d'agents qui tendent vers l'infini. Les arrivées et les départs des appels forment un mouvement qui ressemble à un fluide continu [75]. La variance stochastique de l'état du système est alors négligeable, et le problème devient quasiment statique et déterministe. Le modèle fluide peut être une approximation utile lorsque l'intensité du volume d'appels est en *surcharge* $\lambda > \mu y$, où y est le nombre d'agents. Par contre, le modèle fluide est moins intéressant si le centre d'appels est en *sous-charge* $\lambda < \mu y$, car les appels seront répondus immédiatement à leur arrivée avec probabilité 1, et la qualité de service sera excellente dans le modèle fluide. Les modèles fluides possèdent quelques inconvénients importants. L'erreur d'approximation d'un modèle fluide peut être substantielle quand il y a des types d'appels de faible volume. Les files d'attente ne sont pas modélisées directement. Un autre aspect particulier avec le modèle fluide est que nous supposons qu'il existe un routage optimal associé au mouvement du fluide. Ce routage optimal n'est pas nécessairement réaliste ou implémentable dans la pratique.

Harrison et Zeevi [75] étudient un problème de planification des agents avec des processus d'arrivée doublement stochastiques : les taux d'arrivée $\Lambda(t)$ sont aléatoires et varient continuellement en fonction du temps t. L'objectif est de minimiser le coût des agents et des coûts de pénalités sur les qualités de service. Malgré que les contraintes dans l'industrie soient généralement sur les SL, les auteurs pénalisent le nombre d'abandons, car cette mesure est plus simple à estimer par le modèle fluide. Leur approximation ne permet pas d'estimer le SL convenablement.

Les auteurs proposent une méthode d'optimisation stochastique à 2 étapes. La première étape consiste à trouver le nombre d'agents requis. La deuxième étape est d'optimiser le routage selon le taux d'arrivée des appels et du nombre d'agents.

Dans leur modélisation, une activité représente la paire (type d'appel k, groupe d'agents

i) et il y a $N = \sum_{i=1}^{I} |\mathcal{S}_i|$ activités au total. Pour chaque activité $n = 1, 2, \ldots, N$, la variable $\hat{k}_n$ correspond au type d'appel de l'activité n et $\hat{i}_n$ est le groupe d'agents correspondant. La matrice d'activité $\mathbf{M}$ de taille $I \times N$ a l'élément $m(i,n) = 1$ si $i = \hat{i}_n$, et $m(i,n) = 0$ sinon. La matrice de service $\mathbf{R}$ de taille $K \times N$ a l'élément $r(k,n) = \mu_n$ si $k = \hat{k}_n$, et $r(k,n) = 0$ sinon. La valeur $1/\mu_n$ représente le temps de service moyen d'un appel de type $\hat{k}_n$ par un agent du groupe $\hat{i}_n$. Soient les taux d'arrivée $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_K)^{\mathrm{T}}$, le nombre d'agents par activité $\mathbf{w} = (w_1, \ldots, w_N)^{\mathrm{T}}$, le nombre d'agents par groupe $\mathbf{y} = (y_1, \ldots, y_I)^{\mathrm{T}}$, le coût par agent pour chaque groupe $\mathbf{c} = (c_1, \ldots, c_I)^{\mathrm{T}}$ et $\mathbf{p} = (p_1, \ldots, p_K)^{\mathrm{T}}$ le coût d'abandon par type d'appel. Le problème stochastique sur la période $[0, T]$ est :

$$\min_{\mathbf{y} \geq 0} \left(\mathbf{c}^{\mathrm{T}} \mathbf{y} + \mathbb{E}\left[\int_0^T f(\Lambda(t), \mathbf{y}) dt \right] \right).$$

Soit $\boldsymbol{\lambda}$ une réalisation de $\Lambda(t)$ au temps t, la fonction $f(\boldsymbol{\lambda}, \mathbf{y})$ représente le coût de pénalité minimal du sous-problème avec $\boldsymbol{\lambda}$ et $\mathbf{y}$ fixés :

$$f(\boldsymbol{\lambda}, \mathbf{y}) = \min_{\mathbf{w} \geq 0} \left[\mathbf{p}^{\mathrm{T}} (\boldsymbol{\lambda} - \mathbf{R}\mathbf{w}) : \mathbf{R}\mathbf{w} \leq \boldsymbol{\lambda}, \mathbf{M}\mathbf{w} \leq \mathbf{y} \right].$$

La première contrainte signifie qu'on ne peut pas servir plus que le nombre d'appels arrivés, et la deuxième contrainte indique que le nombre d'agents actifs ne peut pas dépasser le nombre d'agents disponibles. Étant donné que les résultats des modèles fluides sont basés sur des conditions asymptotiques qui tendent vers l'infini, les auteurs modélisent $\mathbf{y}$ et $\mathbf{w}$ comme des variables continues. La politique de routage pour chaque réalisation de $\Lambda(t)$ est donnée par la solution $\mathbf{w}$ associée. Pour chaque activité n, il ne peut pas avoir plus de w_n agents du groupe $\hat{i}_n$ servant des appels de type $\hat{k}_n$, même s'il y a des agents libres. Ce routage est optimal pour le modèle fluide stationnaire (mais pas nécessairement pour un vrai centre d'appels).

Pour résoudre ce problème stochastique à 2 étapes, les auteurs réfèrent aux méthodes présentées dans Birge et Louveaux [24]. Si la loi de probabilité de $\Lambda(t)$ est discrète et les masses de probabilité sont concentrées sur quelques points, le problème peut être résolu exactement. Autrement, la solution peut être approximée en générant un ensemble de

scénarios par simulation Monte Carlo.

Les expériences numériques montrent que cet algorithme est relativement précis pour des centres d'appels avec deux types d'appels et deux groupes d'agents. Cependant, il n'est pas évident d'adapter les coûts de pénalités aux autres mesures de performance. Il serait intéressant d'expérimenter cet algorithme avec des grands centres d'appels (plusieurs types d'appels et groupes d'agents) et les problèmes avec différentes mesures de performances.

Dans la pratique, la modélisation du processus d'arrivée des appels est un problème difficile. Bassamboo et Zeevi [21] proposent une extension en approximant la fonction de répartition des taux d'arrivée par une distribution empirique basée sur des données historiques. Dans leur section numérique, ils expérimentent un exemple générique d'un centre avec 2 types d'appels et 2 groupes d'agents. Ils observent que la fonction de coût autour de la solution optimale s'aplatit quand la variance des taux d'arrivée augmente. Ceci permet d'avoir une "bonne" solution même si elle n'est pas très proche de la solution optimale. Dans le cas inverse où la variance des taux d'arrivée est petite, il est plus important d'être proche de la solution optimale.

Whitt [136] utilise un modèle fluide pour approximer le temps d'attente moyen et le taux d'abandons d'un centre d'appels avec un seul type d'appel et un groupe d'agents. Le problème suppose des taux aléatoires pour les arrivées et le nombre d'absentéisme des agents. Similairement à Harrison et Zeevi [75], les contraintes sur les temps d'attente et les abandons sont intégrées dans la fonction objectif via des pénalités. Le modèle fluide est un système simplifié, déterministe et qui ne contient aucune file d'attente. Il n'est pas évident d'estimer les mesures telles que les temps d'attente. La contribution principale de cet article est la méthode d'approximation basée sur le modèle fluide qui pourrait être utilisée pour optimiser le nombre d'agents. Supposons y agents, un taux d'arrivée λ et un taux de service μ, la probabilité d'abandon est : $\mathbb{P}[\text{Ab}] = \max\{\lambda - \mu y, 0\}/\lambda$. Soit $F(t)$ la fonction de répartition du temps de patience. Dans le modèle fluide, les clients qui abandonnent ont attendu moins longtemps que ceux qui sont servis. Supposons que tous les appels sont immédiatement servis après avoir attendu un seuil de temps w. La probabilité qu'un appel ait abandonné avant un temps t est $F(t)$, où $0 < t < w$. Afin d'établir la relation entre le

temps de patience et la probabilité d'abandon, il faut trouver w tel que :

$$F(w) = \mathbb{P}[\text{Ab}] = \frac{\max\{\lambda - \mu y, 0\}}{\lambda}.$$

La fonction de répartition du temps d'attente W d'un appel est alors :

$$\begin{aligned} \mathbb{P}[W < t] &= F(t), \quad \text{si } 0 \leq t < w, \\ \mathbb{P}[W < t] &= 1, \qquad \text{si } t \geq w. \end{aligned}$$

Il est aussi possible de calculer la longueur moyenne de la file d'attente dans le modèle fluide :

$$Q = \lambda \int_0^w (1 - F(t))dt.$$

Étant donné que tous les appels servis doivent avoir attendu exactement w, tous ces appels doivent être entrés dans la file d'attente.

Pour modéliser l'absentéisme, l'auteur suppose un nombre d'agents $\bar{y} = \gamma y$, où γ est une variable aléatoire dans l'intervalle $(0, 1]$. Afin d'estimer les espérances des mesures de performance, l'auteur utilise l'approche par scénario pour générer différents taux d'arrivée et proportions d'absentéisme. Notons que cet article ne considère qu'un seul groupe d'agents, alors l'effort n'a pas été mis sur la méthode d'optimisation.

Optimisation stochastique par linéarisation des fonctions de SL

Robbins et Harrison [113] présentent une méthode d'optimisation stochastique pour la planification des horaires avec des taux d'arrivée aléatoires. Leur problématique se base sur de vrais centres d'appels de sous-traitance dans l'industrie du support technique. Ils étudient les centres d'appels avec un seul type d'appel et un groupe d'agents. Les gestionnaires doivent satisfaire le SL agrégé par semaine ou par mois. Il n'y a aucune contrainte sur le SL par heure ou par jour. Dans leur modèle, des coûts de pénalités sont imposés si la contrainte sur le SL n'est pas respectée à la fin de la semaine ou du mois.

Similairement à Atlason et al. [12] et Ceẓik et L'Ecuyer [37], leur méthode se base sur la propriété que la fonction du SL d'un système $M/M/n+M$ (avec abandons) a une

forme d'un "S" étiré. La nouveauté proposée par les auteurs est la linéarisation de la partie concave de la fonction du niveau de service $g(y)$. Plusieurs droites sont calculées à l'aide des sous-gradients estimés avec Erlang A, et le minimum de ces droites constitue une approximation de $g(y)$ par une fonction linéaire par partie. La figure 3.6 montre un exemple de linéarisation de la fonction $g(y)$. Les auteurs optimisent un problème stochastique approximatif en générant 50 à 100 scénarios. Il faut linéariser la fonction $g(y)$ pour chaque scénario avant de commencer l'optimisation.

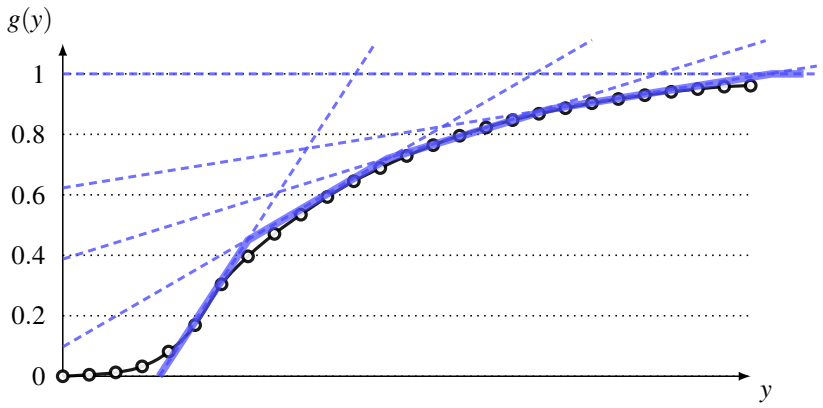


Figure 3.6 – Exemple de linéarisation de la fonction du niveau de service.

Supposons M scénarios, avec probabilité q_m pour le scénario m, P périodes, S quarts de travail, x_s agents affectés au quart de travail s et le coût associé c_s par agent. La variable $a_{p,s} = 1$ si le quart de travail s couvre la période p, et $a_{p,s} = 0$ sinon. Soient l le seuil du SL à atteindre, r le coût de pénalité sur le SL, w_m la mesure du SL raté dans le scénario m, et $v_{p,m}$ le nombre d'appels ayant reçu un bon service dans la période p du scénario m sur $n_{p,m}$ appels reçus. Pour chaque période p du scénario m, la fonction $g(y)$ est linéarisée par H droites de sous-gradients de la forme $d_{h,p,m}y + b_{h,p,m}$. Le modèle permet d'imposer un minimum de u_p agents par période p et un maximum de z_s agents pour le quart de travail

s. Le problème d'optimisation est :

$$
\begin{aligned}
\min \quad & \sum_{s=1}^{S} c_s x_s + r \sum_{m=1}^{M} q_m w_m \\
\text{sujet à :} \quad & v_{p,m} \leq n_{p,m} \left(d_{h,p,m} \sum_{s=1}^{S} a_{p,s} x_s + b_{h,p,m} \right), && \forall h, \forall p, \forall m, \\
& \sum_{p=1}^{P} n_{p,m} w_k \geq \sum_{p=1}^{P} (n_{p,m} l - v_{p,m}), && \forall m, \\
& v_{p,m} \leq n_{p,m}, && \forall p, \forall m, \\
& \sum_{s=1}^{S} a_{p,s} x_s \geq u_p, && \forall p, \\
& x_s \leq z_s, && \forall s, \\
& x_s \in \mathbb{Z}^+, v_{p,m} \geq 0, w_s \geq 0, && \forall p, \forall s, \forall k.
\end{aligned}
$$

La première contrainte représente l'approximation du SL par une fonction linéaire par partie. La deuxième contrainte calcule $w_m \geq 0$ qui est la proportion ratée du SL pour atteindre la cible l.

Les auteurs optimisent des problèmes avec 3696 variables entières et plus de 16 000 variables continues. Ils proposent d'utiliser la méthode de décomposition L-Shaped [24] qui est une variante de la décomposition de Benders. Le problème est séparé en un problème maître et M sous-problèmes (un par scénario). Les coupes sont ajoutées itérativement au problème maître. Notons que la méthode L-Shaped est capable de calculer des bornes d'optimalité. Les auteurs suggèrent de relaxer les variables entières en variables continues durant les premières itérations, puis d'optimiser avec des variables entières lorsque l'algorithme L-Shaped atteint un niveau de convergence suffisant.

Dans les expériences numériques, l'optimisation stochastique trouve des solutions de 12% à 21% moins coûteuses que la solution du problème espéré (un seul scénario avec le taux d'arrivée moyen par période). Cette différence de coûts est la *valeur de la solution stochastique*. Elle mesure l'avantage d'optimiser la version stochastique du problème au lieu du problème espéré. En comparant la méthode stochastique à la méthode SIPP à deux étapes qui utilise Erlang C, la méthode stochastique trouve des solutions de 2% à 27% moins coûteuses que SIPP. La différence de coûts est substantielle lorsqu'il y a peu de

quarts de travail.

Optimisation stochastique par recherche par voisinage variable

Robbins et Harrison [112] présentent un algorithme de *recherche par voisinage variable* [74] pour le problème d'un centre d'appels avec un seul type d'appel avec des taux d'arrivée aléatoires (le même problème que Robbins et Harrison [113]). L'objectif est de concevoir un algorithme pouvant trouver une solution rapidement (en 5 minutes). Les auteurs ne modélisent pas les possibilités de recours d'agents (comme annuler des périodes de formation, faire rentrer des agents sur appel ou offrir des congés sans solde), car ils supposent que ces actions sont généralement très limitées en pratique.

Le problème stochastique est discrétisé en M scénarios. L'initialisation est une méthode de construction partant d'une solution sans agent, puis ajoute un agent à la fois en fonction de la variable de couverture des "mauvais" appels c_s (définie plus loin) pour chaque quart de travail s. Soit w_p le nombre d'appels moyen ayant reçu une mauvaise qualité de service à la période p :

$$w_p = \frac{\sum_{m=1}^{M} n_{p,m}\,(1 - g(\bar{y}_p, n_{p,m}))}{M},$$

où $g(\bar{y}_p, n_{p,m})$ est la fonction du SL calculée avec Erlang A, $n_{p,m}$ est le taux d'arrivée à la période p du scénario m, et $\bar{y}_p = \sum_{s=1}^{S} a_{p,s} x_s$ est le nombre total d'agents qui travaillent à la période p. La variable $c_s = \sum_{p=1}^{P} a_{p,s} w_p / \sum_{p=1}^{P} a_{p,s}$ représente le nombre agrégé moyen de "mauvais" appels couverts par le quart de travail s. La méthode d'initialisation s'arrête lorsque le SL (calculé avec Erlang A) atteint ou dépasse la cible. L'initialisation se fait très rapidement.

La méthode de recherche par voisinage variable utilise la simulation pour estimer les SL. La recherche se fait en ajoutant des agents, en retirant des agents ou en changeant les quarts de travail des agents. Les procédures de recherche ressemblent à la recherche locale de Avramidis et al. [17].

Les expériences numériques montrent que la recherche par voisinage réduit le coût d'environ 5% d'une solution obtenue par la méthode d'initialisation. Les performances de cet algorithme avec un budget de 5 minutes sont comparées à l'approche par programma-

tion linéaire à nombres entiers de Robbins et Harrison [113] avec des budgets de temps de 45 à 120 minutes. Les résultats sont mixtes et montrent que la recherche par voisinage donne des solutions de 6% moins coûteuses à 2.7% plus coûteuses. Sans limite de temps, la recherche par voisinage trouve souvent de meilleures solutions que l'approche par programmation linéaire. Notons que cette dernière n'utilise pas la simulation (ni de recherche locale), et les erreurs d'approximation pourraient être la cause de la sous-optimalité des solutions.

L'avantage de la méthode de recherche par voisinage est sa rapidité et son efficacité à trouver de bonnes solutions. Cependant, il n'est pas évident d'adapter cette méthode pour un centre d'appels multi-compétences, parce qu'il n'existe pas de formule d'approximation précise et la taille du voisinage grandit exponentiellement.

Optimisation stochastique avec actions de recours

Dans la littérature des centres d'appels, pratiquement toutes les études sur les prévisions des appels ne considèrent pas l'affectation des agents, et la majorité des études sur l'optimisation des agents supposent des prévisions parfaites. Mais dans la réalité, un gestionnaire d'un centre d'appels observe le volume d'appels du matin, et il peut mettre à jour les prévisions de l'après-midi, puis appliquer les actions de recours nécessaires. Les processus de prévision des appels et d'optimisation des agents sont fortement interreliés dans la pratique.

Mehrotra et al. [104] et Gans et al. [59] étudient le problème de planification stochastique avec recours intra-journaliers pour un centre d'appels avec un seul type d'appel et un groupe d'agents. Les volumes d'appels observés durant la journée sont utilisés pour corriger les prévisions et les horaires des agents pour les périodes à venir.

Notamment, Gans et al. [59] proposent une méthode qui combine la prévision des appels et l'optimisation stochastique avec recours. Ils proposent un modèle de prévision paramétrique bayésien qui tient compte des dépendances entre les jours de la semaine et des périodes de la journée. Avec l'optimisation stochastique traditionnelle, les scénarios sont générés habituellement par la méthode d'échantillonnage Monte Carlo, et il peut nécessiter beaucoup de scénarios avant d'obtenir une distribution représentative. Comme les

auteurs supposent des paramètres de prévision gaussiens, ils choisissent d'utiliser la méthode de quadrature de Gauss pour générer plus efficacement les scénarios du problème stochastique. Cette méthode permet de réduire le nombre de scénarios et d'accélérer l'optimisation.

Ils considèrent la contrainte sur la proportion d'abandons, agrégée sur la journée. Ils supposent des processus d'arrivée Poisson stationnaires par période, et ils estiment la proportion d'abandons selon une file $M/M/n+M$. Soit $f(\lambda,\mu,\nu,n)$ la fonction retournant la proportion d'abandons, qui a pour paramètres les taux d'arrivée, de service et d'impatience, et n agents, respectivement. Il a été démontré que $f(\lambda,\mu,\nu,n)$ est convexe et non croissante en fonction de n lorsque $\mu > \nu$, avec une courbe inversée semblable à celle du SL représentée à la figure 3.6. Les auteurs remplacent cette fonction par le maximum d'un ensemble d'équations linéaires, ce qui est équivalent à la méthode de Robbins et Harrison [113] pour la contrainte sur le SL. Leur problème d'optimisation est linéaire et n'a pas besoin de simulation.

Ils comparent différentes versions de problèmes stochastiques. Il y a premièrement le problème conventionnel avec les taux d'arrivée moyens (déterministes) sans recours. Il y a les problèmes stochastiques à 1-étape avec ou sans recours d'agents. Le problème stochastique à 1-étape avec recours est en fait 2 problèmes d'optimisation. Tout d'abord, les horaires sont optimisés sans les options de recours, en supposant des taux d'arrivée stochastiques. Puis, les recours sont optimisés plus tard, une fois que les volumes d'appels ont été observés. Les horaires trouvés à l'étape 1 et les taux d'arrivée sont des paramètres fixes dans le problème de recours. Finalement, il y a la version du programme stochastique à 2-étapes avec recours. Dans cette version, les horaires des agents et les actions de recours sont optimisés simultanément, ce qui donne aussi un problème linéaire beaucoup plus grand.

Comme attendu, les résultats numériques montrent que le coût optimal est plus bas quand on optimise la version stochastique (en générant plusieurs scénarios) et on utilise des actions de recours. Ainsi, la programmation stochastique à 2-étapes avec recours donne les meilleurs résultats.

Optimisation robuste

Liao et al. [100] optimisent les quarts de travail d'un centre d'appels avec un seul type d'appel et un groupe d'agents. Tel que dans Avramidis et al. [19], le taux d'arrivée d'une journée quelconque est une variable aléatoire, et le taux est donné par l'expression $\Lambda = \theta\lambda$, où θ est le seul élément aléatoire dans l'équation et généralement de moyenne 1 (cependant cette hypothèse n'est pas nécessaire ici). De plus, les agents doivent répondre à un nombre aléatoire de courriels. Le problème suppose que les courriels à répondre sont ceux qui ont été reçus durant la journée précédente, donc tous les courriels sont disponibles dès l'ouverture du centre. Par contre, un agent ne peut être assigné qu'à un seul type de tâche durant une période : soit répondre aux appels ou aux courriels. Les appels doivent évidemment être servis rapidement, mais les courriels peuvent être répondus à n'importe quelle période de la journée. Cependant, les agents devront faire du temps supplémentaire, ce qui implique un coût additionnel, s'il reste des courriels après la fermeture.

Puisqu'il n'y qu'un seul type d'appel, les auteurs utilisent l'approche SIPP où le nombre d'agents requis pour chaque scénario est précalculé à l'aide de la formule d'Erlang C (le problème suppose aucun abandon). Ils étudient trois méthodes de programmation stochastique : (1) l'approche classique de la programmation stochastique qui optimise sur la moyenne de tous les scénarios du problème, (2) l'optimisation sur la *valeur à risque conditionnelle* (ou "*conditional value at risk*") qui optimise sur la moyenne des $z\%$ pires scénarios, et (3) l'approche de l'optimisation robuste qui optimise le i-ième pire scénario. La programmation stochastique classique aura le meilleur coût parmi les 3 approches, car les deux autres optimisent sur les pires cas. Notons que si $z = 100\%$, alors l'optimisation sur la valeur à risque conditionnelle sera la même que l'approche classique.

Les expériences numériques montrent l'attrait de l'optimisation robuste pour ce type de problème. L'optimisation robuste est plus simple que les 2 autres approches, car elle ne considère qu'un seul scénario, donc moins de variables et de contraintes. Le fait qu'il n'y a qu'un seul type d'appel facilite énormément l'approche robuste, parce que le risque augmente de façon monotone avec le taux d'arrivée des appels. Il ne suffit que de réduire ou d'augmenter le taux d'arrivée pour contrôler le risque (ou la robustesse). Par contre, il n'est

pas évident d'adapter cette approche robuste pour un problème multi-dimensionnel d'un centre avec plusieurs types d'appels et groupes d'agents. Déterminer le nombre d'agents nécessaires par groupe devient déjà un problème difficile en soi, et identifier les pires scénarios n'est pas trivial à moins d'avoir une forte corrélation entre les types d'appels.

3.4 Optimisation des politiques de routage

Le routage des appels joue un rôle beaucoup plus important dans un centre d'appels multi-compétences, car il y a beaucoup plus de choix pour assigner les agents aux appels. La recherche sur les politiques de routage pour les centres d'appels modernes est encore à ses débuts, mais ce sujet suscite un intérêt accru depuis les dernières années. Nous pouvons trouver néanmoins un certain nombre d'études dans la littérature. Par contre, les résultats de ces études se basent quasiment tous sur des conditions asymptotiques avec un volume d'appels et un nombre d'agents qui tendent vers l'infini.

Tout d'abord, il est théoriquement possible de résoudre optimalement le routage à l'aide de la *programmation dynamique* (PDY). Les problèmes sont souvent simplifiés en modélisant le centre d'appels par une CMTC. Typique au problème de PDY, la malédiction de la dimensionnalité (le nombre d'états augmente exponentiellement avec la taille du problème) limite l'utilité de l'approche PDY à de petits centres d'appels, souvent de 2 à 3 types d'appels. Une autre difficulté majeure est le calcul des mesures de performance, surtout lorsqu'elles sont en fonction du temps d'attente. De plus, une simple modification au modèle peut changer complètement la formulation du problème de PDY.

Koole et Pot [91] utilisent la PDY approximative pour trouver les politiques optimales par une méthode d'*itération sur les politiques*. Un coût est associé au temps d'attente de chaque appel, et l'objectif est de minimiser le coût moyen à long terme. Leur modèle suppose un centre d'appels particulier où il y a un groupe d'agents spécialistes par type d'appel et un groupe d'agents généralistes pouvant servir tous les appels. Étant donné le nombre d'états très élevé, la fonction de valeur est approximée en calculant uniquement sur un ensemble d'échantillons d'états générés par la simulation. Les exemples numériques (avec 2 ou 3 type d'appels) montrent que cette méthode fonctionne bien pour les petits

centres d'appels.

Koole et al. [90] modélisent une CMTC avec des taux de transition pour compter le temps d'attente du client à la tête de la file d'attente. Un état de la chaîne représente le nombre d'agents libres ou le compteur du temps d'attente du premier client dans une file (quand tous les agents sont occupés). Plus le taux de transition du compteur de temps est élevé, plus la mesure du temps d'attente sera précise, mais le nombre d'états augmentera également et ralentira l'approximation. En incorporant le temps d'attente dans l'état, cette CMTC permet de modéliser des politiques de routage basées sur les temps d'attente, comme la politique de délais PD par exemple. Leur modèle ne considère pas les abandons. Nous présentons leur modèle avec plus de détails, et nous proposons une variante à leur modèle au chapitre 8. Koole et al. [89] se basent sur ce modèle de CMTC et optimisent le routage, en fonction du temps de délai et du nombre d'agents libres, pour un centre d'appels de modèle N (voir la figure 2.1 à la page 16) par la PDY. Leur objectif est de minimiser les temps d'attente moyens ou de maximiser les SL. Encore une fois, cette approche ne serait efficace que pour de petits centres d'appels.

Gans et Zhou [60] optimisent un centre d'appels particulier sans abandon, du modèle V, avec deux types d'appels et un groupe d'agents généralistes. Il y a une infinité d'appels de type 2 en attente. Seuls les appels de type 1 sont sensibles aux temps d'attente, et ils requièrent une meilleure qualité de service, soit le SL ou le temps d'attente moyen. L'objectif est d'optimiser le routage afin de maximiser le nombre d'appels servis de type 2 tout en satisfaisant les contraintes de qualité de service pour les appels du type 1. Les auteurs démontrent qu'il est optimal d'utiliser une politique de routage en fonction d'un seuil aléatoire du nombre d'agents occupés lorsque les taux de service sont identiques pour les deux types d'appels. Par contre, cette politique n'est pas nécessairement optimale lorsque les taux de service sont différents.

Chevalier et al. [41] étudient le routage de grands centres d'appels sans file d'attente (un "*loss system*"). L'objectif est d'optimiser le routage afin de minimiser la probabilité de blocage. Leur modèle de centres d'appels contient des spécialistes avec une seule habileté et des généralistes avec toutes les habiletés. Ils démontrent que si les généralistes ne servent pas plus vite que les spécialistes, alors il est optimal d'envoyer les appels aux

agents spécialistes en premier. Ils proposent la règle d'affectation heuristique 80/20 qui signifie d'avoir 80% d'agents spécialistes et 20% de généralistes.

Plusieurs études proposent des politiques de routage qui sont asymptotiquement optimales dans un régime à trafic intense, où le volume d'appels tend vers l'infini. Ces solutions imposent généralement des simplifications importantes aux modèles afin d'obtenir des preuves d'optimalité asymptotique. Par exemple, la loi de probabilité du temps de service d'un agent est identique pour tous les types d'appels, ou bien le centre d'appels n'a qu'un seul type d'appel, mais les agents sont hétérogènes (différentes vitesses de service). Dans la littérature, nous distinguons deux grands types de régimes asymptotiques. Le régime asymptotique traditionnel suppose que les taux d'arrivée et de service tendent vers l'infini, mais le nombre d'agents reste fixe. L'autre régime, plus récent et appelé le régime Halfin-Whitt [72], suppose que les taux d'arrivée et le nombre d'agents tendent vers l'infini, mais les taux de service restent fixes.

Supposons que chaque appel de type k engendre un coût $C_k(w)$ défini par une fonction convexe croissante avec pour dérivée C_k' et $C_k(0) = 0$, et qui dépend du temps d'attente w du client. L'objectif est de minimiser la somme sur tous les appels :

$$\sum_{k=1}^{K} \sum_{n=1}^{N_k} C_k(w_{k,n}),$$

où $w_{k,n}$ est le temps d'attente du n-ième appel de type k, et N_k est le nombre d'arrivées de type k. van Mieghem [130] propose la *règle-cμ généralisée* (G$c\mu$) qui est optimale dans le régime asymptotique traditionnel pour un centre d'appels avec un groupe d'agents, plusieurs types d'appels et sans abandon. La politique consiste à servir l'appel qui a le plus grand taux de pénalité instantané :

$$k^* = \arg\max_k C_k'(w_k)\mu_{k,i}, \tag{3.7}$$

où w_k est le temps d'attente de l'appel à la tête de la file k, et $\mu_{k,i} = \mu_k$ puisqu'il n'y qu'un seul groupe d'agents. Si les fonctions de pénalités sont linéaires (disons de la forme $C_k(w) = a_k + c_k w$), alors l'ordre de priorité est déterminé simplement par l'ordre décrois-

sant des produits des coefficients c_k multipliés par les taux de service μ_k, ce qui donne la *règle-cμ* proposée originalement par Cox et Smith [46]. Cette politique est appelée une *politique aveugle*, car elle ne dépend que de l'état présent et observable du système. À part les taux de service, le routage ne dépend aucunement des paramètres du système tels que les taux d'arrivées, le nombre d'agents, etc. Cette propriété augmente la robustesse de la règle G$c\mu$. Cependant, cette politique suppose la condition asymptotique où tous les appels auront un temps d'attente non nul. Cette règle est conçue pour la sélection des appels en attente quand un agent devient libre, et non pour la sélection d'un agent lorsqu'un nouvel appel arrive.

van Mieghem [131] considère l'objectif de maximiser les SL avec la règle G$c\mu$. La mesure du SL est transformée en une fonction géométrique convexe et croissante. Mandelbaum et Stolyar [102] adaptent la règle G$c\mu$ aux systèmes de files d'attente avec plusieurs types de clients et de groupes d'agents, et des fonctions de pénalités convexes strictement croissantes. Atar et al. [11] considèrent un centre d'appels avec K types d'appels, un seul groupe d'agents, des durées de patience exponentielles de moyenne ν_k pour les appels de type k et une fonction de coût linéaire $C_k(w) = c_k w$. La politique optimale asymptotique est la règle-$c\mu/\nu$ qui est une règle de priorité par ordre décroissant des $c_k\mu_k/\nu_k$. Si le temps de patience est indépendant du type d'appel, c'est-à-dire $\nu_k = \nu$, alors cette politique est identique à la règle-$c\mu$. Tezcan et Dai [127] prouvent que la règle-$c\mu$ est asymptotiquement optimale pour le modèle N dans le régime Halfin-Whitt lorsque le taux de service dépend uniquement du groupe d'agents et l'objectif est de minimiser la somme des coûts d'attente et d'abandons.

Atar [10] étudie également les systèmes de files d'attente avec plusieurs types d'appels et groupes d'agents. Pour le cas du service non préemptif (tel que supposé dans cette thèse), la politique de contrôle optimale impose une limite maximale, qui dépend de l'état du système, sur le nombre d'agents pouvant travailler sur chaque activité. Une activité est définie par la paire composée d'un type d'appel et d'un groupe d'agents.

Milner et Olsen [105] étudient les dangers d'optimiser le routage en fonction du SL. Ils optimisent le routage d'un modèle V dans un régime à trafic intense, sans abandon. Seul le type d'appel 1 est contraint à satisfaire un minimum de SL. L'objectif est alors

d'optimiser le routage afin de minimiser le temps d'attente moyen des appels du type 2 tout en respectant le seuil de SL du type 1. Ils observent qu'un tel centre d'appels peut donner priorité au type 1 lorsque le volume d'appels est faible (car il est plus facile de satisfaire le SL), alors que les temps d'attente sont déjà faibles. Mais le type 1 perd parfois la priorité quand le volume du type d'appels 1 est élevé, dans le but de réduire les temps d'attente du type 2. Lorsque le nombre d'appels est grand, les temps d'attente des appels du type 1 risquent d'être au-dessus du seuil acceptable, et il y a moins d'incitatif à favoriser ces appels. Les auteurs expérimentent différentes façons de modéliser la fonction objectif : soit par des fonctions de pénalités ou par des contraintes, ou un mélange des deux. Les auteurs proposent aussi d'utiliser plusieurs contraintes de SL, par exemple, 80% en 20 secondes et 95% en 1 minute.

Gurvich et Whitt [71] supposent que les durées de service dépendent uniquement du groupe d'agents. Ils proposent une politique de routage en fonction des longueurs des files d'attente. Un agent libre choisit de servir le type d'appel ayant le quotient, déterminé par la longueur de la file d'attente sur la somme des longueurs de toutes les files d'attente, qui dépasse le plus un seuil de ratio fixé. Le but est de garder les longueurs des files d'attente à des proportions jugées optimales. Cette politique s'appelle "*fixed-queue-ratio*" (FQR). Gurvich et Whitt [70] proposent une politique plus générale où les seuils de ratios dépendent des longueurs des files d'attente et des taux d'occupation des agents. Lorsqu'un agent devient libre, il va servir le type d'appel dont la longueur de la file excède le plus selon un seuil dynamique. La règle de routage est similaire pour choisir un agent libre quand un appel arrive. Au lieu de la longueur de la file d'attente, le quotient est calculé par le nombre d'agents libres d'un groupe divisé par le nombre total d'agents libres. Le but est de maintenir la répartition des agents libres à travers les groupes selon des proportions considérées optimales. Un nouvel appel est assigné à un agent libre dont le ratio d'inoccupation du groupe excède le plus un certain seuil dynamique. Comme la règle G$c\mu$, cette politique de routage ne dépend que de l'état actuel du centre d'appels. En combinant avec leur méthode d'optimisation des agents basée sur l'approximation fluide, ils prouvent l'optimalité asymptotique de leurs méthodes d'optimisation du routage et d'affectation des agents avec contraintes sur les SL. Perry et Whitt [108] étudient l'utilité des politiques non conserva-

trices de travail dans un centre d'appels de modèle X (voir la figure 2.1 à la page 16), avec des surcharges imprévisibles des appels. Ils proposent une version du FQR avec des seuils (nommée FQR-T) sur les longueurs des files d'attente. Ils observent que la variante FQR-T performe mieux que FQR dans les situations de surcharges d'appels.

Armony [6] étudie le routage pour un centre ayant un seul type d'appel et des agents hétérogènes avec des temps de service différents. Ceci est plus réaliste, car les agents ne servent pas les appels avec la même rapidité. L'efficacité d'un agent dépend généralement de son expérience et du nombre de tâches (un agent spécialiste est souvent plus rapide qu'un généraliste). L'étude démontre qu'il est optimal d'envoyer les appels aux agents les plus rapides en premier, ce qui est la politique "*fastest-server-first*" (FSF). Cependant, cette politique n'est pas équitable aux yeux des agents rapides, car ils seront plus occupés que leurs collègues. La règle standard de l'industrie est de choisir *le premier agent inactif avant* ou "*longest-idle-server-first*". Armony et Ward [7] étudient le problème d'équité pour le même centre d'appels. L'objectif est d'avoir un routage qui minimise le temps d'attente moyen tout en étant équitable pour les agents. La politique de routage proposée est d'envoyer les appels aux agents libres les plus rapides lorsque le nombre d'appels dans le centre est plus grand qu'un seuil, et d'envoyer les appels aux agents les plus lents lorsque le nombre d'appels est sous ce seuil. Armony et Ward [8] considèrent le même problème pour le cas plus général d'un centre avec plusieurs types d'appels. Elles présentent une politique de routage aveugle qui ne dépend que de l'état actuel du système. Un nouvel appel est assigné à un agent libre en fonction de la plus longue période d'inactivité pondérée. Lorsqu'un agent devient libre, la règle G$c\mu$ est utilisée pour choisir le prochain appel à servir.

D'autres études optimisent simultanément plusieurs niveaux de décisions à la fois. Wallace et Whitt [132] supposent un centre d'appels avec la même distribution du temps de service pour tous les types d'appels, ainsi qu'un coût identique pour tous les agents. Leur méthode construit les ensembles d'habiletés des agents, et optimise l'allocation des agents et du routage à l'aide de la formule du "*square-root safety staffing approximation*". Cette formule est dérivée du modèle asymptotique d'un système à trafic intense. La solution est ensuite améliorée et raffinée par une recherche locale à l'aide de la simulation. Nous avons

déjà résumé cet article dans la revue des algorithmes de planification à la section 3.3.2. Sisselman et Whitt [120] adaptent l'étude précédente afin d'associer un gain ou une valeur lors de l'affectation d'un appel à un agent. Ce gain peut correspondre au profit généré lorsqu'un type d'appel est servi par un agent particulier, ou il peut représenter la préférence d'un agent à servir certains types d'appels.

Koole et al. [92] présentent un algorithme d'optimisation des habiletés des agents avec un nombre total d'agents constant. Ce problème peut être vu comme une optimisation du coût du personnel et du routage. Le coût d'un agent varie selon le nombre d'habiletés qu'il possède. L'objectif est de minimiser la somme des habiletés : $\sum_{i=1}^{I} |\mathcal{S}_i| y_i$ où $\mathcal{S}_i$ est l'ensemble d'habiletés du groupe i, et y_i est le nombre d'agents dans le groupe i. En partant d'une solution où tous les agents ont toutes les habiletés, l'algorithme réduit itérativement le nombre d'habiletés d'un agent. À chaque itération, l'algorithme choisit la solution qui minimise le quotient : la réduction du SL divisée par la réduction du nombre d'habiletés.

Harrison et Zeevi [75] présentent une politique de routage pour le problème avec des taux d'arrivées doublement stochastiques qui varient en fonction du temps. Le problème consiste à minimiser le coût des agents et les pénalités causées par les abandons. Le routage et la planification des agents sont obtenus en optimisant un programme linéaire basé sur les équations d'un modèle fluide. La solution donne le nombre d'agents de chaque groupe pouvant servir chaque type d'appel. Autrement dit, la solution divise chaque groupe en des agents spécialistes avec une seule habileté. Nous avons déjà résumé leur méthode dans la section 3.3.3. Bassamboo et al. [20] démontrent que cette approche est asymptotiquement optimale pour les centres d'appels à trafic intense.

Chapitre 4

Optimisation à l'aide d'un simulateur d'une chaîne de Markov en temps discret

Dans ce chapitre, nous proposons d'accélérer l'optimisation de la planification en remplaçant la simulation par *événements discrets* (ED) par la simulation d'une *chaîne de Markov en temps discret* (CMTD). Nous présentons les adaptations afin d'utiliser efficacement le simulateur CMTD avec un algorithme d'affectation des agents pour une période. Ce chapitre présente les résultats de notre article [30] en mettant l'emphase sur la partie d'optimisation.

Une difficulté majeure dans l'optimisation d'un centre d'appels multi-compétences est le manque de formule analytique ou d'approximation précise et rapide. Si tous les processus (inter-arrivées, services et abandons) sont exponentiels, le centre d'appels peut être approximé par une *chaîne de Markov en temps continu* (CMTC). Ceci a été fait dans la littérature pour des centres d'appels avec 2 ou 3 types d'appels, voir la section 3.2 à la page 30. Pour des problèmes plus grands ou complexes, il est nécessaire de recourir à la simulation, ou une combinaison : optimiser les premières itérations avec des algorithmes d'approximations grossières, puis raffiner la solution à l'aide d'un simulateur précis, comme dans Avramidis et al. [18]. La précision des approximations est particulièrement impor-

tante pour les méthodes d'optimisation basées sur les sous-gradients [13, 17, 37], car elles sont sensibles aux erreurs et aux bruits des fonctions. La simulation consomme souvent la majorité du temps d'exécution de ces algorithmes.

Nous proposons d'utiliser un simulateur CMTD qui est beaucoup plus précis que les formules d'approximations analytiques existantes et plus rapide que la simulation ED. Si tous les processus sont exponentiels (ce que la majorité des problèmes pratiques suppose), alors le simulateur CMTD convergera vers les valeurs exactes. Autrement, le simulateur CMTD pourrait être utilisé lors des premières itérations des algorithmes d'optimisation, puis changer à un simulateur ED après quelques itérations. Dans le cadre de cette thèse, nous nous intéressons principalement à l'utilisation d'un simulateur CMTD afin d'améliorer les méthodes d'affectation des agents, plus spécifiquement l'algorithme de Ceẓik et L'Ecuyer [37]. Nous verrons que remplacer bêtement le simulateur ED par un simulateur CMTD ne donne pas nécessairement les améliorations attendues. Nous décrivons les modifications et les adaptations au simulateur CMTD afin d'augmenter l'efficacité de l'optimisation. Nous discutons aussi des cas où il serait avantageux d'utiliser le simulateur CMTD, accompagnés par quelques exemples.

4.1 Simulation à l'aide d'une chaîne de Markov en temps discret (CMTD)

Nous présentons dans cette section les concepts de base d'un simulateur CMTD, mais une étude plus approfondie se trouve dans la thèse de Buist [28].

4.1.1 Chaîne de Markov en temps continu (CMTC)

Nous commençons par la définition d'une chaîne de Markov en temps continu ; la description est disponible dans plusieurs ouvrages dont Ross [115]. Soit un processus stochastique en temps continu $\{X(t), t \geq 0\}$, où $X(t) \in \mathcal{S}$ et $\mathcal{S}$ est un ensemble d'états discret fini ou infini. Ce processus est une CMTC si pour tout $s,t \geq 0$ et $i, j, x(u) \in \mathcal{S}$:

$$\mathbb{P}[X(t+s) = j | X(s) = i, X(u) = x(u), 0 \leq u < s] = \mathbb{P}[X(t+s) = j | X(s) = i].$$

Ceci signifie que l'état futur du processus au temps $t+s$ ne dépend que de l'état présent au temps s, et aucunement de l'historique des événements antérieurs à $X(s)$. Cette propriété *sans mémoire* est appelée la propriété de Markov. Dans cette thèse, nous considérons le centre d'appels, à chaque période, comme une CMTC *stationnaire* telle que pour tout $s,t \geq 0$:

$$\mathbb{P}[X(t+s) = j | X(s) = i] = p_{i,j}(t).$$

Le *taux de transition instantané* de l'état i à j est défini par :

$$q_{i,j} = \lim_{t \to 0} \frac{p_{i,j}(t)}{t}, \quad i \neq j.$$

Le taux de transition sortant de l'état i est la somme $q_i = \sum_{j \in \mathcal{S} \setminus \{i\}} q_{i,j}$. La *matrice génératrice* $\mathbf{Q}$ de la CMTC de taille $|\mathcal{S}| \times |\mathcal{S}|$ est formée par les éléments $q_{i,j}$ et $q_{i,i} = -q_i$.

Soit T_i le temps que le processus passe dans l'état i avant de partir vers un autre état $j \neq i$. Les propriétés sans mémoire et de stationnarité impliquent que la probabilité que le processus reste à l'état i pour une durée supplémentaire t soit indépendante du temps s passé depuis l'arrivée à l'état i :

$$\mathbb{P}[T_i > t+s | T_i > s] = \mathbb{P}[T_i > t].$$

La variable T_i suit donc une loi exponentielle. Ceci signifie que la CMTC est équivalente à un processus suivant la CMTD *imbriquée* $\{X_n = X(\tau_n), n \geq 0\}$ où $0 = \tau_0 < \tau_1 < \tau_2 < \ldots$ correspondent aux temps de saut dans la CMTC. La durée moyenne de T_i est $1/q_i$, et la *matrice de transitions* $\mathbf{P}$ de la CMTD est alors donnée par les $p_{i,j} = q_{i,j}/q_i$ et $p_{i,i} = 0$.

4.1.2 Modéliser une CMTC par une CMTD

En principe, il est possible de calculer les probabilités stationnaires $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_{|\mathcal{S}|})$ et d'obtenir une évaluation exacte (sur horizon infini) de la CMTC en résolvant le système d'équations $\boldsymbol{\pi}\mathbf{Q} = \mathbf{0}$ et $\sum_{j \in \mathcal{S}} \pi_j = 1$. Cependant, la taille de la matrice est généralement trop élevée pour que cette approche soit applicable en pratique. Ceci nous amène à l'approximation d'une CMTC par la simulation de la CMTD imbriquée.

Étant donné que nous utilisons la simulation et qu'un centre d'appels n'est pas forcément ouvert continuellement sur 24 heures, nous considérons l'évaluation du modèle de la CMTC sur un horizon fini de durée T, où T peut être la durée d'une période ou d'une journée (avec des taux stationnaires). Le temps entre deux transitions consécutives $\tau_{n+1} - \tau_n$ suit une loi exponentielle. Une première méthode pour simuler la CMTC est la suivante :

Étape 1 Supposons que le processus soit à l'état $X_n = i$ à l'étape n. Générer t selon une loi exponentielle de taux q_i. Puis affecter $\tau_{n+1} = \tau_n + t$.

Étape 2 Si $\tau_{n+1} > T$, alors terminer la simulation.

Étape 3 Générer le prochain état tel que $X_{n+1} = j$ avec probabilité $p_{i,j}$. Retourner à l'étape 1.

Simuler un modèle de CMTD a l'avantage que le simulateur n'a pas besoin de listes d'événements et requiert beaucoup moins d'*objets* (dans le contexte de la programmation orientée objet). Cependant, un désavantage de cette méthode est qu'il faut générer le même nombre de variables exponentielles qu'avec un simulateur ED pour un modèle markovien, parce que l'instant de saut dépend de l'état présent. La prochaine méthode de simulation évite cette difficulté en rendant les τ_n indépendants des états.

4.1.3 Uniformisation de la CMTC

La deuxième méthode se base sur l'*uniformisation* [53, 66, 69] de la CMTC avec des taux de transition égaux, c'est-à-dire $q_i = q_j$ pour tout $i, j \in \mathcal{S}$. Pour uniformiser la CMTC, il faut que les taux q_i soient bornés : il existe un $q < \infty$ tel que $q_i \leq q, \forall i$, et le taux d'uniformisation minimal est :

$$q_{\text{umin}} = \max_{i \in \mathcal{S}} q_i. \tag{4.1}$$

La CMTC uniformisée correspond à la CMTC originale avec l'ajout de transitions fictives. Une *transition fictive* est définie par une boucle qui sort et entre dans le même état. Soit un taux d'uniformisation q, la probabilité qu'une transition sortant de l'état i soit fictive est $1 - q_i/q$. Pour simuler la CMTC, il suffit de générer un temps exponentiel de moyenne

$1/q$, et de faire une transition de l'état présent i à l'état j avec probabilité :

$$\bar{p}_{i,j} = \begin{cases} 1 - q_i/q, & i = j \\ p_{i,j} q_i/q, & i \neq j. \end{cases}$$

Étant donné que le taux q est identique pour tous les états, il devient facile de générer le nombre aléatoire de transitions $N(T)$ dans une période T par une loi de Poisson de moyenne qT. Il est possible d'approximer analytiquement la CMTD à l'aide des probabilités stationnaires $\boldsymbol{\pi}$ obtenues en résolvant le système d'équations $\boldsymbol{\pi}\bar{\mathbf{P}} = \boldsymbol{\pi}$ et $\sum_{j\in\mathcal{S}} \pi_j = 1$, où $\bar{\mathbf{P}}$ est la matrice de transitions définie par les éléments $\bar{p}_{i,j}$. Cependant, la taille de la CMTD est généralement trop grande pour résoudre analytiquement $\boldsymbol{\pi}$, et il faut recourir à la simulation.

La deuxième méthode pour simuler la CMTC consiste à :

Étape 1 Choisir un état initial s_0 (généralement l'état sans appel et tous les agents libres).

Étape 2 Générer le nombre total de transitions $N(T)$ selon une loi de Poisson de taux qT.

Étape 3 Simuler la CMTD imbriquée $\{X_n = X(\tau_n), X_0 = s_0, n = 1, \ldots, N(T)\}$ selon la matrice de transitions $\bar{\mathbf{P}}$.

Pour générer les instants de saut τ_n, nous utilisons la propriété du processus de Poisson (voir Taylor et Karlin [124]) telle que : conditionnellement à $N(T)$, les τ_n sont distribués uniformément sur T. Il suffit donc de générer $N(T)$ variables aléatoires réparties uniformément sur T, puis de les ordonner par ordre croissant pour obtenir les $\tau_1, \ldots, \tau_{N(T)}$. Cette méthode élimine ainsi la génération d'un grand nombre de variables exponentielles. Un autre avantage avec l'uniformisation est qu'il est plus efficace d'estimer les moyennes de temps. Supposons qu'un appel entre dans la file d'attente au k-ième événement et qu'il est répondu au l-ième événement, avec $l > k$. Le temps d'attente exact simulé est $\tau_l - \tau_k$, mais si nous nous intéressons au temps d'attente moyen, ceci correspond simplement à $(l-k)/q$. Ainsi, nous pouvons estimer le temps d'attente moyen d'un appel uniquement par les indices d'événements, et éviter de générer les instants de saut τ_n.

Il y a aussi des désavantages à simuler une CMTC uniformisée. Le nombre de transitions fictives doit être relativement bas, car il arrive que le temps de simulation de la CMTD soit supérieur à celui d'un simulateur ED si q est trop grand par rapport au q_i. En principe, le meilleur choix est $q = q_{umin}$, défini dans (4.1), mais ceci réduit la synchronisation lors de l'utilisation de *variables aléatoires communes* (VAC) [9, 97], qui sont importantes pour l'optimisation. Nous présenterons les compromis possibles à faire afin d'utiliser les VAC à la section 4.3.

4.2 Simulation d'un centre d'appels par une CMTD

Nous modélisons un centre d'appels avec K types d'appels et I groupes d'agents pour une période selon un processus markovien. Les appels de type k arrivent en suivant un processus de Poisson avec un taux λ_k, et les durées de service et patience sont des variables exponentielles indépendantes et identiquement distribuées (i.i.d.) de moyennes $1/\mu_{i,k}$ et $1/\nu_k$ respectivement, telles que définies dans le chapitre 2. La probabilité qu'un appel abandonne immédiatement à l'entrée de la file k est η_k. Il y a K files d'attente et le groupe i contient y_i agents.

Un état $X(t)$ de la CMTC contient les informations suivantes : $S_{k,i} \geq 0$ le nombre d'agents du groupe i en train de servir un appel de type k et $Q_k \geq 0$ le nombre d'appels de type k en attente. Un état contient $KI + K$ variables. Évidemment, le nombre d'agents occupés ne peut pas être supérieur au nombre d'agents présents, c'est-à-dire $\sum_{k=1}^{K} S_{k,i} \leq y_i$. Nous imposons $S_{k,i} = 0$ si le groupe i ne peut pas servir le type d'appel k. Le simulateur débute à l'état initial où toutes les files d'attente sont vides et tous les agents sont libres, soient $S_{k,i} = 0$ et $Q_k = 0$ pour tout k et i.

La politique de routage implémentée dans ce simulateur est le routage par priorités P décrit à la section 2.2.2 à la page 18. Cette politique est préservatrice de travail, et elle s'incorpore bien au modèle de la CMTC puisqu'elle ne dépend que de l'information de l'état présent. Il serait également facile d'implémenter une politique de routage conditionnelle aux nombres d'agents libres ou aux longueurs des files d'attente. Par contre, la politique de routage par priorités et délais serait plus difficile à implémenter, car il faudrait ajouter les

temps d'attente aux états. Koole et al. [90] approximent un centre d'appels avec 2 types, 2 groupes et sans abandon par un modèle de CMTC avec délais en remplaçant le compteur sur la taille de chaque file d'attente par le temps d'attente du client à la tête de chaque file.

La rapidité de la simulation CMTD provient du fait que les milliers, voire les millions d'objets que la simulation ED doit générer sont remplacés par $KI + K$ compteurs à valeurs entières. Le simulateur incrémente le compteur $S_{k,i}$ de 1 quand un appel de type k arrive et se fait répondre par un agent libre du groupe i. Dans le cas où cet appel ne trouve aucun agent, il peut abandonner sur-le-champ avec probabilité η_k et l'état $X(\tau)$ de la CMTD reste inchangé, autrement l'appel entre dans la file d'attente k et le compteur Q_k est incrémenté de 1. Le simulateur décrémente $S_{k,i}$ de 1 lorsqu'un agent du groupe i finit de servir un appel de type k. Si l'agent répond à un autre appel en attente de type k', alors $S_{k',i}$ sera incrémenté de 1, sinon l'agent reste libre. Le compteur Q_k décroît de 1 quand un appel dans la file d'attente k abandonne. Le tableau 4.I résume les différents types de transitions, les conditions aux transitions et les effets engendrés dans la CMTC.

Condition	Événement	Taux	Effet
Il existe un agent libre capable de servir un appel de type k.	Arrivée d'un appel de type k.	λ_k	Incrémenter $S_{k,i}$ où i est choisi par le routeur.
Aucun agent libre capable de servir le type k.	Arrivée d'un appel de type k avec abandon immédiat.	$\eta_k\lambda_k$	Reste dans le même état.
Aucun agent libre capable de servir le type k.	Arrivée d'un appel de type k et entre dans la file d'attente.	$(1-\eta_k)\lambda_k$	Incrémenter Q_k de 1.
Il y a $Q_k > 0$ appels de type k en attente.	Un appel abandonne la file k.	$Q_k\nu_k$	Décrémenter Q_k de 1.
$S_{k,i} > 0$ agents servent des appels de type k.	Un agent du groupe i finit de servir un appel de type k.	$S_{k,i}\mu_{k,i}$	Décrémenter $S_{k,i}$, et possiblement incrémenter $S_{k',i}$ et décrémenter $Q_{k'}$ de 1 si l'agent répond à un appel de type k'.

Tableau 4.I – Résumé des transitions de la CMTC d'un centre d'appels.

Comme nous avons mentionné dans la section précédente, il faut que la somme des taux de transition soit bornée afin de pouvoir uniformiser la CMTC. Par conséquent, les

capacités des files d'attente doivent être finies, sinon les taux d'abandons augmenteraient jusqu'à l'infini. Dans notre simulateur, nous imposons une limite $H < \infty$ sur la somme totale des appels en attente, soit $\sum_{k=1}^{K} Q_k \leq H$. Quand il y a déjà H appels en attente, tout nouvel appel qui désire entrer dans une file d'attente sera bloqué. Soient $\mu_{\max,i} = \max_k \mu_{k,i}$ et $\nu_{\max} = \max_k \nu_k$, une borne supérieure sur le taux de transition de la CMTC est la somme maximale de tous les taux :

$$q = \sum_{k=1}^{K} \lambda_k + \sum_{i=1}^{I} y_i \mu_{\max,i} + H\nu_{\max}. \tag{4.2}$$

Notons qu'il pourrait exister une borne plus petite que q puisqu'à l'état où tous les agents sont occupés et les files d'attente sont remplies, les transitions d'arrivées λ_k peuvent être éliminées de la CMTC. Cependant, la borne définie par l'équation (4.2) est facile à calculer, et elle facilite la génération des transitions dans le simulateur. Nous donnerons quelques détails sur la génération des transitions à la section 4.2.1.

Le choix de la capacité H influence grandement sur l'efficacité du simulateur CMTD. Si la capacité H est trop grande, alors le simulateur peut passer la quasi totalité du temps à générer des transitions fictives et la simulation CMTD sera plus lente que la simulation ED. Par contre, si H est trop faible, alors trop d'appels risquent d'être bloqués et ceci faussera les estimations. Les taux de transition fictive sont généralement plus élevés dans les états où les agents sont peu occupés ou qu'il y a peu d'appels. Dans nos problèmes d'optimisation, nous ajustons grossièrement H, en testant le simulateur, avant d'exécuter l'optimisation. Nous choisissons H de manière à ce qu'elle soit légèrement supérieure à la longueur de file d'attente d'une solution "raisonnable" non optimisée.

La figure 4.1 présente la CMTC uniformisée pour un exemple simple d'un centre d'appels avec un type, un groupe composé de y agents et une file d'attente de capacité H. L'état $X(t)$ est défini par deux compteurs (S,Q). Les taux de transition sont :

$$\hat{\lambda}_{S,Q} = \begin{cases} \lambda, & \text{si } S < y, \\ (1-\eta)\lambda, & \text{sinon,} \end{cases}$$

$$\hat{\mu}_{S,Q} = \begin{cases} S\mu, & \text{si } S \leq y, \\ y\mu + Q\nu, & \text{sinon,} \end{cases}$$

et $\gamma_{S,Q} = q - (\hat{\lambda}_{S,Q} + \hat{\mu}_{S,Q})$ est le taux de transition fictive. Une transition qui n'existe pas a un taux de zéro. L'évolution de la CMTC étant monotone pour ce simple modèle, nous aurions pu représenter l'état par le nombre d'appels dans le centre : $X(t) = S + Q$. La CMTC devient rapidement très complexe dès qu'il y a plus qu'un type d'appel ou un groupe d'agents.

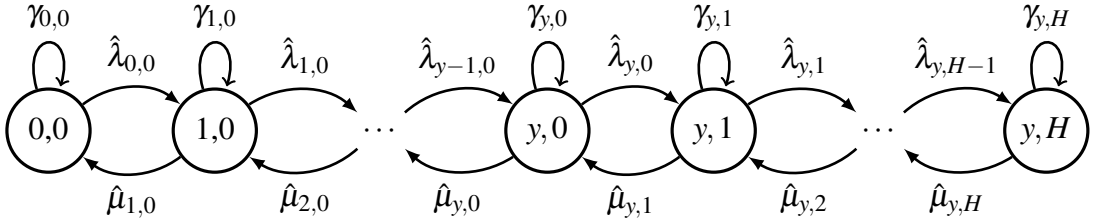


Figure 4.1 – Modèle de la CMTC uniformisée d'un centre d'appels avec un type d'appel et un groupe d'agents.

4.2.1 Simuler la CMTD par une recherche indexée

Le simulateur CMTD ne gère ni objet d'appel ou d'agent, ni de liste d'événement. À chaque instant de saut τ_n de la CMTD imbriquée, le simulateur se déplace aléatoirement en suivant une loi de probabilité discrète donnée par la matrice de transitions $\bar{\mathbf{P}}$ définie à la section 4.1.3, page 72. Nous utilisons la méthode par inversion qui est bien connue (voir Ross [115]). Supposons que l'état présent de la CMTD soit $X(\tau_n) = i$, nous avons la somme des masses de probabilité $\sum_{j=1}^{|\mathcal{S}|} \bar{p}_{i,j} = 1$, et nous divisons l'intervalle $[0,1]$ aux points suivants : $\bar{p}_{i,1}, \bar{p}_{i,1} + \bar{p}_{i,2}, \ldots, \sum_{j=1}^{|\mathcal{S}|-1} \bar{p}_{i,j}$. Nous générons une variable uniforme U dans $[0,1)$, puis nous choisissons la transition d'aller à $X(\tau_{n+1}) = m$, si $\sum_{j=1}^{m-1} \bar{p}_{i,j} \leq U < \sum_{j=1}^{m} \bar{p}_{i,j}$.

Vu le nombre élevé de transitions et d'instants de saut, il est important de pouvoir chercher la transition qui correspond à U rapidement. Il faudrait une division différente pour chaque état si la CMTC n'était pas uniformisée ; ce qui est inefficace. Mais grâce à l'uni-

formisation et par le choix de la borne q donnée par (4.2), nous utilisons une seule division indépendamment de l'état du simulateur. Nous implémentons une recherche indexée pour identifier rapidement la transition. L'intervalle $[0, 1]$ est divisé en 2^a sous-intervalles égaux, où a est un entier. Nous associons un type d'événement à chaque sous-intervalle. Selon le choix de a, il pourrait que plusieurs événements partagent le même sous-intervalle. Dans ce cas, nous divisons à nouveau ce sous-intervalle en 2^b sous-intervalles égaux, où b est un autre entier, ainsi de suite au besoin. Notons que les transitions fictives ne correspondent pas à des sous-intervalles fixes. Un sous-intervalle peut se traduire en une transition réelle ou fictive en fonction l'état présent. Par exemple, un sous-intervalle qui représente un événement d'abandon d'un appel de type k est une transition fictive si la file d'attente k est vide.

Pour générer une transition avec la recherche indexée, il suffit de générer a variables binaires (et b autres variables binaires au besoin) pour sélectionner un des 2^a sous-intervalles. Si a est grand, il y a plus de chance de générer une transition en une opération, mais le tableau de recherche occupera plus d'espace mémoire. Si a est petit, il faudra faire plusieurs sous-divisions, donc plus d'opérations à exécuter. Buist [28] propose une formule pour choisir a et donne plus de détails sur la recherche indexée.

4.2.2 Estimation du niveau de service

Le niveau de service est une des principales mesures de performance utilisées dans les centres d'appels. Dans cette section, nous décrivons une méthode pour estimer le SL avec la simulation CMTD. Buist [28] présente d'autres méthodes pour estimer diverses mesures de performance et fonctions plus générales.

Nous nous intéressons à f_S, l'espérance du SL à long terme, définie par l'équation (2.1) à la page 20. Au lieu d'estimer f_S en mesurant directement les temps d'attente comme dans un simulateur ED, nous pouvons estimer f_S en fonction du nombre de transitions passées en attente. Chaque file d'attente est modélisée par un tableau dont les éléments sont les numéros des instants d'arrivée des appels, c'est-à-dire un appel qui arrive au temps τ_n aura le numéro n. Si la transition à l'instant τ_n correspond à l'arrivée d'un appel de type k et qu'aucun agent n'est libre pour le servir, alors nous ajoutons le nombre n à la fin du

tableau k. Lorsqu'un agent devient libre et choisit de servir un appel de la file d'attente k, nous retirons le numéro à la tête du tableau k. Lorsqu'un appel de type k abandonne, nous retirons un chiffre au hasard, avec probabilité égale, parmi tous les éléments du tableau k. Retirer un élément du tableau requiert de décaler tous les éléments qui suivent. Ceci n'est pas efficace, en particulier si nous retirons l'appel à la tête de la file (ce qui arrive fréquemment). Nous utilisons des tampons circulaires pour gérer plus efficacement les tableaux. Nous aurions pu utiliser une liste doublement chaînée, mais nous voulons éviter de créer des objets.

Le temps d'attente W d'un appel, arrivé au temps τ_{n_1} et répondu ou ayant abandonné au temps τ_{n_2}, dépend du nombre de transitions passées $D = n_2 - n_1$ entre ces deux événements. Il se trouve que conditionnellement à $N(T)$ et $D \leq N(T)$, le temps d'attente W a la même distribution que les D premières transitions, ce qui correspond à la distribution de la statistique d'ordre de rang D d'un échantillon composé de $N(T)$ variables uniformes réparties sur l'intervalle $[0,T]$, voir David [48] sur la statistique d'ordre. Nous avons :

$$\mathbb{P}[W > s|N(T),D] = \mathbb{P}[B < D] = \sum_{j=0}^{D-1} \binom{N(T)}{j} (s/T)^j (1-s/T)^{N(T)-j},$$

où s est le temps d'attente acceptable et B est une variable binomiale de paramètres $n = N(T)$ et $p = s/T$. Pour accélérer la simulation, nous calculons au préalable la distribution de B jusqu'à une valeur proche de 1, mais comme elle dépend de $N(T)$, nous devons recalculer cette distribution à chaque nouvelle réplication.

4.3 Optimisation à l'aide d'un simulateur d'une CMTC uniformisée

Dans cette section, nous regardons comment utiliser le simulateur CMTD dans l'optimisation de l'affectation des agents dans un centre d'appels. Nous présentons les adaptations apportées à l'algorithme de coupes par sous-gradients de Cezik et L'Ecuyer [37], décrit à la section 3.3.2, page 42. En somme, le problème consiste à minimiser le coût des agents sur une période tout en satisfaisant des seuils minimaux de SL pour un centre

d'appels avec K types et I groupes :

$$\begin{array}{rl} \min & \sum_{i=1}^{I} c_i y_i \\ \text{sujet à :} & g_k(\mathbf{y}) \geq l_k, \quad k = 1, \ldots, K, \\ & g(\mathbf{y}) \geq l, \\ & \mathbf{y} \geq 0 \text{ et entiers,} \end{array} \tag{P1}$$

où $\mathbf{y} = (y_1, \ldots, y_I)^{\mathrm{T}}$ est le vecteur d'agents, c_i est le coût d'un agent du groupe i, g_k est la fonction du SL du type k, g est le SL agrégé sur tous les appels, et l_k et l sont les seuils minimaux correspondants. Notons que g correspond à la fonction f_{S} définie à la section 2.3, mais g varie en fonction du vecteur d'agents $\mathbf{y}$. Cet algorithme fait l'hypothèse que les parties supérieures des fonctions g_k et g soient concaves (régions qui représentent de bons SL). La technique de coupes par sous-gradients est utilisée pour éliminer les solutions irréalisables jusqu'à l'obtention d'une solution réalisable. Cette méthode est une heuristique puisque nous n'avons aucune preuve que g et g_k sont réellement concaves. Les modifications que nous présentons dans cette section sont importantes, parce que remplacer bêtement le simulateur ED par un simulateur CMTD peut augmenter le temps d'optimisation.

La première idée simple est de réduire le nombre de transitions fictives en limitant la capacité H des files d'attente, définie à la section 4.2. Mettre une capacité trop élevée, même si elle correspond à la capacité réelle, peut facilement augmenter le nombre de transitions fictives. Le nombre d'appels en attente est souvent bas, car les problèmes d'optimisation cherchent habituellement des solutions qui ont de bonnes qualités de service. En général, nous nous attendons à ce que le nombre d'appels en attente grandit proportionnellement avec la racine des charges d'appels : $\rho_{\max} = \sum_{k=1}^{K} \sqrt{\lambda_k / \mu_{k,\min}}$, où $\mu_{k,\min} = \min_i \{\mu_{k,i} | \mu_{k,i} > 0\}$. Le choix de la racine carrée est basé sur le principe du "*square-root safety formula*" basé sur le régime de Halfin-Whitt [72, 134]. Au lieu de configurer H en fonction de la capacité la file d'attente, il vaut mieux choisir H par rapport au nombre d'appels qui seront en attente. Nous suggérons de placer une limite de capacité $H = 100$ ou $a\rho_{\max}$ avec un petit facteur a.

4.3.1 Simulation avec des variables aléatoires communes (VAC)

L'algorithme d'optimisation estime les sous-gradients de la fonction g à l'aide de la méthode des différences finies. Nous employons le terme "sous-gradient", parce que g est discrète et non différentiable. Nous approximons l'élément j du sous-gradient $\mathbf{q}(\bar{\mathbf{y}})$ de g au point $\bar{\mathbf{y}}$ avec un pas entier de longueur d par

$$q_j(\bar{\mathbf{y}}) = \frac{g(\bar{\mathbf{y}} + d\mathbf{e}_j, \boldsymbol{\xi}_2) - g(\bar{\mathbf{y}}, \boldsymbol{\xi}_1)}{d},$$

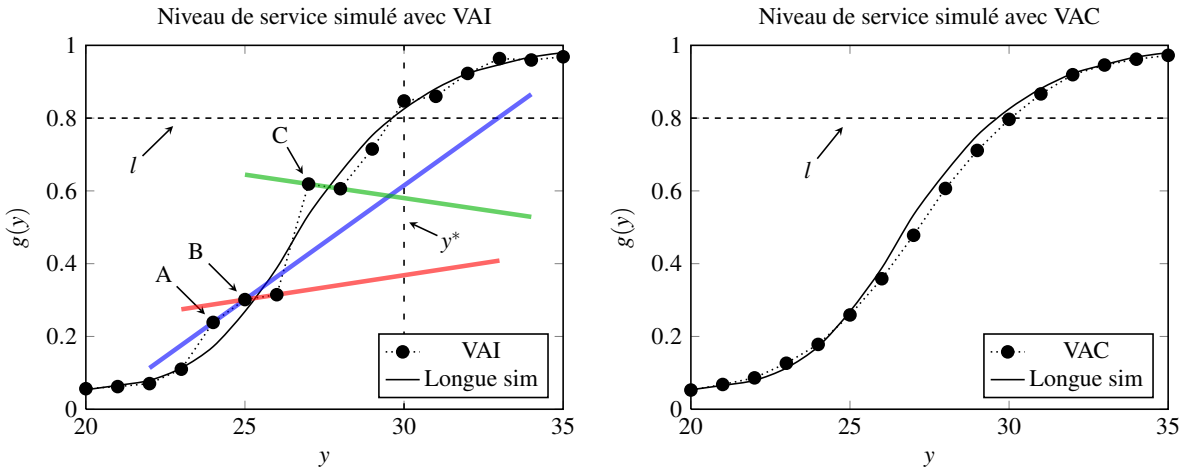
où $\mathbf{e}_j$ est un vecteur unitaire avec l'élément 1 à la position j et 0 ailleurs. Nous explicitons les séquences de nombres aléatoires, représentées par $\boldsymbol{\xi}_1$ et $\boldsymbol{\xi}_2$, utilisées par le simulateur pour calculer g. La variance de l'estimateur $q_j(\bar{\mathbf{y}})$ est :

$$\mathrm{Var}[q_j(\bar{\mathbf{y}})] = \frac{\mathrm{Var}[g(\bar{\mathbf{y}} + d\mathbf{e}_j, \boldsymbol{\xi}_2)] + \mathrm{Var}[g(\bar{\mathbf{y}}, \boldsymbol{\xi}_1)] - 2\mathrm{Cov}[g(\bar{\mathbf{y}} + d\mathbf{e}_j, \boldsymbol{\xi}_2), g(\bar{\mathbf{y}}, \boldsymbol{\xi}_1)]}{d^2}.$$

Si nous utilisons des *variables aléatoires indépendantes* (VAI), c'est-à-dire que $\boldsymbol{\xi}_1$ et $\boldsymbol{\xi}_2$ sont des séquences indépendantes, alors le terme de covariance sera nul. Par contre, si $\boldsymbol{\xi}_1$ et $\boldsymbol{\xi}_2$ sont corrélées positivement, en particulier si $\boldsymbol{\xi}_1 = \boldsymbol{\xi}_2$, alors la covariance sera positive et la variance de $q_j(\bar{\mathbf{y}})$ sera réduite. Cette technique s'appelle la réduction de variance par *variables aléatoires communes* (VAC) [9, 97], et elle joue un rôle important sur l'efficacité des coupes par sous-gradients. Il est vrai que si les simulations sont assez longues, alors la variance de $q_j(\bar{\mathbf{y}})$ sera négligeable dans les deux cas. Cependant, l'algorithme d'optimisation doit souvent simuler des centaines ou des milliers de solutions, et il ne serait pas abordable en pratique d'exécuter de longues simulations. L'optimisation doit alors faire appel à de courtes simulations qui risquent d'être fortement bruitées, ainsi la technique VAC devient encore plus importante.

La figure 4.2, inspirée de Ceẓik et L'Ecuyer [37], présente le danger de générer des coupes par sous-gradients lorsque les estimations sont très bruitées. La figure montre un échantillon typique simulé du SL en fonction du nombre d'agents d'un centre d'appels avec un type d'appel et un groupe d'agents. La solution optimale est y^* et le SL minimal est $l = 80\%$. La courbe solide est obtenue à l'aide de longues simulations précises et les

courbes pointillées sont générées par de courtes simulations avec VAI dans la figure 4.2(a) et VAC dans la figure 4.2(b). Les coupes générées avec VAI aux points A et B élimineront la solution optimale, alors que la coupe calculée au point C sera invalide puisque la pente du sous-gradient est négative. Les bruits sont totalement aléatoires et indépendants dans la simulation avec VAI, alors que les bruits sont corrélés dans le cas avec VAC. Notons que les coupes générées avec VAC peuvent aussi éliminer la solution optimale, mais l'erreur est beaucoup plus petite.



(a) VAI (b) VAC

Figure 4.2 – Comparaison entre l'utilisation de variables aléatoires indépendantes (a) et de variables aléatoires communes (b). Les coupes basées sur les sous-gradients estimés aux points A, B et C sont problématiques.

Avec un simulateur ED, il est facile d'implanter la technique VAC en utilisant un différent générateur de nombres aléatoires pour chaque processus ou variable aléatoire. Cette méthode est particulièrement efficace si les variables aléatoires ne dépendent pas de la solution (du nombre d'agents dans notre cas), car un seul générateur peut être utilisé pour toutes ces variables. L'Ecuyer et Buist [97] proposent de générer un historique de clients où chaque client a un profil défini par son type d'appel, son temps d'arrivée, sa durée de service requise par groupe et sa durée de patience. En simulant ce même historique de clients pour toutes les solutions $\mathbf{y}$, les estimations seront toutes corrélées positivement.

Par contre, il est moins évident d'implanter les VAC dans un simulateur CMTD, car

un seul générateur de nombres aléatoires est utilisé durant toute la simulation. Le nombre total de transitions $N(T)$ est généré au début, puis les transitions de la CMTD sont générées séquentiellement. Pour synchroniser les simulations, il faut au moins que le taux maximal q demeure constant durant toute l'optimisation. Les taux d'arrivée des appels ne changent pas, mais il faut choisir une limite maximale $\hat{y}_i$ sur le nombre d'agents pour chaque groupe i. Nous remplaçons y_i par $\hat{y}_i$ dans l'équation (4.2) et nous obtenons

$$\hat{q} = \sum_{k=1}^{K} \lambda_k + \sum_{i=1}^{I} \hat{y}_i \mu_{\max,i} + H\nu_{\max}. \tag{4.3}$$

Imposer une limite $\hat{y}_i$ sur le nombre d'agents par groupe crée des inconvénients importants. Premièrement, ceci demande à choisir un hypercube à I dimensions qui couvriraient toutes les solutions $\mathbf{y}$ à évaluer durant l'optimisation ainsi que la solution optimale $\mathbf{y}^*$ (qui n'est pas nécessairement unique). Il est très difficile d'avoir une bonne estimation de $\mathbf{y}^*$ dès qu'il y a plusieurs groupes. Comme nous optimisons des centres d'appels multi-compétences, il peut avoir une quantité astronomique de solutions possibles. Un choix naïf pour garder $\hat{q}$ constant serait de sélectionner une limite $\hat{\mathbf{y}} = (\hat{y}_1, \ldots, \hat{y}_I)^{\mathrm{T}}$ très grande. D'autre part, prendre $\hat{y}_i > y_i$ augmente inévitablement le nombre de transitions fictives. Notamment si nous choisissons une limite $\hat{\mathbf{y}}$ très grande, il est probable que les transitions fictives rendent la simulation CMTD plus lente que la simulation ED.

Nous proposons un compromis entre la réduction de variance par VAC et la restriction sur le nombre de transitions fictives. Nous appelons cette approche VAC-SG. L'idée est d'utiliser la technique VAC uniquement lors de l'estimation d'un sous-gradient, car l'optimisation est plus sensible à la qualité des coupes par sous-gradients. Si le sous-gradient au point $\bar{\mathbf{y}}$ est estimé par la méthode des différences finies de longueur d, alors le vecteur d'agents maximaux $\hat{\mathbf{y}} = \bar{\mathbf{y}} + \mathbf{d}$, où $\mathbf{d} = (d, \ldots, d)^{\mathrm{T}}$. La figure 4.3 montre des exemples où nous comparons les sous-gradients estimés avec VAC-SG et VAI, et un pas $d = 1$. Les simulations demeurent très bruitées tout au long de l'optimisation, mais les sous-gradients estimés avec VAC-SG sont plus fiables. Les points A, B et C, qui généraient de mauvaises coupes dans la figure 4.2, n'éliminent plus la solution optimale y^* dans la figure 4.3. Un sous-gradient peut être mauvais même s'il n'élimine pas la solution optimale. Prenons le

cas de la coupe générée au point D dans la figure 4.3(d). Avec VAI, cette coupe est plutôt inutile et gaspille du temps d'optimisation, alors qu'elle serait efficace si elle est calculée avec VAC-SG.

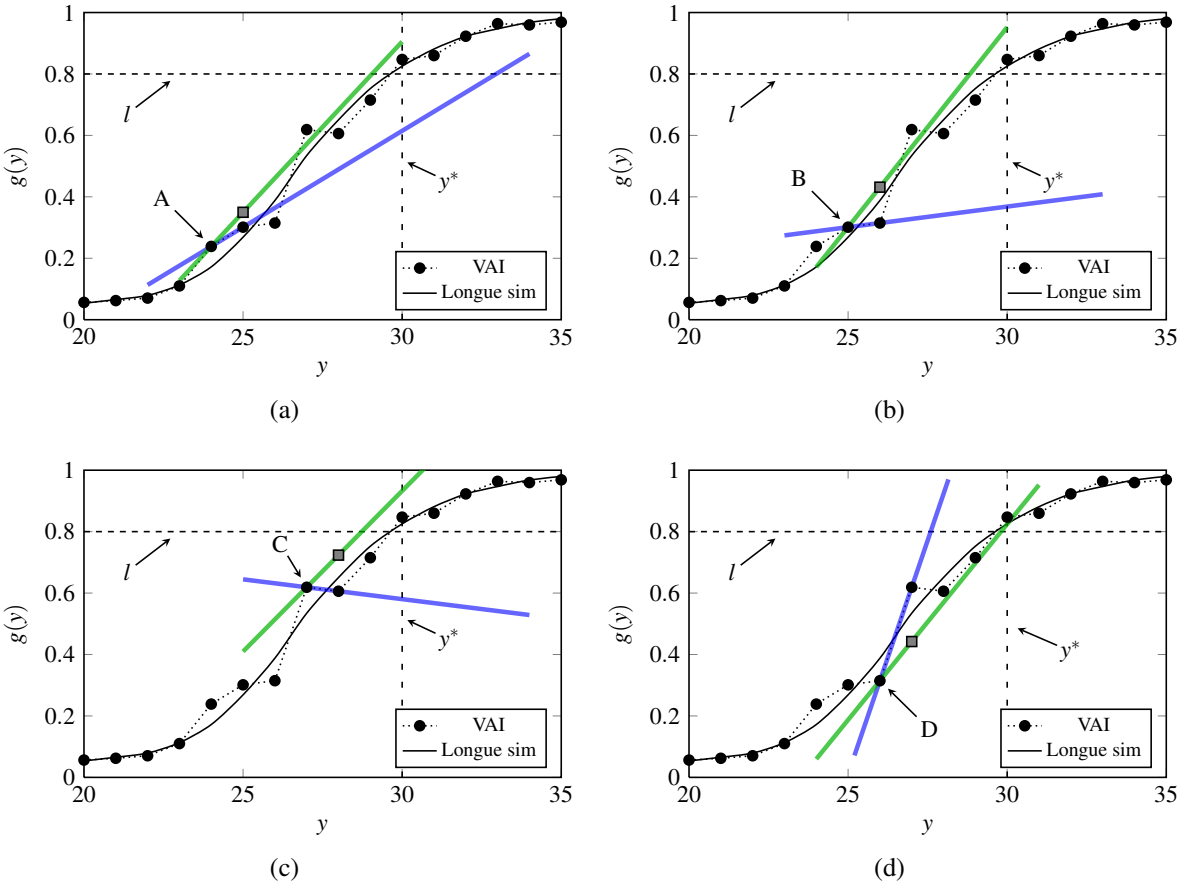


Figure 4.3 – Utilisation des VAC-SG (lignes vertes) et VAI (lignes bleues) pour l'estimation du sous-gradient. Le carré est simulé avec les mêmes VAC que le point qui le précède.

Les expériences numériques montrent que le compromis VAC-SG est profitable, car le nombre de transitions fictives générées est similaire au cas VAI, mais la qualité des solutions est comparable au cas où la technique VAC est appliquée sur toute l'optimisation avec $\hat{q}$ très grand. Notons que simuler toutes les solutions avec les mêmes VAC revient à optimiser un échantillon particulier du problème et la solution est alors biaisée. Le biais se réduit en augmentant les simulations, mais nous pouvons aussi réduire un peu ce biais en mixant les VAI et VAC.

4.3.2 Autre variante pour la simulation avec VAC

Par ailleurs, nous avons essayé une autre variante de l'application des VAC, que nous appelons VAC-B. Au lieu de changer $\hat{q}$ à chaque itération comme avec VAC-SG, nous tentons de choisir le vecteur $\hat{\mathbf{y}}$ de manière à pouvoir utiliser le même $\hat{q}$ et la même recherche indexée sur plusieurs itérations consécutives. Notre approche est de commencer avec un vecteur initial $\hat{\mathbf{y}}^{(0)}$ et d'incrémenter les éléments $\hat{y}_i$ au besoin si $\bar{y}_i + d > \hat{y}_i$. Ainsi, l'hypercube défini par $\hat{\mathbf{y}}$ augmente de façon monotone tout au long de l'optimisation tel que $\hat{y}_i^{(0)} \leq \hat{y}_i^{(1)} \leq \hat{y}_i^{(2)} \leq \dots$ pour tout i, où $\hat{y}_i^{(m)}$ est la valeur de $\hat{y}_i$ à l'itération m. La procédure de mise à jour de $\hat{y}_i$ est comme suit. Au moment d'estimer le sous-gradient au point $\bar{\mathbf{y}}$, si nous avons $\bar{y}_i + d > \hat{y}_i$, alors mettre à jour $\hat{y}_i = \lceil (1+\beta)(\bar{y}_i + d) \rceil$, où $\beta \geq 0$ est un paramètre, et $\lceil x \rceil$ retourne le plus petit entier égal ou supérieur à x. Choisir un petit β réduit le nombre de transitions fictives, mais un grand β diminue la fréquence de changements de $\hat{\mathbf{y}}$, ce qui permet d'utiliser les mêmes VAC sur plusieurs itérations. Notons que choisir $\beta = 0$ n'est pas équivalent au compromis VAC-SG, parce que nous ne réduisons pas les $\hat{y}_i$ avec VAC-B.

Dans les expériences numériques, nous avons essayé $\beta = \{0, 0.1, 0.2, 0.5, 2\}$. Les résultats ne montrent pas de différence significative entre les coûts des solutions avec les différents β et VAC-SG. L'approche VAC-B obtient de bonnes solutions réalisables un peu plus fréquemment qu'avec VAC-SG. Mais l'approche VAC-SG est la plus rapide, suivie de VAC-B avec $\beta = 0$, puis $\beta = 0.1$, ainsi de suite. Dans un exemple, le temps moyen d'optimisation avec $\beta = 2$ était presque le double qu'avec $\beta = 0$. Nous suggérons d'utiliser l'VAC-SG ou VAC-B avec β proche de 0.

4.4 Exemples numériques

Dans cette section, nous comparons les performances des simulateurs ED et CMTD pour différents exemples de centres d'appels. Nous commençons par comparer simplement les simulateurs avec tous les paramètres fixes afin d'évaluer leur vitesse de simulation. Par la suite, nous comparons les performances dans le contexte d'un problème d'optimisation des agents. Les simulateurs ED et CMTD sont disponibles dans la librairie de simulation

ContactCenters de Buist et L'Ecuyer [31]. Toutes les expériences numériques ont été exécutées à l'aide d'un processeur AMD Opteron 2.0Ghz, à l'exception de l'exemple 3 et des tableaux 4.III et 4.V qui ont été réalisés plus tard avec un processeur beaucoup plus rapide, un Intel i7 3.40Ghz. Les temps d'exécution de ces derniers sont alors plus courts.

Exemple 1. Nous avons un centre d'appels avec un type d'appel et un groupe d'agents. Les durées de service et de patience sont des variables exponentielles de moyennes $\mu_1^{-1} = 100$ et $\nu_1^{-1} = 1000$ secondes, respectivement, et la probabilité d'abandon immédiat est $\eta_1 = 0.1$. Nous varions le taux d'arrivée λ_1 et le nombre d'agents.

Exemple 2. Nous considérons un centre composé de 3 types d'appels et 6 groupes d'agents. Les trois premiers groupes ont une seule habileté : $\mathcal{S}_1 = \{1\}$, $\mathcal{S}_2 = \{2\}$ et $\mathcal{S}_3 = \{3\}$, et les trois autres groupes ont chacun 2 habiletés : $\mathcal{S}_4 = \{1,3\}$, $\mathcal{S}_5 = \{1,2\}$ et $\mathcal{S}_6 = \{2,3\}$. Pour tout k, la probabilité d'abandon immédiat est $\eta_k = 0.01$ et le temps de patience est une variable exponentielle de moyenne $\nu_k^{-1} = 1000$ secondes. Les taux de service par seconde sont $\mu_{1,1} = \mu_{2,2} = \mu_{3,3} = 11/3600$ et $\mu_{1,4} = \mu_{1,5} = \mu_{2,5} = \mu_{2,6} = \mu_{3,4} = \mu_{3,6} = 1/360$. Les agents spécialistes sont 10% plus rapides que leurs collègues possédant deux habiletés. Le centre utilise la politique de routage par priorités P, définie à la section 2.2.2, page 18, avec les listes de priorités suivantes : $\mathcal{L}_1 = (1), \mathcal{L}_2 = (2), \mathcal{L}_3 = (3)$, $\mathcal{L}_4 = (1,3), \mathcal{L}_5 = (2,1), \mathcal{L}_6 = (3,2)$, $\mathcal{G}_1 = (1,4,5), \mathcal{G}_2 = (2,5,6)$ et $\mathcal{G}_3 = (3,6,4)$. Les priorités sont strictes (aucune égalité) dans cet exemple. Soit le taux d'arrivée total λ, les taux d'arrivée par type d'appel sont $\lambda_1 = \lambda_2 = 4\lambda/11$ et $\lambda_3 = 3\lambda/11$. Nous varions λ et le vecteur $\mathbf{y} = (y_1, \ldots, y_6)^{\mathrm{T}}$ dans nos exemples.

Exemple 3. Nous considérons le grand exemple basé sur un centre d'appels réel présenté dans Ceşik et L'Ecuyer [37] et Avramidis et al. [18]. Cet exemple est en fait un centre d'appels virtuel formé de deux centres d'appels localisés dans deux régions (villes) différentes. La région 1 possède 22 types d'appels et 15 groupes d'agents, et la région 2 compte 43 types et 74 groupes. Il y a au total 65 types d'appels et 89 groupes. Le nombre d'habiletés qu'un agent possède varie de 1 à 24. Les arrivées suivent des processus de Poisson et les taux λ_k varient de 1.046 à 416.6 par heure, avec un taux agrégé de 3581.7 appels par heure. La durée de service d'un appel ne dépend que de son type, et non de l'agent qui répond. Les taux de service μ_k varient de 0.6777 à 600. La charge de travail agrégée

est $\sum_{k=1}^{65} \lambda_k/\mu_k = 500$. Les durées de patience sont des variables exponentielles de moyennes $\nu_k^{-1} = 3$ minutes. Dans l'exemple original, la politique de routage utilisée s'appelle *spécialiste local* (SPEC) et elle est une variante de la politique de routage par priorités. Au moment d'arrivée d'un appel et pendant les 6 premières secondes d'attente, seuls les agents appartenant à la même région que l'appel peuvent y répondre (en suivant un routage par priorités). Dès que l'appel attend au-delà de 6 secondes, il peut être répondu par les agents des deux régions, avec priorité donnée aux agents locaux. Vu que notre simulateur CMTD n'a pas implanté cette politique, nous simulons cet exemple avec la politique de routage par priorités, en ignorant la condition de délai de 6 secondes.

4.4.1 Expériences de simulations

Le tableau 4.II présente les résultats de la comparaison entre les simulations CMTD et ED pour les 3 exemples. Chaque exemple est simulé avec 1000 réplications indépendantes et un temps d'horizon $T = 46\,800$ secondes (ou 13 heures). Pour chaque centre d'appels, nous expérimentons quatre différents taux d'arrivée qui se trouvent sous la colonne λT. Le temps d'attente acceptable du SL est de 20 secondes dans tous les exemples. Les vecteurs $\mathbf{y}$ sont choisis de manière à ce que tous les types d'appels ont un SL autour de 80% pour les exemples 1 et 2. Pour l'exemple 2, les vecteurs $\mathbf{y}$ sont $(36,35,27,3,5,4)^{\mathrm{T}}$, $(75,71,50,5,6,14)^{\mathrm{T}}$, $(114,108,85,3,8,13)^{\mathrm{T}}$ et $(149,149,106,9,4,14)^{\mathrm{T}}$, pour des totaux de 110, 221, 331 et 431 agents respectivement. Pour l'exemple 3, le vecteur $\mathbf{y}$ a été optimisé pour satisfaire des seuils de SL à 50% par type d'appel et à 80% agrégé sur tous les appels. Le nombre total d'agents est 484. La colonne $\sum y_i$ donne le nombre total d'agents. La capacité H est suffisamment grande pour éviter d'avoir des appels bloqués. La colonne $\mathbb{E}[N(T)]$ donne l'espérance du nombre total de transitions, ce qui correspond à qT. Puisque nous ne faisons pas d'optimisation dans ce test, nous avons $\hat{\mathbf{y}} = \mathbf{y}$, ce qui implique $\hat{q} = q$, afin de réduire le nombre de transitions fictives. Les dernières colonnes comparent les temps de simulation et donnent les ratios de vitesse.

Les résultats montrent que la simulation CMTD est plus rapide que la simulation ED. Nous observons que l'augmentation du taux d'arrivée réduit la proportion de transitions fictives et augmente le ratio de vitesse du simulateur CMTD. Une raison de l'amélioration

	λT	$\sum y_i$	H	$\mathbb{E}[N(T)]$	Trans. fictives	Temps CMTD	Temps ED	Ratio
Ex. 1	1 660	5	30	5 404	39.6%	1s	6s	4.6
	25 000	52	80	57 292	8.7%	11s	1m 29s	8.2
	50 000	100	130	104 756	6.4%	18s	3m 08s	10.0
	75 000	148	170	152 220	5.3%	25s	4m 59s	12.0
Ex. 2	14 300	110	100	34 554	17.7%	9s	59s	6.6
	30 000	221	100	65 958	9.5%	19s	2m 15s	7.1
	45 000	331	150	99 041	9.6%	27s	3m 29s	7.7
	60 000	431	150	128 302	7.0%	37s	4m 38s	7.5
Ex. 3	46 562	484	100	224 198	58.7%	1m 05s	1m 17s	1.18
	46 562	484	1 000	458 198	79.8%	1m 28s	1m 18s	0.89
	46 562	484	10 000	2 798 198	96.7%	4m 42s	1m 16s	0.27

Tableau 4.II – Comparaison des temps de simulation entre les simulateurs CMTD et ED.

du ratio de vitesse, lorsque λ augmente, est attribuable au fait que le simulateur CMTD utilise seulement des compteurs, alors que le simulateur ED doit créer de plus en plus d'objets. Le simulateur CMTD peut être jusqu'à 12 fois plus rapide que le simulateur ED pour l'exemple 1, et jusqu'à 7.7 fois plus rapide pour l'exemple 2. Quant au grand exemple 3, le simulateur CMTD avec $H = 100$ est 18% plus rapide que le simulateur ED, mais il peut devenir 3.7 fois plus lent si H est inutilement grand. Il est plus difficile de réduire la quantité de transitions fictives dans le grand centre d'appels, car il y a beaucoup de types de transitions possibles. Comme attendu, le temps du simulateur ED ne change quasiment pas, même si la capacité des files d'attente passe de 100 à 10 000.

Dans le tableau 4.III, nous expérimentons différents paramètres de capacité H et du maximum d'agents $\hat{\mathbf{y}}$ sur l'exemple 2 avec $\lambda T = 14\,300$. Ce test s'applique davantage à un contexte d'optimisation où nous ne connaissons pas les bonnes valeurs de H et $\hat{\mathbf{y}}$. Quand la colonne $\hat{y}_i$ vaut y_i, ceci signifie que $\hat{\mathbf{y}} = \mathbf{y}$. La capacité des files d'attente n'affecte pas en général le temps de simulation ED, et nous fixons arbitrairement cette capacité à 10 000. Comme prévu, la proportion de transitions fictives et le temps de simulation augmentent en fonction de H et $\hat{\mathbf{y}}$. Le simulateur CMTD peut même devenir plus lent que le simulateur ED.

Simulateur	H	$\hat{y}_i$	Trans. fictives (%)	Temps (sec)
ED	-	-	-	24.9
CMTD	100	y_i	17.7	4.5
	200	y_i	27.5	5.0
	500	y_i	46.6	6.3
	1000	y_i	62.9	8.5
	10 000	y_i	94.3	40.8
	100	50	52.5	7.2
	100	100	71.8	9.7
	100	200	84.4	16.4
	100	500	93.4	35.6
	500	50	63.8	8.9
	500	100	76.2	12.0
	500	200	82.9	18.4
	500	500	93.6	38.3

Tableau 4.III – Exemple 2 avec $\lambda T = 14\,300$: comparaison des temps de simulation entre les simulateurs CMTD et ED.

4.4.2 Optimisation d'un modèle markovien

Dans cette section, nous considérons un problème d'affectation de l'exemple 2 avec $\lambda T = 14\,300$. Le problème est formulé sous la même forme que le problème (P1) défini à la section 4.3, page 79. Le vecteur de coût par agent est $\mathbf{c} = (1, 1, 1, 1.05, 1.05, 1.05)$ et les seuils minimaux des SL sont $l_k = l = 80\%$ avec un AWT de 20 secondes. Nous optimisons le problème à l'aide de l'algorithme de coupes par sous-gradients de Ceẓik et L'Ecuyer [37]. Nous appelons CP-CMTD lorsque l'algorithme de coupes utilise le simulateur CMTD, et nous nommons CP-ED dans le cas avec le simulateur ED. Les problèmes linéaires à nombres entiers sont résolus à l'aide de CPLEX 9.0. Nous exécutons la recherche locale de Avramidis et al. [18] pour raffiner la solution finale. La recherche locale utilise le même simulateur que l'algorithme de coupes. Nous trouvons la solution optimale empirique en exécutant l'algorithme d'optimisation avec un très grand budget de temps. Nous obtenons le vecteur optimal empirique $\mathbf{y}^* = (36, 35, 27, 3, 5, 4)^{\mathrm{T}}$ avec un coût total de 110.60. Cette solution est identique à celle testée pour le même exemple à la section 4.4.1.

Nous commençons par comparer les performances des algorithmes CP-CMTD et CP-

ED avec un budget d'exécution moyen de 2 minutes (incluant la recherche locale). Nous n'imposons pas une limite de temps stricte, donc les algorithmes ne sont jamais interrompus avant la fin. Nous contrôlons le temps d'optimisation en fonction du temps de simulation puisque que la simulation occupe la majorité des temps d'exécution. Ainsi, le simulateur CMTD exécute 40 réplications et le simulateur ED exécute 8 réplications à chaque évaluation d'une solution. Le temps occupé par CPLEX est négligeable. Lors de l'estimation du sous-gradient, nous prenons un pas $d = 2$, et nous utilisons l'approche VAC-SG dans CP-CMTD. La capacité des files d'attente est fixée à $H = 100$. Un désavantage d'avoir des contraintes dures est qu'une solution peut être déclarée irréalisable à cause du bruit de la simulation. Dans un tel cas, il est commun d'accepter une solution avec une tolérance d'erreur ε, et nous disons qu'une solution est ε-réalisable lorsque les SL sont égaux ou supérieurs à $(l-\varepsilon)\%$ pour tous les seuils. Par exemple, nous disons qu'une solution est réalisable pour $\varepsilon = 0.1$ si aucun SL n'est inférieur à 79.9%.

Nous effectuons 100 exécutions indépendantes des algorithmes CP-CMTD et CP-ED, et nous rapportons les résultats dans le tableau 4.IV. Pour vérifier la ε-réalisabilité, les solutions finales sont simulées à l'aide d'un simulateur ED avec 2000 réplications. Les largeurs des intervalles de confiance à 95% obtenues pour les SL sont inférieures à 0.4%. Ces estimations sont alors relativement précises. Parmi les F solutions ε-réalisables, le tableau montre les coûts minimal et médian, suivis du nombre de solutions ayant un coût au plus $1+0.01z$ fois supérieur au coût optimal empirique dans la colonne O_z.

Algorithme	ε	Min	Méd	F	$O_{0.2}$	$O_{0.5}$	O_1
CP-ED	0.01	110.85	111.50	36	0	9	23
CP-CMTD	0.01	110.70	111.40	73	16	27	58
CP-ED	0.1	110.70	111.48	42	1	13	28
CP-CMTD	0.1	110.65	111.35	80	20	32	65

Tableau 4.IV – Exemple 2 avec $\lambda T = 14\,300$: comparaison des performances d'optimisation avec les simulateurs CMTD et ED, basée sur 100 exécutions.

Les résultats montrent que CP-CMTD et CP-ED sont tous les deux capables de trouver de bonnes solutions. Les coûts minimal et médian trouvés par CP-CMTD sont légèrement meilleurs à ceux obtenus par CP-ED. Cependant, CP-CMTD a environ deux fois plus

de chance de trouver une solution ε-réalisable que CP-ED. Ceci s'explique par les faits que l'exemple est un système markovien et la rapidité du simulateur CMTD nous permet d'avoir de meilleures approximations en simulant 5 fois plus de réplications. Pour la majorité des exécutions de CP-CMTD et CP-ED, l'algorithme de coupes s'arrête après 60 à 90 secondes, et le reste du temps est occupé par la recherche locale. Naturellement, les deux algorithmes trouvent moins fréquemment de bonnes solutions ε-réalisables lorsque la tolérance d'erreur est plus stricte (ε plus petit).

4.4.3 Optimisation d'un modèle non markovien

Dans cette expérience, nous modifions le centre d'appels de l'exemple 2 pour avoir un modèle non markovien. Nous gardons les mêmes moyennes de taux d'arrivée et de temps de service, mais nous changeons la famille de lois de probabilité. La loi de probabilité des temps de patience demeure cependant inchangée.

Premièrement, nous considérons des processus d'arrivée doublement stochastiques. Nous ajoutons un *facteur d'achalandage* journalier aléatoire B qui multiplie les taux d'arrivée $(\lambda_1, \lambda_2, \lambda_3) = (400, 400, 300)$ par heure. Le processus d'arrivée du type k reste Poisson, mais son taux $\Lambda_k = B\lambda_k$ est aléatoire et varie d'une journée à l'autre. Dans notre simulateur, le facteur B est unique et s'applique à tous les types d'appels. (Il est possible aussi d'avoir un facteur B_k indépendant pour chaque type d'appel k.) Les taux d'arrivée des types d'appels sont alors positivement corrélés. Nous modélisons B par une loi triangulaire, qui est souvent utilisée en pratique à cause de sa simplicité. Nous choisissons une loi triangulaire symétrique définie sur l'intervalle $[0.7, 1.3]$ avec un mode et une espérance de 1. Les taux d'arrivée varient d'une journée à l'autre de 70% à 130% du taux moyen.

Par ailleurs, nous remplaçons les lois exponentielles des durées de service par des lois log-normales, tout en gardant les mêmes espérances. Soient $\kappa_{k,i}$ et $\sigma_{k,i}$ les paramètres d'échelle et de forme de la loi log-normale pour le type d'appel k et le groupe i, l'espérance est $e^{\kappa_{k,i}+\sigma_{k,i}^2/2}$. Les paramètres d'échelle pour les agents spécialistes sont $\kappa_{1,1} = \kappa_{2,2} = \kappa_{3,3} = -2.8979$, et pour les autres, $\kappa_{1,4} = \kappa_{1,5} = \kappa_{2,5} = \kappa_{2,6} = \kappa_{3,4} = \kappa_{3,6} = -2.8026$. Les paramètres de forme sont $\sigma_{k,i} = 1$ pour tout k et tout i. Ces distributions génèrent des durées de service en unité d'heure. Les durées moyennes sont de $1/11$ heure pour les agents

spécialistes et $1/10$ heure pour les autres agents.

Bien entendu, nous ne pouvons pas optimiser ce centre d'appels avec l'algorithme CP-CMTD. L'idée est alors d'exécuter l'algorithme CP en deux étapes. Débuter avec l'algorithme CP-CMTD, plus rapide, pour optimiser la version markovienne du problème, puis de continuer avec l'algorithme CP-ED, plus précis, lorsque la solution se rapproche de la réalisabilité. Nous définissons les paramètres t_1 et t_2 tels que l'algorithme remplace le simulateur CMTD par le simulateur ED à la première itération à laquelle $g(\bar{\mathbf{y}}) + t_1 \geq l$ et $g_k(\bar{\mathbf{y}}) + t_2 \geq l_k$ pour tout k, où $\bar{\mathbf{y}}$ est la solution courante. Plus que t_1 et t_2 sont grands, plus que la première étape avec CP-CMTD sera courte. Nous appelons cette méthode CP-CMTD-ED. Les simulations ont beaucoup plus de variance à cause des processus d'arrivée doublement stochastiques. Ainsi, nous augmentons le nombre de réplications à 50 pour le simulateur CMTD et à 25 pour le simulateur ED. Nous augmentons également le temps d'optimisation à 3 minutes, mais ce n'est pas une limite dure.

La solution optimale empirique est $\mathbf{y}^* = (39, 39, 29, 3, 3, 3)^{\mathrm{T}}$ avec un coût total de 116.45. L'exemple non markovien requiert 6 agents supplémentaires et coûte 5.85 de plus que la solution optimale du modèle markovien. Il est intéressant de constater que le nombre d'agents avec multiples habiletés n'a pas augmenté dans la solution optimale.

Le tableau 4.V compare les performances de CP-CMTD-ED et CP-ED basées sur 100 exécutions indépendantes. La réalisabilité des solutions est vérifiée avec 2000 réplications du simulateur ED et une tolérance $\varepsilon = 0.1$. Nous indiquons si nous exécutons la recherche locale de Avramidis et al. [18] (ACL) ou sinon, la recherche primitive de Ceẓik et L'Ecuyer [37]. Parmi les F solutions réalisables, nous rapportons le coût minimal, le coût médian, le nombre de solutions ayant un coût au plus $1 + 0.01z$ fois supérieur au coût optimal dans la colonne O_z et le temps d'exécution moyen.

Les algorithmes CP-ED et CP-CMTD-ED sont tous les deux capables de trouver de bonnes solutions réalisables, mais CP-CMTD-ED en trouve plus souvent. Les deux algorithmes obtiennent plus souvent de solutions réalisables lorsque nous utilisons la recherche locale ACL, mais cette recherche demande généralement plus de temps d'exécution. Les choix des paramètres t_1 et t_2 ne semblent pas donner de différences significatives sur les solutions finales.

Algorithme	t_1	t_2	ACL	Min	Méd	F	$O_{0.2}$	$O_{0.5}$	O_1	Temps
CP-ED	-	-	Oui	116.50	118.40	76	2	6	16	259s
CP-CMTD-ED	0.02	1	Oui	116.55	118.00	77	6	9	31	195s
CP-CMTD-ED	0.1	1	Oui	116.45	118.05	79	5	8	25	192s
CP-CMTD-ED	0.1	0.1	Oui	116.60	118.25	82	2	10	24	276s
CP-ED	-	-	Non	116.95	119.35	43	0	2	4	210s
CP-CMTD-ED	0.02	1	Non	116.80	118.25	43	0	3	14	238s
CP-CMTD-ED	0.1	1	Non	116.55	118.28	48	2	7	14	121s
CP-CMTD-ED	0.1	0.1	Non	116.70	118.40	47	0	7	16	122s

Tableau 4.V – Exemple 2 non markovien avec $\lambda T = 14\,300$ et $\varepsilon = 0.1$: comparaison des algorithmes CP-CMTD-ED et CP-ED, basée sur 100 exécutions.

En résumé, nous constatons qu'il est avantageux d'utiliser un simulateur CMTD même pour optimiser un modèle non markovien. L'algorithme CP-CMTD-ED termine plus rapidement que la version CP-ED, et ceci permet d'exécuter une meilleure recherche locale (avec de plus longues simulations).

4.4.4 Méthodes de synchronisation du simulateur CMTD durant l'optimisation

Nous expérimentons quatre façons pour contrôler le niveau de synchronisation du simulateur CMTD lors de l'optimisation de l'exemple markovien présenté à la section 4.4.2 :

1. VAI : un taux maximal $\hat{q}$ différent à chaque simulation,
2. VAC-global : $\hat{q}$ constant pour toutes les simulations,
3. VAC-SG : $\hat{q}$ constant seulement pendant l'estimation d'un sous-gradient,
4. VAC-B(β) : $\hat{q}$ constant pour plusieurs estimations consécutives de sous-gradients, avec paramètre de contrôle β.

Nous gardons la même capacité de la file d'attente $H = 100$ et nous choisissons le vecteur $\hat{\mathbf{y}} = (100, \ldots, 100)$ pour VAC-global. La méthode VAC-B, présentée à la section 4.3.2, page 85, est une variante intermédiaire entre VAC-global et VAC-SG. Plus le paramètre β est grand, plus VAC-B devient similaire à VAC-global. Notons que le test d'optimisation à la section 4.4.2 utilisait la méthode VAC-SG.

Le tableau 4.VI compare les résultats sur 100 exécutions de l'algorithme CP-CMTD avec chaque méthode. Dans ces expériences, nous effectuons la méthode de recherche lo-

cale plus rudimentaire et courte de Ceẓik et L'Ecuyer [37], au lieu de la méthode ACL. Ceci réduit l'influence de la recherche locale sur les résultats. Parmi les F solutions ε-réalisables avec $\varepsilon = 0.1$, le tableau donne le coût minimal, le coût médian et $O_{0.5}$, le nombre de solutions ayant un coût inférieur à 1.005 fois le coût optimal, c'est-à-dire 111.153. Nous rapportons également le ratio moyen de transitions fictives, le nombre moyen de sous-gradients évalués, le nombre moyen de vecteurs $\hat{\mathbf{y}}$ utilisés et le temps moyen d'exécution. Puisque H est constant, la synchronisation du simulateur CMTD est contrôlée par le vecteur $\hat{\mathbf{y}}$.

Méthode	Min	Méd	F	$O_{0.5}$	Trans. fict. (%)	Nombre sous-grad.	Nombre $\hat{\mathbf{y}}$	Temps (sec)
VAI	110.80	112.95	83	3	11.7	11.8	-	66
VAC-global	110.70	111.90	70	9	70.8	10.6	1	118
VAC-SG	110.65	112.40	73	11	15.1	11.0	11.0	62
VAC-B(0)	110.60	111.98	76	17	23.5	10.8	7.0	74
VAC-B(0.1)	110.70	111.93	76	18	26.9	11.1	6.1	76
VAC-B(0.2)	110.70	112.18	80	14	29.3	10.4	5.3	76
VAC-B(0.5)	110.60	111.80	81	19	35.4	10.4	4.8	86
VAC-B(2)	110.80	112.25	66	17	53.7	10.7	3.7	133

Tableau 4.VI – Exemple 2 avec $\lambda T = 14\ 300$ et $\varepsilon = 0.1$: comparaison de l'algorithme CP-CMTD avec différentes méthodes pour synchroniser les simulations CMTD, basée sur 100 exécutions.

Les méthodes VAI et VAC-SG sont les plus rapides et génèrent les moins de transitions fictives. VAC-global est parmi les plus lentes et génère le plus de transitions fictives, mais VAC-B peut être encore plus lente si β est élevé. Même si VAI est plus rapide, elle a évalué en moyenne le plus grand nombre de sous-gradients, donc plus d'itérations. Toutes les méthodes sont capables de trouver de bonnes solutions réalisables, mais VAI en trouve le moins souvent. Les méthodes VAC-B trouvent presque 2 fois plus de bonnes solutions que VAC-global et VAC-SG. Par contre, VAC-SG est 20% plus rapide que VAC-B. En somme, les résultats montrent que VAC-B avec un petit β serait un bon choix.

La méthode VAC-SG doit se synchroniser à chaque évaluation d'un sous-gradient, alors que les fréquences de synchronisation avec VAC-B se réduisent en augmentant le paramètre β. Comme mentionné à la section 4.3.2, page 85, la méthode VAC-B(0) n'est pas équiva-

lente à VAC-SG. Nous observons un plus grand ratio de transitions fictives et moins de changement du vecteur $\hat{\mathbf{y}}$ avec VAC-B(0). Notons que l'algorithme CP-CMTD a tendance à nécessiter (légèrement) moins d'itérations quand la synchronisation dure plus longtemps (c'est-à-dire lorsque $\hat{\mathbf{y}}$ change moins souvent).

4.5 Conclusion

Dans ce chapitre, nous avons présenté l'optimisation de l'affectation des agents dans un centre d'appels à l'aide d'un simulateur d'une chaîne de Markov en temps discret (CMTD). Un centre d'appels peut être modélisé par une chaîne de Markov en temps continu (CMTC) quand les appels arrivent en suivant des processus de Poisson, et les durées de service et de patience sont des variables exponentielles. Les taux de transition de la CMTC diffèrent d'un état à l'autre, ce qui rend la simulation de la CMTC plus ardue, car il faut une loi de probabilité différente pour chaque état.

À l'aide de l'uniformisation de la CMTC (présentée à la section 4.1.3), des transitions fictives sont ajoutées de manière à ce que le taux de transition total sortant de chaque état soit égal. Le nombre total de transitions de la CMTC peut alors être généré indépendamment, par une loi de Poisson, des transitions réalisées durant la simulation. Les transitions sont générées en simulant la CMTD imbriquée de la CMTC. Une recherche indexée est utilisée dans le but d'accélérer la génération des transitions. La simulation d'une CMTC par une CMTD est présentée à la section 4.2.

À la section 4.3.1, nous avons présenté des stratégies d'utilisation du simulateur CMTD dans l'optimisation des agents dans un centre d'appels. Nous avons choisi l'algorithme de coupes par sous-gradients et la simulation de Ceẓik et L'Ecuyer [37]. Une stratégie importante est la synchronisation des simulations par des variables aléatoires communes (VAC). Les VAC réduisent significativement les bruits de simulation lors des estimations de sous-gradients par différences finies. Étant donné que nous ne connaissons pas les solutions que l'algorithme d'optimisation visitera, il est difficile d'appliquer efficacement les VAC. Il faudrait considérer un grand espace de solutions, mais l'augmentation subséquente des transitions fictives ralentira la simulation.

Les exemples numériques de la section 4.4 ont montré que l'application des VAC uniquement pour l'estimation de chaque sous-gradient offre un bon compromis entre la réduction du bruit et le temps de simulation. Comme prévu, la simulation CMTD est plus rapide que la simulation par événements discrets (ED) pour les centres d'appels basés sur des modèles markoviens. La réduction de temps est plus significative quand les taux d'arrivée sont élevés, car le simulateur ED doit créer beaucoup d'objets (de programmation), alors que le simulateur CMTD n'utilise que des compteurs. Pour un même budget de temps d'optimisation, les solutions trouvées à l'aide du simulateur CMTD sont meilleures, parce que nous pouvions exécuter des simulations plus longues et précises. Pour des problèmes non markoviens, nous avons exécuté les premières itérations de l'optimisation avec le simulateur CMTD, puis nous l'avons remplacé par le simulateur ED vers la fin. Les résultats numériques montrent que cette approche améliore la vitesse d'optimisation.

CHAPITRE 5

PLANIFICATION DES HORAIRES

Dans ce chapitre, nous cherchons à optimiser les quarts de travail des agents sur plusieurs périodes dans un centre d'appels multi-compétences. Nous adaptons les algorithmes de coupes par sous-gradients d'Atlason et al. [12] et de Ceẓik et L'Ecuyer [37], et nous proposons des heuristiques additionnelles. Les deux articles précédents ne résolvent que des versions simplifiées (des sous-problèmes) du problème de planification considéré. Ce chapitre présente les résultats de notre article [17].

5.1 Modèle du problème de planification

Nous considérons un centre d'appels composé de K types d'appels et de I groupes d'agents. Nous cherchons à planifier les agents sur un temps d'horizon divisé en P périodes. Le temps d'horizon peut être par exemple, une journée ou une semaine, et la longueur typique d'une période est souvent de 15 ou 30 minutes. Il y a S configurations de quart de travail possibles, numérotées de 1 à S. Les quarts de travail se distinguent selon les heures de début et de fin de travail, ainsi que les périodes de pause et de repas. Le vecteur $\mathbf{x} = (x_{1,1}, \ldots, x_{1,S}, \ldots, x_{I,1}, \ldots, x_{I,S})^{\mathrm{T}}$ définit le nombre d'agents par groupe par quart de travail. La matrice $\mathbf{A}$ de taille $IP \times IS$ détermine les périodes couvertes par chaque quart

de travail ; une définition plus détaillée se trouve à la section 2.1, page 15. Le vecteur $\mathbf{y} = \mathbf{A}\mathbf{x} = (y_{1,1}, \ldots, y_{1,P}, \ldots, y_{I,1}, \ldots, y_{I,P})^{\mathrm{T}}$ fait le lien entre $\mathbf{x}$ et $\mathbf{A}$, et donne le nombre d'agents par groupe à chaque période. Le vecteur de coût est $\mathbf{c} = (c_{1,1}, \ldots, c_{1,S}, \ldots, c_{I,1}, \ldots, c_{I,S})^{\mathrm{T}}$ où $c_{i,s}$ représente le coût pour assigner un agent du groupe i au quart de travail s. Par généralité, nous supposons que les agents de tous les groupes peuvent être assignés à n'importe quel quart de travail. Toutefois, notre algorithme fonctionne même si certains groupes d'agents sont contraints à n'occuper qu'un sous-ensemble de quarts de travail.

Notre problème de planification est sujet à satisfaire des contraintes sur les SL. Le temps d'attente acceptable τ étant fixe, nous redéfinissons la fonction $f_{\mathrm{S}}(\tau)$ du SL, définie par l'équation (2.1) à la page 20, par la fonction $g(\mathbf{y})$ qui varie en fonction du vecteur d'agents $\mathbf{y}$. Ainsi, le SL du type d'appel k à la période p est :

$$g_{k,p}(\mathbf{y}) = \frac{\mathbb{E}[X_{\mathrm{B},k,p}]}{\mathbb{E}[N_{k,p} - A_{\mathrm{B},k,p}]}, \tag{5.1}$$

où $N_{k,p}$ est le nombre d'appels de type k arrivés durant la période p, $X_{\mathrm{B},k,p}$ représente le nombre d'appels, parmi ces $N_{k,p}$ appels, qui sont servis dans un délai d'attente d'au plus $\tau_{k,p}$, et $A_{\mathrm{B},k,p}$ est le nombre d'abandons ayant attendu au plus $\tau_{k,p}$. Les SL peuvent être agrégés par type ou par période, ou les deux à la fois. Nous représentons ces mesures agrégées par $g_k(\mathbf{y}), g_p(\mathbf{y})$ et $g(\mathbf{y})$ et les temps d'attente acceptables par τ_k, τ_p et τ. La solution doit satisfaire les seuils minimaux des SL : $l_{k,p}, l_k, l_p$ et l.

Comme nous avons mentionné à la section 2.3, page 20, il faut apporter une attention additionnelle lors de la collecte des données statistiques, car il existe différentes façons de comptabiliser un appel dont le passage dans le centre d'appels chevauche sur plusieurs périodes. Dans notre simulateur, nous choisissons de comptabiliser les appels en fonction des périodes d'arrivée. L'avantage est que le nombre de périodes demeure constant à P. Par contre, les mesures de performance, par exemple $g_{k,p}(\mathbf{y})$, ne sont pas toujours disponibles dès la fin de la période p, parce que les appels arrivés durant la période p peuvent abandonner ou être répondus à une période ultérieure $p' > p$. Par ailleurs, certains centres d'appels choisissent de comptabiliser les appels en fonction de la période d'événement. L'avantage est que les mesures de performance sont disponibles dès la fin de chaque période. Il y a

cependant des inconvénients importants : il peut avoir plus de P périodes dans une journée (s'il reste des appels après la fermeture), et les données peuvent sembler incohérentes, par exemple il est possible d'avoir plus d'abandons que d'arrivées : $A_{\mathrm{B},k,p} > N_{k,p}$. De plus, les SL de la première période $g_{k,1}$ risquent d'être plus bas si les durées de service sont longues.

Le problème standard de planification sur plusieurs périodes est :

$$\begin{aligned}
\min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} = \sum_{i=1}^{I}\sum_{s=1}^{S} c_{i,s}x_{i,s} \\
\text{sujet à :} \quad & \mathbf{A}\mathbf{x} \geq \mathbf{y}, \\
& g(\mathbf{y}) \geq l, \\
& g_k(\mathbf{y}) \geq l_k, \qquad k = 1,\ldots,K, \\
& g_p(\mathbf{y}) \geq l_p, \qquad p = 1,\ldots,P, \\
& g_{k,p}(\mathbf{y}) \geq l_{k,p}, \quad k = 1,\ldots,K, p = 1,\ldots,P, \\
& \mathbf{x},\mathbf{y} \geq 0 \text{ et entiers.}
\end{aligned} \tag{SP0}$$

(SP0) est un problème à nombres entiers, non linéaire et stochastique à cause des fonctions g. Ce problème est NP-difficile, en plus des fonctions g qui sont complexes à évaluer. Toutes les méthodes d'optimisation proposées à ce jour pour des centres réalistes sont des heuristiques.

Au lieu de résoudre (SP0), nous suivons l'idée de Bhulai et al. [23], résumée à la section 3.3.2.1, page 47, qui est d'ajouter la possibilité de transférer temporairement des agents ayant plusieurs habiletés vers tout autre groupe dont ils possèdent déjà toutes les habiletés demandées. Autrement dit, un agent du groupe i, avec l'ensemble d'habiletés $\mathcal{S}_i$, peut être transféré vers un groupe j si $\mathcal{S}_j \subset \mathcal{S}_i$. Notons que nous ne permettons pas de transférer un agent vers un groupe avec le même ensemble d'habiletés. Nous définissons la variable de transfert $z_{i,j,p} \geq 0$ pour indiquer le nombre d'agents du groupe i transférés au groupe j durant la période p. Il peut exister énormément de variables de transfert. Si un groupe possède K' habiletés, les agents de ce groupe peuvent être transférés vers un maximum possible de $2^{K'} - 2$ groupes différents (sans compter les groupes ayant des ensembles d'habiletés identiques). Heureusement, il est possible d'éviter d'énumérer toutes ces variables de transfert grâce aux transferts en chaîne. Par exemple, si les groupes i, j, k sont tels que $\mathcal{S}_k \subset \mathcal{S}_j \subset \mathcal{S}_i$,

alors $z_{i,k,p} = 1$ est équivalent à transférer un agent du groupe i vers j, puis du groupe j à k, c'est-à-dire $z_{i,j,p} = z_{j,k,p} = 1$ (en supposant que $z_{i,j,p} = z_{j,k,p} = 0$ originalement). La variable $z_{i,k,p}$ devient donc redondante avec les transferts en chaîne.

Pour minimiser le nombre de variables de transfert $z_{i,j,p}$, nous définissons les ensembles suivants. Nous définissons l'*ensemble des groupes transférables* **vers** le groupe i par $\mathcal{Z}_i^+ = \{j : \mathcal{S}_j \supset \mathcal{S}_i \text{ et } \nexists\ k : \mathcal{S}_j \supset \mathcal{S}_k \supset \mathcal{S}_i\}$. L'ensemble $\mathcal{Z}_i^+$ contient les groupes j dont les ensembles d'habiletés $\mathcal{S}_j$ sont des **sur-ensembles** minimaux propres de $\mathcal{S}_i$. Inversement, l'*ensemble des groupes transférables* **à partir** du groupe i est défini par $\mathcal{Z}_i^- = \{j : \mathcal{S}_j \subset \mathcal{S}_i \text{ et } \nexists\ k : \mathcal{S}_j \subset \mathcal{S}_k \subset \mathcal{S}_i\}$. L'ensemble $\mathcal{Z}_i^-$ contient les groupes j dont les ensembles d'habiletés $\mathcal{S}_j$ sont des **sous-ensembles** maximaux propres de $\mathcal{S}_i$. Prenons par exemple un centre d'appels composé de $K = 3$ types et $I = 4$ groupes ayant les habiletés suivantes : $\mathcal{S}_1 = \{1\}, \mathcal{S}_2 = \{2\}, \mathcal{S}_3 = \{2,3\}$ et $\mathcal{S}_4 = \{1,2,3\}$. Les ensembles $\mathcal{Z}_i^-$ sont : $\mathcal{Z}_1^- = \mathcal{Z}_2^- = \emptyset$, $\mathcal{Z}_3^- = \{1,2\}$ et $\mathcal{Z}_4^- = \{3\}$, puis les ensembles $\mathcal{Z}_i^+$ sont : $\mathcal{Z}_1^+ = \mathcal{Z}_2^+ = \{3\}$, $\mathcal{Z}_3^+ = \{4\}$ et $\mathcal{Z}_4^+ = \emptyset$. La variable $z_{i,j,p}$ existe si $j \in \mathcal{Z}_i^-$ ou $i \in \mathcal{Z}_j^+$. Dans cet exemple, nous réduisons de $5P$ à $3P$ variables de transfert : $z_{4,3,p}, z_{3,2,p}$ et $z_{3,1,p}$.

Le nombre d'agents travaillant dans le groupe i durant la période p est calculé par :

$$y_{i,p} = \sum_{s=1}^{S} \bar{a}_{p,s} x_{i,s} + \sum_{j \in \mathcal{Z}_i^+} z_{j,i,p} - \sum_{j \in \mathcal{Z}_i^-} z_{i,j,p}, \tag{5.2}$$

où $\bar{a}_{p,s} = 1$ si le quart de travail s couvre la période p, sinon $\bar{a}_{p,s} = 0$. Nous représentons l'ensemble d'équations pour les I groupes et P périodes sous la forme matricielle $\mathbf{y} = \mathbf{Ax} + \mathbf{Tz}$, où $\mathbf{T}$ est la matrice de transfert composée des éléments -1, 0 et 1.

En ajoutant les variables de transfert et les contraintes (5.2) à (SP0), nous obtenons le

problème d'optimisation stochastique suivant :

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} = \sum_{i=1}^{I}\sum_{s=1}^{S} c_{i,s}x_{i,s} \\ \text{sujet à :} \quad & \mathbf{A}\mathbf{x}+\mathbf{T}\mathbf{z} \geq \mathbf{y}, \\ & g(\mathbf{y}) \geq l, \\ & g_k(\mathbf{y}) \geq l_k, \qquad k = 1,\ldots,K, \\ & g_p(\mathbf{y}) \geq l_p, \qquad p = 1,\ldots,P, \\ & g_{k,p}(\mathbf{y}) \geq l_{k,p}, \quad k = 1,\ldots,K, p = 1,\ldots,P, \\ & \mathbf{x},\mathbf{y},\mathbf{z} \geq 0 \text{ et entiers.} \end{aligned} \tag{SP1}$$

Comme nous cherchons une solution qui satisfait les SL, il est permis d'avoir plus d'agents que nécessaire, ainsi nous avons $\mathbf{A}\mathbf{x}+\mathbf{T}\mathbf{z} \geq \mathbf{y}$ au lieu d'une égalité stricte. Remarquons que l'ajout des variables $z_{i,j,p}$ n'augmente pas le coût optimal de (SP1) puisque nous obtenons le problème original (SP0) en fixant $\mathbf{z} = 0$. Ces variables permettent plutôt de réduire le coût optimal en rendant le problème plus flexible. Voici un exemple. Notons que l'exemple 1 et les exemples 2 et 3 à la section 5.2 sont tirés de notre article [17].

Exemple 1. Prenons un centre d'appels avec $K = 2$ types, $I = 3$ groupes, $P = 2$ périodes et $S = 1$ quart de travail couvrant les 2 périodes. Les groupes 1 et 2 sont des spécialistes qui ne peuvent que répondre aux types 1 et 2, respectivement, tandis que le groupe 3 contient des généralistes pouvant répondre aux deux types d'appels. Un agent généraliste coûte 1.2 et un agent spécialiste coûte 1. Supposons que le vecteur $\mathbf{y}$ optimal soit : $y_{1,1} = 4, y_{1,2} = 2$ pour le groupe 1, $y_{2,1} = 2, y_{2,2} = 4$ pour le groupe 2 et $y_{3,1} = y_{3,2} = 0$ (aucun généraliste). Sans variable de transfert, la solution $\mathbf{x}$ sera $(4,4,0)$ pour un coût total de 8. Par contre, nous obtenons la solution $\mathbf{x} = (2,2,2)$ pour un coût total de 6.4 avec les variables de transfert $z_{3,1,1} = z_{3,2,2} = 2$.

En pratique, les agents ont tendance à être plus efficaces quand ils se concentrent à servir qu'un petit nombre de types d'appels. Ceci motiverait les transferts temporaires de généralistes aux postes de spécialistes. Notons qu'une politique de routage dynamique pourrait remplacer les variables de transfert, mais ce routage serait plus complexe à implémenter et il faudrait également l'optimiser. Dans nos exemples numériques, nous considérons des

politiques de routage relativement simples avec des paramètres fixés au préalable et qui demeurent inchangés durant l'optimisation des horaires. Idéalement, il faudrait optimiser les agents et le routage en même temps, mais ce problème est encore plus difficile.

5.2 Optimisation à deux étapes (TS)

En pratique, le problème de planification des agents est souvent résolu en deux étapes ou "*two-stage*" (TS). Notez que nous utiliserons le même acronyme TS que dans notre article [17]. Cette méthode est similaire à l'approche SIPP décrite à la section 3.3.1, page 34, où chaque période est considérée indépendante et stationnaire. La première étape consiste à trouver, indépendamment pour chaque période p, les nombres d'agents optimaux $\hat{y}_{1,p}$, $\hat{y}_{2,p}, \ldots, \hat{y}_{I,P}$ sous contrainte de satisfaire les SL :

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{I} c_i y_{i,p} \\ \text{sujet à :} \quad & \tilde{g}_p(y_{1,p}, y_{2,p}, \ldots, y_{I,p}) \geq l_p, \\ & \tilde{g}_{k,p}(y_{1,p}, y_{2,p}, \ldots, y_{I,p}) \geq l_{k,p}, \quad k = 1, \ldots, K, \\ & y_{1,p}, y_{2,p}, \ldots, y_{I,p} \geq 0 \text{ et entiers}, \end{aligned} \tag{SP2a-p}$$

où c_i est le coût d'un agent i pour une seule période, puis $\tilde{g}_p$ et $\tilde{g}_{k,p}$ sont les SL stationnaires de la période p. Puis la deuxième étape consiste à optimiser les quarts de travail afin de couvrir le vecteur d'agents requis $\hat{\mathbf{y}} = (\hat{y}_{1,1}, \ldots, \hat{y}_{1,P}, \ldots, \hat{y}_{I,1}, \ldots, \hat{y}_{I,P})^{\mathrm{T}}$ tout en minimisant les coûts :

$$\begin{aligned} \min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} = \sum_{i=1}^{I} \sum_{s=1}^{S} c_{i,s} x_{i,s} \\ \text{sujet à :} \quad & \mathbf{A}\mathbf{x} + \mathbf{T}\mathbf{z} \geq \hat{\mathbf{y}}, \\ & \mathbf{x}, \mathbf{z} \geq 0 \text{ et entiers}. \end{aligned} \tag{SP2b}$$

Contrairement à SIPP, nous ne pouvons pas optimiser les problèmes (SP2a-p) simplement à l'aide des formules d'Erlang A ou C. Les algorithmes heuristiques disponibles sont les algorithmes basés sur la simulation de Ceẓik et L'Ecuyer [37] ou d'Avramidis et al. [18], ou bien l'algorithme de Pot et al. [109] dans le cas où il n'y a qu'une seule contrainte agrégée sur l_p et aucun par type d'appel $l_{k,p}$.

L'approche TS ne peut pas considérer les contraintes sur g_k et g, c'est-à-dire les SL agrégés sur plusieurs périodes, étant donné que les périodes sont optimisées séparément. De plus, il peut exister un écart important entre les coûts optimaux des problèmes (SP1) et (SP2b), car les $\hat{y}_{i,p}$ sont choisis indépendamment de $\mathbf{x}$ et $\mathbf{A}$ dans la méthode TS. Nous donnons un exemple.

Exemple 2. Considérons un centre d'appels avec $K = 3$ types, $I = 3$ groupes, $S = 1$ quart de travail couvrant toutes les $P = 3$ périodes et les ensembles d'habiletés suivants : $\mathcal{S}_1 = \{1,2\}, \mathcal{S}_2 = \{1,3\}$ et $\mathcal{S}_3 = \{2,3\}$. Le coût par agent est identique pour les 3 groupes. Notons qu'il n'y a aucune variable de transfert, car il n'existe aucune paire i, j telle que $\mathcal{S}_j \subset \mathcal{S}_i$. Supposons les appels arrivent en suivant des processus de Poisson et que le taux d'arrivée total soit 100 appels par minute répartis de la façon suivante : $\lambda_{1,1} = \lambda_{2,1} = 50$ pour la période 1, $\lambda_{1,2} = \lambda_{3,2} = 50$ pour la période 2 et $\lambda_{2,3} = \lambda_{3,3} = 50$ pour la dernière période. Les taux d'arrivée sont nuls pour $\lambda_{1,3}, \lambda_{2,2}$ et $\lambda_{3,1}$. Les durées de service sont des variables exponentielles avec une moyenne d'une minute par appel pour les 3 types. Nous supposons que les durées de patience sont infinies (aucun abandon) et que la contrainte sur le SL agrégé pour chaque période p est $l_p = 80\%$ avec un AWT de 20 secondes. La politique de routage est celle à priorités égales, définie à la section 2.2, page 17, où les appels sont servis par ordre d'arrivée et les agents sont choisis selon la règle du premier agent inactif avant.

Avec l'approche TS, nous pouvons employer la formule d'Erlang C, à cause des conditions simplifiées, pour résoudre les problèmes (SP2a-p) à l'étape 1. La solution obtenue pour $p = 1$ est $\hat{y}_{1,1} = 104$ et $\hat{y}_{2,1} = \hat{y}_{3,1} = 0$, produisant un SL agrégé de 83.4%. Nous trouvons des solutions semblables pour les périodes 2 et 3 : $\hat{y}_{2,2} = \hat{y}_{3,3} = 104$ et le reste à 0. La solution de (SP2b) à l'étape 2 donne $\mathbf{x} = (104, 104, 104)^{\mathrm{T}}$ pour un total de 312 agents, soit 200% plus d'agents que nécessaire.

Nous trouvons toutefois une solution beaucoup moins chère en optimisant directement (SP1) à l'aide de notre algorithme basé sur la simulation et la programmation linéaire, présenté à la prochaine section. La solution obtenue est $\mathbf{x} = (35, 34, 34)^{\mathrm{T}}$ pour un total de 104 agents. À chaque période, il y a 2 groupes qui travaillent comme des agents spécialistes. Par exemple, durant la première période, les agents du groupe 2 ne servent que les appels du

type 1 et les agents du groupe 3 ne servent que les appels du type 2, alors que les agents du groupe 1 servent les deux types. Cette solution est nettement meilleure que les 312 agents obtenus avec l'approche TS.

Exemple 2b. Ajoutons 3 groupes spécialistes à l'exemple 2 avec les habiletés suivantes : $\mathcal{S}_4 = \{1\}, \mathcal{S}_5 = \{2\}$ et $\mathcal{S}_6 = \{3\}$. Chaque agent spécialiste demande un coût de 6, et chaque agent ayant deux habiletés requiert un coût de 7. Les agents spécialistes servent les appels à la même vitesse que les autres agents. Lorsque nous optimisons les périodes séparément avec l'approche TS, nous obtenons les solutions $\hat{y}_{1,1} = 2, \hat{y}_{4,1} = \hat{y}_{5,1} = 52$ pour la période 1, $\hat{y}_{2,2} = 2, \hat{y}_{4,2} = \hat{y}_{6,2} = 52$ pour la période 2, $\hat{y}_{3,3} = 2, \hat{y}_{5,3} = \hat{y}_{6,3} = 52$ pour la période 3, et le reste à 0. Les solutions optimales aux problèmes (SP2a-p) sont clairement moins coûteuses avec l'ajout des agents spécialistes. Chaque période requiert 106 agents et un coût de 638, alors qu'il en coûtait 728 par période dans l'exemple 2.

Sans variable de transfert, la solution à (SP2b) est $\mathbf{x} = (2,2,2,52,52,52)^{\mathrm{T}}$ pour un coût total de 978. Avec les variables de transfert, la solution à (SP2b) est $\mathbf{x} = (2,52,52,0,0,0)^{\mathrm{T}}$ et le coût total se réduit à 742. Même si la solution $\hat{\mathbf{y}}$ demande 104 agents spécialistes par période, il n'y en a aucun dans la solution $\mathbf{x}$. Les agents avec 2 habiletés remplacent les agents spécialistes de la façon suivante. Dans la première période, les 52 agents du groupe 2 travaillent à la place du groupe 4 et les 52 agents du groupe 3 remplacent les agents du groupe 5. Pour la période 2, les 2 agents du groupe 1 et 50 agents du groupe 2 sont transférés au groupe 4, et les 52 agents du groupe 3 remplacent le groupe 6. Pour la période 3, les 2 agents du groupe 1 et 50 agents du groupe 3 sont transférés au groupe 5, et les 52 agents du groupe 2 sont envoyés au groupe 6. L'étape 1 influence tout de même la solution $\mathbf{x}$ puisque le nombre total d'agents par période est de 106 agents, ce qui est déterminé par $\hat{\mathbf{y}}$. Cette solution est sous-optimale puisqu'en optimisant le problème (SP1) nous trouvons la même solution qu'à l'exemple 2 avec 104 agents ayant 2 habiletés et aucun spécialiste, pour un coût total de 728.

Remarquons que même si une solution est optimale pour (SP1), il peut avoir un sur-effectif d'agents selon la flexibilité des quarts de travail disponibles. Dans le cas le plus flexible, toutes les $2^P - 1$ configurations possibles de quarts de travail sont disponibles. Cependant, il est facile de trouver des exemples où le manque de flexibilité des quarts de

travail serait la cause principale d'un sureffectif d'agents. Nous donnons l'exemple suivant.

Exemple 3. Soit un centre d'appels avec un type, un groupe et $P = 10$ périodes. Chaque quart de travail doit couvrir 8 périodes, avec 7 périodes de travail et une période de pause après 3 ou 4 périodes de travail. La figure 5.1 présente les 6 quarts de travail possibles. Les périodes de travail sont colorées en vert, et les périodes de pauses sont colorées en rose et dénotées par la lettre P. Observons que les quarts de travail se chevauchent tous partiellement. Supposons maintenant qu'il nécessite 100 agents par période. Il est alors inévitable d'avoir un sureffectif d'agents à cause des configurations limitées des quarts de travail. Il faut affecter au minimum 200 agents et le nombre total de périodes de travail rémunéré est 1400. Le nombre de périodes de travail payé serait de 1000 s'il n'y avait aucune contrainte sur les quarts de travail. Ainsi, il y a une sur-affectation de 400 périodes de travail.

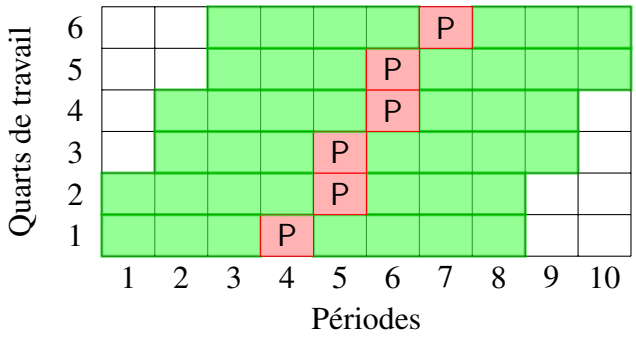


Figure 5.1 – Exemple 3. Configuration de 6 quarts de travail sur 10 périodes. Les périodes de travail sont colorées en vert, alors que les pauses sont colorées en rose et dénotées par la lettre P.

5.3 Optimisation par simulation et programmation linéaire (SCP)

Nous présentons l'algorithme de planification des agents basé sur la simulation et les coupes par sous-gradients. Nous utilisons l'acronyme SCP pour "*scheduling cutting-plane*" afin de différencier de l'algorithme CP pour l'affectation d'une seule période. Cet algorithme est une extension de Ceẓik et L'Ecuyer [37] et d'Atlason et al. [12]. Cette méthode se repose sur l'hypothèse que la fonction de niveau de service $g(\mathbf{y})$ possède une forme

concave lorsque le SL est suffisamment élevé. Jagers et van Doorn [81] ont démontré que la fonction de délai est convexe pour un système $M/M/n$ sans abandon (conditionnellement à couvrir le trafic offert, c'est-à-dire pour un nombre de serveurs $n \geq \lceil \lambda/\mu \rceil$). Leur preuve a pour corollaire que la fonction du SL est concave pour le même système $M/M/n$. Il n'existe cependant aucune preuve de concavité pour les centres d'appels plus complexes ou avec abandons. Pour un système $M/M/n+M$ avec abandons, la courbe du SL a une forme d'un "S" étiré, voir la figure 3.2 à la page 40. La courbe a une forme concave seulement à partir d'un certain nombre de serveurs. Notre algorithme est donc une heuristique comme ses deux algorithmes prédécesseurs.

Nous n'optimisons pas directement le problème (SP1), car nous ne savons pas comment calculer directement (et analytiquement) les fonctions g. Nous pouvons cependant approximer g par des fonctions $\hat{g}$ évaluées à l'aide de la simulation. Nous supposons que les fonctions $\hat{g}$ possèdent les mêmes courbes que g. Les simulations sont synchronisées en utilisant des *variables aléatoires communes* (VAC) [9, 97] telles que le centre d'appels reçoit la même séquence de clients avec les attributs (avec temps d'arrivée, durée de service requise par groupe d'agents et temps de patience) identiques à chaque exécution du simulateur. Nous pourrions générer l'historique et les attributs de tous les clients avant l'optimisation et garder ces informations en mémoire, puisqu'ils sont indépendants des solutions $\mathbf{y}$. Ainsi, nous optimisons un échantillon du problème original (SP1) :

$$
\begin{aligned}
\min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} = \sum_{i=1}^{I}\sum_{s=1}^{S} c_{i,s}x_{i,s} \\
\text{sujet à :} \quad & \mathbf{A}\mathbf{x} + \mathbf{T}\mathbf{z} \geq \mathbf{y}, \\
& \hat{g}(\mathbf{y}) \geq l, \\
& \hat{g}_k(\mathbf{y}) \geq l_k, \qquad k = 1,\ldots,K, \\
& \hat{g}_p(\mathbf{y}) \geq l_p, \qquad p = 1,\ldots,P, \\
& \hat{g}_{k,p}(\mathbf{y}) \geq l_{k,p}, \quad k = 1,\ldots,K, p = 1,\ldots,P, \\
& \mathbf{x},\mathbf{y},\mathbf{z} \geq 0 \text{ et entiers},
\end{aligned}
\tag{SP1$_n$}
$$

où n est la taille de l'échantillonnage (le nombre de jours simulés).

5.3.1 Génération de coupes linéaires basées sur les sous-gradients

Nous décrivons la méthode pour calculer une coupe basée sur le sous-gradient de la fonction $\hat{g}$, mais elle se généralise pour $\hat{g}_k, \hat{g}_p$ et $\hat{g}_{k,p}$. Nous utilisons le terme "sous-gradient", car la fonction $\hat{g}$ est discrète et non différentiable. Soient $\bar{\mathbf{y}}$ la solution courante, $\hat{\mathbf{q}}(\bar{\mathbf{y}})$ un sous-gradient de $\hat{g}$ au point $\bar{\mathbf{y}}$ et l le SL minimal requis. Supposons que la solution courante n'est pas encore réalisable, c'est-à-dire que $\hat{g}(\bar{\mathbf{y}}) < l$. Nous avons par l'hypothèse de concavité de $\hat{g}$:

$$\hat{g}(\bar{\mathbf{y}}) + \hat{\mathbf{q}}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \bar{\mathbf{y}}) \geq \hat{g}(\mathbf{y}).$$

Nous cherchons une solution $\mathbf{y}$ réalisable telle que $\hat{g}(\mathbf{y}) \geq l$. Donc :

$$\hat{g}(\bar{\mathbf{y}}) + \hat{\mathbf{q}}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \bar{\mathbf{y}}) \geq \hat{g}(\mathbf{y}) \geq l,$$

$$\hat{g}(\bar{\mathbf{y}}) + \hat{\mathbf{q}}(\bar{\mathbf{y}})^{\mathrm{T}}(\mathbf{y} - \bar{\mathbf{y}}) \geq l,$$

puis en regroupant les termes connus ensemble, nous obtenons l'équation linéaire suivante :

$$\hat{\mathbf{q}}(\bar{\mathbf{y}})^{\mathrm{T}}\mathbf{y} \geq l - \hat{g}(\bar{\mathbf{y}}) + \hat{\mathbf{q}}(\bar{\mathbf{y}})^{\mathrm{T}}\bar{\mathbf{y}}. \tag{5.3}$$

L'équation (5.3) est une coupe par sous-gradient puisqu'elle n'élimine que les solutions irréalisables (si $\hat{g}$ est concave et non décroissante, et $\hat{\mathbf{q}}$ est vraiment un sous-gradient). Ceci signifie que nous pouvons remplacer les contraintes non linéaires sur $\hat{g}$ par un ensemble d'équations linéaires. Les problèmes linéaires (et continus) sont beaucoup plus faciles à optimiser que les problèmes non linéaires, et il existe plusieurs outils d'optimisation.

Cependant, la difficulté principale constitue à estimer le sous-gradient $\hat{\mathbf{q}}$, puisque nous ne connaissons pas analytiquement les fonctions $\hat{g}$ ou g et encore moins leurs dérivées. De plus, comme la fonction $\hat{g}$ est approximée par une simulation finie, il est probable que $\hat{g}$ ne soit pas concave même si la fonction originale g est concave. Néanmoins, nous suivons Ceşik et L'Ecuyer [37] et nous estimons $\hat{\mathbf{q}}$ par la méthode des différences finies. Le vecteur

$\hat{\mathbf{q}} = (\hat{q}_{1,1}, \ldots, \hat{q}_{1,P}, \ldots, \hat{q}_{I,1}, \ldots, \hat{q}_{I,P})^{\mathrm{T}}$ est défini par :

$$\hat{q}_{i,p} = \frac{\hat{g}(\bar{\mathbf{y}} + d_{i,p}\mathbf{e}_{i,p}) - \hat{g}(\bar{\mathbf{y}})}{d_{i,p}}, \tag{5.4}$$

où $d_{i,p} \geq 1$ et entier est la mesure du pas, et $\mathbf{e}_{i,p}$ est un vecteur unitaire de longueur IP avec l'élément 1 à la position (i, p) et 0 ailleurs. Dans notre implémentation, nous choisissons de prendre le même pas $d = d_{i,p}, \forall i, p$ durant l'estimation d'un sous-gradient. La valeur de d est choisie en fonction du problème à optimiser. Dans nos exemples numériques, les seuils agrégés par type d'appel sont $l = l_p = 0.8$ et, si présents, $l_k = l_{k,p} = 0.5$. Étant donné que les simulations occupent la majorité du temps d'exécution, nous nous limitons à de courtes simulations qui risquent d'avoir beaucoup de bruit. Nous voulons éviter d'utiliser $d = 1$, alors nous prenons $d = 3$ lorsque les SL (agrégés ou non) sont inférieurs à 0.65, autrement $d = 2$.

Soient $\bar{\mathbf{x}}$ et $\bar{\mathbf{z}}$ les autres variables de la solution courante, il est important à noter que le sous-gradient $\hat{\mathbf{q}}$ est évalué au point $\bar{\mathbf{y}}$ et pas selon les nombres d'agents réellement affectés $\mathbf{A}\bar{\mathbf{x}} + \mathbf{T}\bar{\mathbf{z}}$. La raison est que nous voulons déterminer la frontière réelle du domaine réalisable de $g(\mathbf{y})$ indépendamment des quarts de travail. En prenant $\mathbf{A}\bar{\mathbf{x}} + \mathbf{T}\bar{\mathbf{z}}$, il y a un risque de sur-affecter le nombre d'agents et de couper des solutions réalisables. Évidemment, nous simulons le vecteur $\mathbf{A}\bar{\mathbf{x}} + \mathbf{T}\bar{\mathbf{z}}$ (et aussi $\mathbf{y}$) pour vérifier la réalisabilité de la solution. Il se pourrait que les quarts de travail soient tellement inflexibles que le vecteur $\mathbf{A}\bar{\mathbf{x}} + \mathbf{T}\bar{\mathbf{z}}$ soit réalisable même si $\mathbf{y}$ ne l'est pas. Si $\mathbf{y}$ est réalisable, nous nous attendons à ce que $\mathbf{A}\bar{\mathbf{x}} + \mathbf{T}\bar{\mathbf{z}}$ soit réalisable également.

L'estimation des sous-gradients $\hat{\mathbf{q}}$ par simulation occupe la majorité du temps d'exécution (à moins que le problème à nombres entiers soit difficile à résoudre, mais nous relaxons les variables entières dans ce cas). Chaque itération de l'algorithme SCP demande au maximum $IP + 1$ simulations. Nous ajoutons quelques heuristiques pour réduire le temps de simulation.

Un centre d'appels est un système transitoire où les performances d'une période dépend plus ou moins des périodes précédentes. La corrélation entre les périodes est plus forte lorsque les périodes sont courtes. Cependant, nous ignorons l'effet transitoire lorsque

nous générons une coupe par sous-gradient de $\hat{g}_p$ ou $\hat{g}_{k,p}$ pour une période précise. Nous calculons uniquement les éléments $q_{i,p}$ correspondant à la période p et nous fixons pour le reste $q_{i,r} = 0, \forall r \neq p$. Ceci permet réduire considérablement le nombre de simulations. Une autre procédure pour réduire le temps est d'estimer $\hat{\mathbf{q}}$ avec des simulations plus courtes, soit $n_0 = n/10$ réplications (jours) dans notre implémentation. Ainsi, l'algorithme de coupes optimise le problème (SP1$_{n_0}$) au lieu du problème (SP1$_n$). Puis, nous exécutons une recherche locale à la fin avec plus de réplications pour augmenter les chances de trouver une solution réalisable pour (SP1$_n$).

En général, nous considérons également un petit n pour limiter le temps d'exécution et les simulations ont alors du bruit. De multiples exécutions de l'algorithme SCP peuvent retourner des solutions différentes. En pratique, nous conseillons à l'usager d'exécuter quelques fois l'algorithme et de garder la meilleure solution trouvée.

5.3.2 Description de l'algorithme SCP

L'algorithme SCP est itératif et commence avec une solution initiale $\mathbf{y}^{(1)}$ irréalisable. Puis, il retranche les solutions irréalisables du domaine de solutions par des coupes linéaires jusqu'à l'obtention de la première solution réalisable. Étant donné que l'équation (5.3) n'élimine (théoriquement) aucune solution réalisable, incluant la solution optimale, alors la première solution réalisable trouvée est également optimale.

Plus concrètement, nous remplaçons les contraintes sur $\hat{g}$ par un ensemble d'équations linéaires $\mathbf{By} \geq \mathbf{b}$. Soit $\mathbf{B}^{(v)}\mathbf{y} \geq \mathbf{b}^{(v)}$ l'ensemble des contraintes à l'itération v, auquel s'ajoute de nouvelles coupes basées sur les sous-gradients à chaque itération. L'algorithme débute avec l'ensemble $\mathbf{B}^{(1)}\mathbf{y} \geq \mathbf{b}^{(1)}$ vide. Cependant, nous voulons éviter que la solution initiale soit nulle, $\mathbf{y}^{(1)} = (0, \ldots, 0)^{\mathrm{T}}$, car cette solution ne serait pas dans une région concave de $\hat{g}$.

Nous savons que le nombre d'agents doit couvrir au minimum le trafic total offert lorsque les temps de patience sont infinis. Si les clients peuvent abandonner, alors il peut nécessiter moins d'agents. Nous ajoutons les contraintes linéaires supplémentaires (5.5) à (5.7) pour couvrir une proportion $\alpha_{k,p} \geq 0$ du volume d'appels de type k durant la période p :

$$\sum_{i \in \mathcal{T}_k} w_{k,i,p} \mu_{k,i} \geq \alpha_k \lambda_{k,p}, \qquad k = 1, \ldots, K, \tag{5.5}$$

$$\sum_{k \in \mathcal{S}_i} w_{k,i,p} = y_{i,p}, \qquad i = 1, \ldots, I, \tag{5.6}$$

$$w_{k,i,p} \geq 0, \qquad \forall k, \forall i, \forall p. \tag{5.7}$$

La variable $w_{k,i,p}$ représente la répartition (fractionnaire) des agents du groupe i affectés à servir des appels de type k durant la période p. Les ensembles $\mathcal{S}_i$ et $\mathcal{T}_k$ ont été définis à la section 2.1, page 15. Les paramètres $\alpha_{k,p}$ servent à contrôler la solution de départ et à maintenir un minimum de SL. Nous conseillons $\alpha_{k,p} = 1$ lorsqu'il n'y a aucun abandon et $\alpha_{k,p}$ proche de 1 quand il y a des abandons. Il est généralement rapide de tester quelques valeurs de $\alpha_{k,p}$ lors de l'initialisation et de choisir grossièrement la valeur qui procure une solution initiale ayant un SL ni trop proche du seuil l ou de 0. Avec un seuil agrégé $l = 80\%$ dans tous nos exemples numériques (avec abandons), nous utilisons $\alpha_{k,p} = 1$ pour tout k et les solutions de départ ont un SL agrégé $\hat{g}$ situé entre 45% et 60%. Nous espérons par conséquent que les g_k, g_p et $g_{k,p}$ soient suffisamment au-dessus de 0.

Il serait tentant d'optimiser le problème seulement en contrôlant les paramètres $\alpha_{k,p}$ et oublier les coupes basées sur les sous-gradients. Par exemple, incrémenter $\alpha_{k,p}$ tant que la solution est irréalisable. Il n'y aurait qu'une simulation à faire par itération. Cependant, cette approche n'est pas aussi efficace, car elle ne considère pas l'efficacité et l'impact réel de chaque groupe d'agents sur les SL.

Les équations (5.5) à (5.7) sont présentes durant toute l'optimisation. À l'itération v,

notre algorithme résout le problème linéaire suivant :

$$
\begin{aligned}
\min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} = \sum_{i=1}^{I}\sum_{s=1}^{S} c_{i,s}x_{i,s} \\
\text{sujet à :} \quad & \mathbf{A}\mathbf{x} + \mathbf{T}\mathbf{z} \geq \mathbf{y}, \\
& \mathbf{B}^{(v)}\mathbf{y} \geq \mathbf{b}^{(v)}, \\
& \sum_{i\in\mathcal{T}_k} w_{k,i,p}\mu_{k,i} \geq \alpha_k\lambda_{k,p}, \quad \forall k, \forall p, \qquad (\mathrm{SP1}_{n_0}^{(v)}) \\
& \sum_{k\in\mathcal{S}_i} w_{k,i,p} = y_{i,p}, \qquad \forall i, \forall p, \\
& \mathbf{w} \geq 0, \\
& \mathbf{x}, \mathbf{y}, \mathbf{z} \geq 0 \text{ et entiers},
\end{aligned}
$$

où $\mathbf{w}$ représente l'ensemble des variables $w_{k,i,p}$. Nous écrivons l'indice n_0 au lieu de n pour expliciter que l'algorithme résout un problème avec encore plus de bruit.

Tel que présenté dans le problème (SP1), nous considérons 3 types d'agrégation sur les SL $\hat{g}, \hat{g}_p$ et $\hat{g}_k$, et sans agrégation $\hat{g}_{k,p}$. Nous pourrions ajouter une coupe linéaire au système $\mathbf{B}^{(v)}\mathbf{y} \geq \mathbf{b}^{(v)}$ pour chaque contrainte non satisfaite à chaque itération. Mais nous observons que quand le SL agrégé $\hat{g}$ est faible, les autres SL moins agrégés risquent d'avoir plus de bruit et peuvent générer des mauvaises coupes. Le raisonnement est qu'une mesure agrégée a une courbe plus lisse et contient moins de bruits de simulation. Améliorer une mesure individuelle $\hat{g}_{k,p}$ augmentera les mesures agrégées, mais améliorer une mesure agrégée n'augmentera pas forcément tous les $\hat{g}_{k,p}$. Soit $\mathbf{y}^{(v)}$ la solution à $(\mathrm{SP1}_{n_0}^{(v)})$, nous choisissons un compromis en divisant les contraintes en 5 niveaux de priorité :

1. Si $\hat{g}(\mathbf{y}^{(v)}) < 0.65$ (en supposant que $l = 0.8$ dans nos exemples),
2. S'il y a au moins un p tel que $\hat{g}_p(\mathbf{y}^{(v)}) < l_p$,
3. S'il y a au moins un k tel que $\hat{g}_k(\mathbf{y}^{(v)}) < l_k$,
4. S'il y a au moins une paire (p,k) telle que $\hat{g}_{k,p}(\mathbf{y}^{(v)}) < l_{k,p}$,
5. Si $\hat{g}(\mathbf{y}^{(v)}) < l$ et $\hat{g}(\mathbf{y}^{(v)}) \geq 0.65$.

À chaque itération, nous ajoutons uniquement les coupes correspondant au plus niveau de priorité non satisfait (le niveau 1 étant le plus important). Nous commençons donc par

satisfaire **partiellement** le SL agrégé globalement, puis les SL agrégés par période ou par type d'appel, les SL non agrégés, puis finalement nous retournons à satisfaire le SL agrégé globalement. Cette procédure est une extension d'une heuristique de Ceẓik et L'Ecuyer [37].

5.3.3 Relaxation des variables entières et méthodes d'arrondissement

Il est possible d'optimiser les horaires de centres d'appels de taille moyenne, avec une dizaine de types et de groupes, et quelques centaines de quarts de travail (possiblement plus si nous implémentons la génération de colonnes), par la programmation linéaire à nombres entiers. Lorsque le temps d'exécution est important, il n'est pas nécessaire de résoudre optimalement le problème linéaire entier pour avoir une bonne solution. Il est connu que les méthodes d'optimisation à nombres entiers passe la majeure partie du temps à vérifier qu'une solution est optimale. Ainsi, il est possible de limiter l'optimisation du problème $(\mathrm{SP1}_{n_0}^{(v)})$ à quelques minutes et d'avoir tout de même une bonne solution entière. Nous pouvons aussi résoudre partiellement $(\mathrm{SP1}_{n_0}^{(v)})$ en arrêtant l'optimisation à 1% d'écart d'optimalité par exemple. Par contre, pour les grands centres d'appels, il faut relaxer les variables $\mathbf{x}, \mathbf{y}$ et $\mathbf{z}$ en variables continues, car l'espace de solutions est tout simplement trop vaste. En particulier lorsque plusieurs variables ont de petites valeurs, comme dans nos exemples, il peut avoir un grand écart de coûts entre la solution continue et la solution arrondie. Il est alors important d'utiliser une bonne méthode pour arrondir les solutions.

Relaxons les variables entières $\mathbf{x}, \mathbf{y}$ et $\mathbf{z}$ dans $(\mathrm{SP1}_{n_0}^{(v)})$ et appelons ce problème relaxé $(\mathrm{SP1r}_{n_0}^{(v)})$. Optimiser le problème relaxé est facile et rapide, mais la difficulté principale consiste à retrouver une bonne solution entière. Puisque le nombre d'agents dans un centre d'appels (et le simulateur) doit être entier, il est nécessaire d'arrondir les solutions avant d'exécuter les simulations. Soient $\mathbf{x}^{(v)}, \mathbf{y}^{(v)}$ et $\mathbf{z}^{(v)}$ les solutions continues au problème $(\mathrm{SP1r}_{n_0}^{(v)})$ à l'itération v, et $\check{\mathbf{x}}^{(v)}, \check{\mathbf{y}}^{(v)}$ et $\check{\mathbf{z}}^{(v)}$ les solutions arrondies. Idéalement, nous voudrions que la solution $\check{\mathbf{x}}^{(v)}, \check{\mathbf{y}}^{(v)}, \check{\mathbf{z}}^{(v)}$ soit réalisable également pour $(\mathrm{SP1}_{n_0}^{(v)})$, mais ceci n'est pas obligatoire. D'après nos expériences, il vaut mieux éviter de couper des solutions réalisables de $(\mathrm{SP1}_n)$ que de satisfaire tous les problèmes intermédiaires $(\mathrm{SP1}_{n_0}^{(v)})$. Nous avons expérimenté deux stratégies pour arrondir les variables. La méthode retenue dans notre

implémentation est celle à l'aide d'un seuil δ.

5.3.3.1 Méthode par seuil

Choisir un paramètre $\delta \in [0,1]$ et arrondir les variables $\mathbf{x}^{(v)}$ et $\mathbf{y}^{(v)}$ au plafond si la partie fractionnaire est supérieure à δ, sinon arrondir au plancher. Autrement dit, nous avons $\breve{y}^{(v)}_{i,p} = \lceil y^{(v)}_{i,p} \rceil$ si $y^{(v)}_{i,p} - \lfloor y^{(v)}_{i,p} \rfloor > \delta$, sinon $\breve{y}^{(v)}_{i,p} = \lfloor y^{(v)}_{i,p} \rfloor$. Nous arrondissons toujours les variables $\mathbf{z}^{(v)}$ au plancher pour des raisons qui seront données plus loin. Nous utilisons $\delta = 0.5$ dans nos expériences. À la fin de l'algorithme de coupes, lorsque nous trouvons une solution arrondie réalisable, mais avant l'exécution de la recherche locale, nous raffinons le paramètre δ à l'aide d'une recherche binaire sur l'intervalle $[0,1]$ afin de trouver le plus grand δ qui maintient la solution réalisable.

Nous avons expérimenté aussi d'autres choix de δ. Un choix conservateur serait de prendre $\delta = 1$ (ou proche) pour toujours arrondir au plancher, comme pour $\mathbf{z}^{(v)}$. L'algorithme avancera plus lentement, mais ceci augmente le risque que l'algorithme boucle sur la même solution. Il y a aussi plus de risque de générer de mauvaises coupes au début de l'algorithme lorsque le nombre d'agents est petit. Si nous prenons $\delta = 0$ (ou proche), alors toutes les solutions $\mathbf{x}^{(v)}$ et $\mathbf{y}^{(v)}$ seront arrondies au plafond. L'algorithme avancera plus rapidement, mais il risque facilement d'éliminer des bonnes solutions réalisables. Selon nos expériences, un choix autour de 0.5 offre un bon compromis.

Avec cette méthode par seuil, les solutions $\mathbf{x}^{(v)}, \mathbf{y}^{(v)}$ et $\mathbf{z}^{(v)}$ sont arrondies indépendamment les unes des autres, ce qui peut causer des problèmes de réalisabilité (et pas uniquement par rapport aux SL). Par exemple, supposons que le quart de travail 1 couvre la période 1, les agents du groupe 1 peuvent être transférés aux groupes 2 et 3, et que nous trouvons la solution suivante : $x^{(v)}_{1,1} = 1.2$, $z^{(v)}_{1,2,1} = 0.6$, $z^{(v)}_{1,3,1} = 0.6$ et tous les autres $x^{(v)}_{i,s}$ et $z^{(v)}_{i,j,p}$ sont à 0. Si $\delta = 0.5$, la solution arrondie est $\breve{x}^{(v)}_{1,1} = 1$, $\breve{z}^{(v)}_{1,2,1} = 1$ et $\breve{z}^{(v)}_{1,3,1} = 1$. Le groupe 1 n'a qu'un seul agent, mais il doit transférer deux agents aux groupes 2 et 3 ! Cette solution n'est donc pas réalisable. Nous pouvons éviter ce problème en arrondissant $\mathbf{z}^{(v)}$ toujours au plancher. Par contre, il existe des cas où arrondir au plancher n'est pas suffisant. Supposons que le quart de travail 1 couvre la période 1, et que la solution soit $x^{(v)}_{1,1} = x^{(v)}_{2,1} = 1$, $x^{(v)}_{3,1} = x^{(v)}_{4,1} = 0$, $z^{(v)}_{1,3,1} = z^{(v)}_{2,3,1} = 0.9$, $z^{(v)}_{3,4,1} = 1.8$, et les autres $x^{(v)}_{i,s}$ et $z^{(v)}_{i,j,p}$ à

0. En arrondissant au plancher, nous obtenons $\breve{z}^{(v)}_{1,3,1} = \breve{z}^{(v)}_{2,3,1} = 0$ et $\breve{z}^{(v)}_{3,4,1} = 1$. Or le groupe 3 doit transférer un agent au groupe 4, mais il n'y a aucun agent au groupe 3, puisqu'il dépend du transfert des agents des groupes 1 et 2 ! Cette solution est également irréalisable. Nous optons tout de même pour le choix conservateur de toujours arrondir $\breve{z}^{(v)}_{i,j,p}$ au plancher dans notre implémentation. Étant donné que $\breve{z}^{(v)}$ n'est pas assurément réalisable, nous utilisons une autre procédure pour déterminer le nombre réel de transferts d'agents à partir de $\breve{z}^{(v)}$ que nous présenterons plus loin à la section 5.3.3.3, page 117.

Entrées : $\mathbf{x}^{(V)}$, $\mathbf{z}^{(V)}$, n
Sorties : $\breve{\mathbf{x}}^{(V)}$, $\breve{\mathbf{z}}^{(V)}$

$\breve{z}^{(V)}_{i,j,p} \leftarrow \lfloor z^{(V)}_{i,j,p} \rfloor$, pour tout i, j, p
$\delta_{\mathrm{L}} \leftarrow 0$, $\delta_{\mathrm{U}} \leftarrow 1$, $\delta_{\text{Meilleur}} \leftarrow \delta_{\mathrm{L}}$.
tant que $(\delta_{\mathrm{U}} - \delta_{\mathrm{L}}) > 0.01$ **faire**
    $\delta \leftarrow (\delta_{\mathrm{U}} + \delta_{\mathrm{L}})/2$
    $\breve{x}^{(V)}_{i,s} \leftarrow$ ARRONDIR $(x^{(V)}_{i,s}, \delta)$, pour tout i, s
    $\mathbf{y} \leftarrow$ TROUVEREFFECTIF $(\breve{\mathbf{x}}^{(V)}, \breve{\mathbf{z}}^{(V)})$ (Voir l'algorithme 5.2 à la page 118.)
    Simuler la solution $\mathbf{y}$ avec n réplications.
    si $\mathbf{y}$ est réalisable pour (SP1$_n$) **alors**
        $\delta_{\text{Meilleur}} \leftarrow \delta$
        $\delta_{\mathrm{L}} \leftarrow \delta + 0.01$
    sinon
        $\delta_{\mathrm{U}} \leftarrow \delta - 0.01$
    $\breve{x}^{(V)}_{i,s} \leftarrow$ ARRONDIR $(x^{(V)}_{i,s}, \delta_{\text{Meilleur}})$, pour tout i, s

Algorithme 5.1 : RECHERCHEBINAIRE. Soit V la dernière itération de l'algorithme de coupes, effectuer une recherche binaire pour maximiser δ telle que la solution finale demeure réalisable.

Soient V la dernière itération de l'algorithme de coupes et les solutions continues $\mathbf{x}^{(V)}$ et $\mathbf{z}^{(V)}$; nous ignorons $\mathbf{y}^{(V)}$ puisqu'elle n'est plus utile. L'algorithme 5.1 décrit la maximisation du paramètre δ sur l'intervalle $[0,1]$ à l'aide d'une recherche binaire tout en conservant la réalisabilité des solutions finales arrondies $\breve{\mathbf{x}}^{(V)}$ et $\breve{\mathbf{z}}^{(V)}$. Plus que δ est grand, plus de variables seront arrondies au plancher et le coût diminuera. Notons que nous vérifions la réalisabilité avec n réplications au lieu de n_0 afin que la solution soit réalisable pour (SP1$_n$). Nous exécutons la fonction TROUVEREFFECTIF, décrite plus loin dans l'algorithme 5.2,

à la page 118, pour rectifier les variables de transfert et trouver le nombre réel d'agents affectés. La fonction ARRONDIR arrondit les variables $\mathbf{x}$ au plafond si leurs parties fractionnaires sont supérieures à δ, sinon elles sont arrondies au plancher, et les variables $\mathbf{z}$ sont toujours arrondies au plancher. Dans le cas où la recherche binaire ne trouve pas de solution réalisable, alors $\delta_{\text{Meilleur}} = 0$ et la solution $\mathbf{x}^{(V)}$ entièrement arrondie au plafond. Cette situation peut survenir si l'algorithme de coupes se termine à cause que la limite de temps est atteinte et la solution finale n'est pas encore réalisable. Nous optimisons δ jusqu'à une précision de 0.01.

5.3.3.2 Méthode par fixation progressive des variables (FPV)

Au lieu d'arrondir les solutions $\mathbf{x}^{(v)}, \mathbf{y}^{(v)}$ et $\mathbf{z}^{(v)}$ indépendamment les unes des autres (ainsi créer des problèmes de réalisabilité), cette deuxième méthode cherche à fixer progressivement les variables continues à des valeurs entières dans le problème relaxé, jusqu'à ce que toutes les variables soient fixées. Nous ajoutons itérativement de simples contraintes d'égalité dans ($\text{SP1r}_{n_0}^{(v)}$), et nous appelons ce problème ($\text{SP1r}_{n_0}^{(v,r)}$) à l'itération r, avec ($\text{SP1r}_{n_0}^{(v,1)}$) = ($\text{SP1r}_{n_0}^{(v)}$). Il faudra résoudre le problème relaxé plusieurs fois pour chaque itération v. Avec cette méthode, la solution arrondie aura plus de chance de demeurer réalisable pour ($\text{SP1r}_{n_0}^{(v)}$). Cette solution n'est pas assurément réalisable à cause des coefficients -1 dans la matrice de transfert $\mathbf{T}$. Cette stratégie de relaxation et d'arrondissement n'est pas nouvelle et elle est connue dans la littérature. Notons que nous n'avons pas retenu cette méthode dans notre logiciel.

Observons que les variables $\mathbf{y}$ sont indépendantes de $\mathbf{x}$ et $\mathbf{z}$ dans ($\text{SP1r}_{n_0}^{(v)}$) (mais ces deux dernières dépendent de $\mathbf{y}$). Il serait alors logique d'arrondir les solutions $\mathbf{y}^{(v)}$ en premier. Nous commençons par fixer les variables $\mathbf{y}$, puis les $\mathbf{x}$, et finalement les $\mathbf{z}$ car elles ont le moins d'impact. Idéalement, nous voudrions fixer une variable à la fois, mais le nombre de problèmes linéaires à résoudre sera égal au nombre de variables, et le temps d'exécution sera trop long. Nous allons donc fixer plusieurs variables à la fois. Soit $y_{i,p}^{\text{m}} = y_{i,p}^{(v)} - \lfloor y_{i,p}^{(v)} \rfloor$ la partie fractionnaire de $y_{i,p}^{(v)}$. Nous ordonnons les $y_{i,p}^{\text{m}}$ par ordre décroissant, et voici deux possibilités pour fixer les variables à l'itération r de ($\text{SP1r}_{n_0}^{(v,r)}$) :

1. Ajouter $N \geq 1$ contraintes $y_{i,p} = \lceil y^{(v)}_{i,p} \rceil$, correspondant aux N plus grands $y^{\mathrm{m}}_{i,p}$, au problème ($\mathrm{SP1r}^{(v,r)}_{n_0}$). Plus le paramètre N est proche de 1, plus il faudra d'itérations, mais la solution arrondie sera possiblement meilleure.
2. Diviser l'intervalle $[0,1]$ en plusieurs partitions, disons aux points $0.1, 0.2, \ldots, 0.9$. Fixer toutes les variables $y_{i,p} = \lceil y^{(v)}_{i,p} \rceil$ telles que les $y^{\mathrm{m}}_{i,p}$ sont dans le premier intervalle non vide, commençant par $[0.9, 1[$. S'il n'y en a aucune, alors passer à l'intervalle $[0.8, 0.9[$, ainsi de suite.

Nous utilisons les mêmes procédures pour fixer les variables **x**. Sans les variables **z**, nous obtenons assurément une solution entière réalisable pour ($\mathrm{SP1r}^{(v)}_{n_0}$) en arrondissant toujours au plafond. Il arrive souvent vers la fin de la procédure que les variables restantes ont des parties fractionnaires inférieures à 0.5. Arrondir ces variables au plafond peut augmenter significativement le coût.

Malheureusement, avec la présence des variables **z**, la solution arrondie pourrait être non réalisable à cause des coefficients -1 de la matrice **T**. Notons que les variables **z** n'ont aucun coût dans la fonction objectif de ($\mathrm{SP1}^{(v)}_{n_0}$) ; plus précisément, le coût de la solution ne change pas une fois que les variables **x** sont fixées. À la différence de **x** et **y**, nous arrondissons **z** de manière à ce que le nombre d'agents par groupe et période couvre le mieux possible les variables **y** fixées à $\breve{\mathbf{y}}^{(v)}$:

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{I} \sum_{p=1}^{P} h_{i,p} \\
\text{sujet à :} \quad & h_{i,p} \geq \breve{y}^{(v)}_{i,p} - \sum_{s=1}^{S} \bar{a}_{p,s} \breve{x}^{(v)}_{i,s} - \sum_{j \in \mathcal{Z}^+_i} z_{j,i,p} + \sum_{j \in \mathcal{Z}^-_i} z_{i,j,p}, && \forall i, \forall p, \\
& z_{i,j,p} = \breve{z}^{(v)}_{i,j,p}, && \text{pour quelques } (i,j,p), \\
& h_{i,p}, z_{i,j,p} \geq 0, && \forall i, \forall j, \forall p.
\end{aligned}
$$

Nous arrondissons **z** itérativement en fixant $N \geq 1$ variables $z_{i,j,p}$ à la fois en assignant $\breve{z}^{(v)}_{i,j,p} = \lfloor z^{(v)}_{i,j,p} \rfloor$ si la partie fractionnaire est inférieure à 0.5, et $\breve{z}^{(v)}_{i,j,p} = \lceil z^{(v)}_{i,j,p} \rceil$ sinon.

Nous observons dans nos expériences que la méthode FPV a une plus grande chance d'éliminer de bonnes solutions réalisables pour ($\mathrm{SP1}_n$) que la méthode par seuil. La méthode FPV travaille plus fort que la méthode par seuil pour trouver des solutions entières

$\check{\mathbf{x}}^{(v)}, \check{\mathbf{y}}^{(v)}, \check{\mathbf{z}}^{(v)}$ réalisables pour (SP1r$_{n_0}^{(v)}$). Malgré que ceci semble une bonne idée, la méthode FPV a beaucoup plus de chance de sur-affecter les solutions, car les erreurs s'accumulent avec les itérations. Nous choisissons d'implémenter la méthode d'arrondissement par seuil pour ces raisons.

5.3.3.3 Trouver le nombre réel de transfert d'agents

Comme mentionné précédemment, les deux méthodes d'arrondissement ne peuvent pas garantir que les valeurs arrondies $\check{\mathbf{z}}^{(v)}$ mènent à une solution non négative telle que $\mathbf{A}\check{\mathbf{x}}^{(v)} + \mathbf{T}\check{\mathbf{z}}^{(v)} \geq \mathbf{0}$. Au lieu de corriger les valeurs de $\check{\mathbf{x}}^{(v)}$ et $\check{\mathbf{z}}^{(v)}$, nous cherchons ici à déterminer les variables de transfert effectives $0 \leq \dot{z}_{i,j,p}^{(v)} \leq \check{z}_{i,j,p}^{(v)}$ et entières, telles que $\mathbf{A}\check{\mathbf{x}}^{(v)} + \mathbf{T}\dot{\mathbf{z}}^{(v)} = \dot{\mathbf{y}}^{(v)} \geq \mathbf{0}$ où $\dot{\mathbf{y}}^{(v)}$ est l'affectation réelle des agents. Nous simulons $\dot{\mathbf{y}}^{(v)}$ pour vérifier la réalisabilité de la solution, alors que nous estimons les sous-gradients au point $\check{\mathbf{y}}^{(v)}$. Notons que $\dot{\mathbf{y}}^{(v)}$ peut être différent de $\check{\mathbf{y}}^{(v)}$, puisque $\dot{\mathbf{y}}^{(v)}$ dépend des matrices $\mathbf{A}$ et $\mathbf{T}$, mais pas $\check{\mathbf{y}}^{(v)}$.

L'algorithme 5.2 présente une procédure très simple pour déterminer le nombre de transferts possibles $\dot{\mathbf{z}}^{(v)}$ et de trouver le nombre réel d'agents affectés par groupe et période $\dot{\mathbf{y}}^{(v)}$. Remarquons que nous pouvons réécrire l'équation (5.2) avec uniquement les ensembles $\mathcal{Z}_i^-$:

$$y_{i,p} = \sum_{s=1}^{S} \bar{a}_{p,s} x_{i,s} + \sum_{i \in \mathcal{Z}_j^-} z_{j,i,p} - \sum_{j \in \mathcal{Z}_i^-} z_{i,j,p}.$$

Ainsi, nous pouvons ignorer les sous-ensembles $\mathcal{Z}_i^+$ dans notre implémentation. Puisque les $\mathcal{Z}_i^-$ sont définis par des sous-ensembles propres, il doit avoir au moins un $\bar{z}_{i,j,p}$ qui devient 0 à chaque itération de la boucle "tant que", où $\bar{z}_{i,j,p}$ est le nombre d'agents restant à transférer du groupe i à j. Comme il y a au plus $\sum_{i=1}^{I} |\mathcal{Z}_i^-|$ variables $\bar{z}_{i,j,p}$, où l'opérateur $|\cdot|$ dénombre la taille d'un ensemble, et au moins une variable $\bar{z}_{i,j,p}$ est réduite à 0 par itération, alors l'algorithme 5.2 s'exécute dans un temps de $\mathcal{O}(P(\sum_{i=1}^{I} |\mathcal{Z}_i^-|)^2)$. Il pourrait exister des algorithmes plus efficaces. Puisque les $\mathcal{Z}_i^-$ sont définis par des sous-ensembles propres (donc il n'y a aucun circuit ou boucle de transfert), il est possible de trouver un ordonnancement des $\mathcal{Z}_i^-$ tel que l'algorithme s'exécuterait dans un temps de $\mathcal{O}(P\sum_{i=1}^{I} |\mathcal{Z}_i^-|)$.

Entrées : $\check{\mathbf{x}}^{(v)}$, $\check{\mathbf{z}}^{(v)}$ (solutions arrondies)
Sorties : $\dot{\mathbf{y}}^{(v)}$, $\dot{\mathbf{z}}^{(v)}$ (solutions entières effectives)

$\bar{\mathbf{y}} \leftarrow \mathbf{A}\check{\mathbf{x}}^{(v)}$ (vecteur de départ sans transfert)
$\bar{\mathbf{z}} \leftarrow \check{\mathbf{z}}^{(v)}$ (copier le vecteur de transfert)
pour $p = 1$ **à** P **faire**
    trouvé $\leftarrow$ **vrai**
    tant que trouvé = **vrai faire**
        trouvé $\leftarrow$ **faux**
        pour $i = 1$ **à** I **faire**
            pour chaque $j \in \mathcal{Z}_i^-$ **faire**
                si $\bar{y}_{i,p} > 0$ **et** $\bar{z}_{i,j,p} > 0$ **alors**
                    $m \leftarrow \min\{\bar{y}_{i,p}, \bar{z}_{i,j,p}\}$ (le transfert maximal)
                    $\bar{y}_{i,p} \leftarrow \bar{y}_{i,p} - m$
                    $\bar{y}_{j,p} \leftarrow \bar{y}_{j,p} + m$
                    $\bar{z}_{i,j,p} \leftarrow \bar{z}_{i,j,p} - m$
                    trouvé $\leftarrow$ **vrai**
$\dot{\mathbf{y}}^{(v)} \leftarrow \bar{\mathbf{y}}$
$\dot{\mathbf{z}}^{(v)} \leftarrow \check{\mathbf{z}}^{(v)} - \bar{\mathbf{z}}$ (à présent $\bar{\mathbf{z}}$ contient les agents non transférés)

Algorithme 5.2 : TROUVEREFFECTIF. Méthode pour trouver les nombres d'agents $\dot{\mathbf{y}}^{(v)}$ et de transferts $\dot{\mathbf{z}}^{(v)}$ effectifs lorsque $\check{\mathbf{x}}^{(v)}$ et $\check{\mathbf{z}}^{(v)}$ sont des solutions arrondies à l'itération v.

Mais nous n'avons pas exploré davantage cette idée vu que les temps d'exécution de l'algorithme 5.2 sont négligeables dans nos exemples.

Nous avons choisi de transférer le maximum possible d'agents à chaque itération afin d'accélérer l'algorithme 5.2. Cependant, la distribution des transferts pourrait être inéquitable, car les groupes $j \in \mathcal{Z}_i^-$ visités en premier ont plus de chances de recevoir des agents. Pour mieux balancer les transferts d'agents, nous pourrions distribuer les agents à la ronde (ou "*round-robin*") en fixant $m = 1$. Cette version demandera plus d'itérations. Puisque les $\mathcal{Z}_i^-$ sont des sous-ensembles propres (donc aucun circuit), alors chaque agent peut être transféré au plus $I - 1$ fois à travers la chaîne de transferts, à chaque période. Le temps d'exécution avec $m = 1$ serait dans $\mathcal{O}(I\sum_{p=1}^{P}\sum_{i=1}^{I} y_{i,p})$. Notons que nous n'utilisons pas cette version dans nos expériences numériques, parce qu'elle ne donne pas de différence significative dans nos exemples numériques, et pour économiser du temps de calcul.

5.3.3.4 Modéliser y par des variables continues

D'après nos expériences numériques, lorsque nous ne pouvons pas résoudre par la programmation à nombres entiers, il est préférable de modéliser les variables $\mathbf{y}$ par des variables continues et de **ne pas les arrondir**, sauf pour estimer les sous-gradients. Le problème $(\mathrm{SP1}_{n_0}^{(v)})$ devient :

$$
\begin{aligned}
\min \quad & \mathbf{c}^{\mathrm{T}}\mathbf{x} = \sum_{i=1}^{I}\sum_{s=1}^{S} c_{i,s}x_{i,s} \\
\text{sujet à :} \quad & \mathbf{A}\mathbf{x} + \mathbf{T}\mathbf{z} \geq \mathbf{y}, \\
& \mathbf{B}^{(v)}\mathbf{y} \geq \mathbf{b}^{(v)}, \\
& \sum_{i\in\mathcal{T}_k} w_{k,i,p}\mu_{k,i} \geq \alpha_k\lambda_{k,p}, \quad \forall k, \forall p, \qquad (\mathrm{SP1y}_{n_0}^{(v)}) \\
& \sum_{k\in\mathcal{S}_i} w_{k,i,p} = y_{i,p}, \qquad \forall i, \forall p, \\
& \mathbf{w}, \mathbf{y} \geq 0, \\
& \mathbf{x}, \mathbf{z} \geq 0 \text{ et entiers.}
\end{aligned}
$$

Remarquons que le nombre effectif d'agents affectés, défini par $\mathbf{A}\mathbf{x} + \mathbf{T}\mathbf{z}$, demeure entier puisque $\mathbf{A}, \mathbf{x}, \mathbf{T}$ et $\mathbf{z}$ sont entiers. Le vecteur $\mathbf{y}$ n'est pas forcément entier, car $\mathbf{y}$ n'est borné que par $\mathbf{b}^{(v)}$ et 0, et la contrainte d'intégralité de $\mathbf{y}$ a été relaxée.

Cette relaxation de $\mathbf{y}$ se justifie par différentes lacunes des méthodes d'arrondissement par seuil et FPV. Un inconvénient de la méthode par seuil est que les variables sont arrondies bêtement en fonction de δ. Nous avons remarqué que l'algorithme risque de boucler sur la même solution, surtout vers la fin, lorsque la solution est presque réalisable pour $(\mathrm{SP1}_{n_0})$. La cause est qu'en arrondissant bêtement $\mathbf{y}$, il est possible que la solution courante $\mathbf{y}^{(v)}$ ne soit pas coupée. Donc, l'algorithme se retrouve à la même solution à l'itération suivante !

Notons que le sous-gradient $\hat{\mathbf{q}}(\mathbf{y}^{(v)})$, présenté à la section 5.3.1, n'existe pas toujours en général, parce que $\hat{g}$ est une fonction discrète sur le domaine $\mathbb{N}^{IP}$ et la solution $\mathbf{y}^{(v)}$ est continue dans $\mathbb{R}_{+}^{IP}$. Il faut alors remplacer $\hat{\mathbf{q}}(\mathbf{y}^{(v)})$ par le sous-gradient $\hat{\mathbf{q}}(\breve{\mathbf{y}}^{(v)})$ de la solution arrondie $\breve{\mathbf{y}}^{(v)}$. Nous supposons que l'erreur induite par cette substitution est né-

gligeable. En gardant $\mathbf{y}^{(v)}$ continue et si le sous-gradient $\hat{\mathbf{q}}(\check{\mathbf{y}}^{(v)}) > \mathbf{0}$, alors nous sommes assurés que la solution $\mathbf{y}^{(v)}$ sera coupée. Parfois la solution $\mathbf{y}^{(v)}$ n'est coupée que légèrement et les solutions entières $\mathbf{x}^{(v)}, \mathbf{z}^{(v)}$ demeurent inchangées pendant plusieurs itérations, mais l'algorithme finit toujours par avancer à une nouvelle solution.

Avec la méthode FPV, il n'y a pas de risque que l'algorithme boucle. L'inconvénient est plutôt que les erreurs causées par l'arrondissement au plafond s'accumulent tout au long de l'algorithme. La solution finale peut être significativement sous-optimale, en particulier quand le problème contient beaucoup de variables. Puisque le nombre réel d'agents affectés ne dépend que de $\mathbf{x}$ et $\mathbf{z}$, nous nous permettons de relaxer $\mathbf{y}$ en variables continues afin de réduire les erreurs d'arrondissement. Pour estimer le sous-gradient, nous arrondissons $\mathbf{y}^{(v)}$ à l'aide de la méthode par seuil avec $\delta = 0.5$.

Même si l'algorithme de coupes ne boucle pas, il arrive qu'il converge très lentement vers la fin lorsque $(l - \hat{g})$ tend vers 0. Pour éviter ce problème, nous imposons un incrément minimal $\Delta \geq 0$ du SL lors d'une coupe de sous-gradient. Nous remplaçons l'équation (5.3) par :

$$\hat{\mathbf{q}}(\check{\mathbf{y}}^{(v)})^{\mathrm{T}}\mathbf{y} \geq \hat{\mathbf{q}}(\check{\mathbf{y}}^{(v)})^{\mathrm{T}}\mathbf{y}^{(v)} + \max\{l - \hat{g}(\check{\mathbf{y}}^{(v)}), \Delta\}, \tag{5.8}$$

où nous explicitons que les simulations sont exécutées par rapport à la solution arrondie $\check{\mathbf{y}}^{(v)}$ au lieu de la solution continue $\mathbf{y}^{(v)}$. Nous choisissons $\Delta = 0.005$ dans nos expériences numériques.

Une variante à (5.8) serait de supposer que la fonction $\hat{g}$ soit continue et de remplacer $\hat{g}(\check{\mathbf{y}}^{(v)})$ par $\hat{g}(\mathbf{y}^{(v)})$. Nous pourrions "calculer" $\hat{g}(\mathbf{y}^{(v)})$ par une interpolation linéaire sur les solutions utilisées pour estimer le sous-gradient $\hat{\mathbf{q}}(\check{\mathbf{y}}^{(v)})$. Notons que nous n'avons pas essayé cette variante dans nos exemples numériques.

5.3.3.5 Algorithme de coupes par sous-gradients

Nous pouvons maintenant présenter l'algorithme de planification (algorithme 5.3) par coupes basées sur les sous-gradients avec relaxation des variables entières et arrondissement par seuil δ. Nous utilisons l'algorithme 5.2 pour trouver les nombres d'agents et de transferts effectifs. L'algorithme de coupes s'arrête lorsque le nombre d'agents effectifs

$\dot{\mathbf{y}}^{(v)}$ ou la solution arrondie $\check{\mathbf{y}}^{(v)}$ est réalisable. En tout cas, nous terminons par une recherche locale pour corriger et améliorer la solution. L'optimisation binaire du seuil δ est décrite dans l'algorithme 5.1 à la page 114, et la recherche locale est décrite dans l'algorithme 5.4 à la section 5.3.4.

Entrées : Paramètres du centre d'appels et du simulateur.
Sorties : $\mathbf{x}, \mathbf{z}$

$v \leftarrow 0$
Initialiser le problème relaxé (SP1y$_{n_0}^{(1)}$) avec $\mathbf{B}^{(1)} \leftarrow \emptyset$ et $\mathbf{b}^{(1)} \leftarrow \emptyset$.
fini $\leftarrow$ **faux**
tant que fini = **faux faire**
    $v \leftarrow v+1$
    Résoudre (SP1y$_{n_0}^{(v)}$) et obtenir les solutions continues $\mathbf{x}^{(v)}$, $\mathbf{y}^{(v)}$ et $\mathbf{z}^{(v)}$.
    Obtenir les solutions entières $\check{\mathbf{x}}^{(v)}$ et $\check{\mathbf{y}}^{(v)}$ en arrondissant $\mathbf{x}^{(v)}$ et $\mathbf{y}^{(v)}$ avec la méthode par seuil δ.
    Obtenir la solution entière $\check{\mathbf{z}}^{(v)}$ en arrondissant $\mathbf{z}^{(v)}$ au plancher.
    $\dot{\mathbf{y}}^{(v)} \leftarrow$ TROUVEREFFECTIF $(\check{\mathbf{x}}^{(v)}, \check{\mathbf{z}}^{(v)})$ (Trouver le nombre réel d'agents.)
    Simuler $\hat{g}(\check{\mathbf{y}}^{(v)}), \hat{g}_k(\check{\mathbf{y}}^{(v)}), \hat{g}_p(\check{\mathbf{y}}^{(v)})$ et $\hat{g}_{k,p}(\check{\mathbf{y}}^{(v)})$ pour tout k et p.
    Simuler également les SL pour $\dot{\mathbf{y}}^{(v)}$.
    si $\check{\mathbf{y}}^{(v)}$ ou $\dot{\mathbf{y}}^{(v)}$ est réalisable (satisfait les seuils l, l_k, l_p et $l_{k,p}$) **alors**
        fini $\leftarrow$ **vrai**
    si fini = **faux alors**
        Estimer le sous-gradient $\hat{\mathbf{q}}(\check{\mathbf{y}}^{(v)})$.
        Ajouter des coupes linéaires à $\mathbf{B}^{(v+1)}$ et $\mathbf{b}^{(v+1)}$ (voir la section 5.3.2).
$(\mathbf{x}, \mathbf{z}) \leftarrow$ RECHERCHEBINAIRE $(\mathbf{x}^{(v)}, \mathbf{z}^{(v)})$
$\mathbf{x} \leftarrow$ RECHERCHELOCALE $(\mathbf{x}, \mathbf{z})$

Algorithme 5.3 : ALGOPLANIFICATION. Algorithme de planification par coupes de sous-gradients.

5.3.4 Recherche locale

Nous effectuons une recherche locale à la fin de l'algorithme de coupes afin d'améliorer la solution finale. Rappelons que pour réduire le temps d'exécution, nous simulons avec un nombre réduit de réplications $n_0 < n$. La solution finale pourrait être irréalisable lorsque nous simulons avec n réplications. Le but de la recherche locale est de réduire le coût et

d'augmenter les chances d'obtenir une solution réalisable pour (SP1$_n$).

Effectuer la recherche locale directement en simulant n réplications demandera beaucoup trop de temps d'exécution. Nous exécutons la recherche locale de manière itérative en augmentant progressivement le nombre de réplications du simulateur. Soit $n^{(t)}$ le nombre de réplications de simulation à l'itération t de la recherche locale. Nous commençons avec $n^{(1)} = n_2 \geq n_0$ réplications, et nous augmentons $n^{(t)}$ de $n_2/2$ à chaque itération jusqu'à un maximum de $n_1 \leq n$, c'est-à-dire $n^{(t)} = \min\left\{n^{(t-1)} + \frac{n_2}{2}, n_1\right\}$. Nous avons aussi essayé une mise à jour plus rapide par incrément géométrique : $n^{(t)} = \min\left\{\frac{3}{2}n^{(t-1)}, n_1\right\}$. La recherche locale s'arrête lorsque nous complétons l'itération t avec $n^{(t)} = n_1$ ou si la limite de temps allouée est atteinte. Nous arrêtons également la recherche locale lorsque la solution à la fin d'une itération est réalisable pour le problème de départ (SP1$_n$).

Pour économiser du temps, nous nous limitons à optimiser uniquement les variables $\mathbf{x}$, qui sont les plus importantes puisqu'elles seules déterminent les coûts. Supposons que l'algorithme de coupes ait terminé après V itérations et que les solutions entières soient $\mathbf{x}^{(V)}, \mathbf{y}^{(V)}$ et $\mathbf{z}^{(V)}$. Rappelons que dans le cas de la relaxation continue avec arrondissement par seuil, nous maximisons le paramètre $\delta \in [0,1]$ par une recherche binaire avant d'exécuter la recherche locale. Notons que dans notre implémentation, la recherche binaire sur δ est effectuée avec n_1 et pas nécessairement n réplications de simulation. La solution de départ de la recherche locale est $\mathbf{x}^{(V)}$. Nous ignorons la solution $\mathbf{y}^{(V)}$ puisqu'elle n'est plus utile et nous fixons $\mathbf{z} = \mathbf{z}^{(V)}$. Nous optons de ne pas optimiser les variables $\mathbf{z}$, car le potentiel d'amélioration par rapport au temps d'exécution supplémentaire est moins intéressant. Pour chaque solution $\mathbf{x}$, nous appliquons l'algorithme 5.2 afin de corriger et de déterminer les nombres de transferts effectifs $\dot{\mathbf{z}}$ tels que $\mathbf{A}\mathbf{x} + \mathbf{T}\dot{\mathbf{z}} \geq \mathbf{0}$.

L'algorithme 5.4 présente le processus général de la recherche locale. La fonction ARRONDIR arrondit la valeur au plafond si la partie fractionnaire est supérieure à 0.5, sinon au plancher. Chaque itération de la recherche locale comprend trois phases.

Phase 1. L'algorithme 5.5 s'assure que la solution $\mathbf{x}$ est réalisable au problème (SP1$_{n^{(t)}}$). Si la solution n'est pas réalisable, alors il ajoute un agent à la fois. L'ensemble $\mathcal{V}$ contient les indices des contraintes non satisfaites. Nous satisfaisons les contraintes séparément en commençant par celles qui sont les moins agrégées $l_{k,p}$, puis l_k, l_p et l en dernier. (Une

Entrées : $\mathbf{x}^{(V)}, \mathbf{z}^{(V)}, n, n_1, n_2$ (il faut que $n \geq n_1 \geq n_2$)
Sorties : $\mathbf{x}$

$\mathbf{x} \leftarrow \mathbf{x}^{(V)}$, $\mathbf{z} \leftarrow \mathbf{z}^{(V)}$
$t \leftarrow 0$
$p \leftarrow 0$
faire
  $t \leftarrow t+1$
  $n^{(t)} \leftarrow \min\{n_1, \text{ARRONDIR}((1+p)n_2)\}$
  $\mathbf{x} \leftarrow \text{PHASE1}\ (\mathbf{x}, \mathbf{z}, n^{(t)})$
  $\mathbf{x} \leftarrow \text{PHASE2}\ (\mathbf{x}, \mathbf{z}, n^{(t)})$
  $\mathbf{x} \leftarrow \text{PHASE3}\ (\mathbf{x}, \mathbf{z}, n^{(t)})$
  $\mathbf{y} \leftarrow \text{TROUVEREFFECTIF}\ (\mathbf{x}, \mathbf{z})$
  Simuler $\mathbf{y}$ avec n réplications.
  $p \leftarrow p + 0.5$
tant que $n^{(t)} < n_1$ **et** $\mathbf{y}$ est irréalisable pour (SP1_n).

Algorithme 5.4 : RECHERCHELOCALE.

variante est d'inclure toutes les contraintes non satisfaites en même temps.) L'indice $\bullet$ est un métacaractère (ou "*wildcard*") pour indiquer que la contrainte est agrégée par type ou par période. Lorsque nous vérifions si une contrainte est dans $\mathcal{V}$, l'indice $\bullet$ peut prendre n'importe quelle valeur k ou p couverte par la mesure agrégée. La fonction indicatrice $\mathbb{I}$ retourne 1 si son argument est vrai, sinon elle retourne 0.

La fonction $\mathsf{Occupation}(\mathbf{y}, i, p)$ retourne le taux d'occupation, défini par l'équation (2.9) à la page 24, du groupe i à la période p lorsque le centre d'appels est simulé avec le vecteur d'agents $\mathbf{y}$. Au lieu de considérer seulement le taux maximal d'occupation, nous regardons le 10% quantile (l'utilisateur peut changer ce paramètre). Nous choisissons le groupe i^* et le quart de travail s^* à incrémenter de manière différente selon les contraintes insatisfaites. Si seulement la contrainte l agrégée sur les périodes et types est insatisfaite, alors nous choisissons le groupe i^* qui semble le plus occupé sur le plus grand nombre de périodes, et nous sélectionnons le quart s^* couvrant le plus grand nombre de ces périodes. L'élément $\bar{a}(p,s)$, défini à la section 2.1, vaut 1 si le quart de travail s couvre la période p, et vaut 0 autrement. Le processus de sélection est similaire pour les autres contraintes, mais le choix est conditionnel aux types k et périodes p présents dans $\mathcal{V}$. Par exemple, si

la contrainte (k,p) est insatisfaite, alors nous cherchons parmi les groupes i ayant l'habileté à servir le type k, c'est-à-dire $k \in S_i$. Notons que la ligne 15 de l'algorithme 5.5 signifie que la sélection du quart s^* est conditionnelle aux taux d'occupation seulement si toutes les contraintes violées sont agrégées sur les périodes (l et l_k). Autrement, nous savons exactement les périodes à améliorer, et nous ne conditionnons pas le choix sur les taux d'occupation.

L'algorithme implémenté est légèrement plus complexe. Pour ajouter de la diversification, nous considérons les m meilleurs choix de i^* et s^*, et nous sélectionnons i^* et s^* aléatoirement et uniformément parmi les m candidats. Nous utilisons $m = 5$ dans nos expériences.

Phase 2. L'algorithme 5.6 cherche à diminuer le coût en réduisant le nombre d'agents affectés tout en gardant la solution $\mathbf{x}$ réalisable pour $(\mathrm{SP1}_{n^{(t)}})$. Il tentera de réduire toutes les variables $x_{i,s}$ ayant au moins un agent. Pour mieux balancer l'algorithme, nous utilisons une sélection à la ronde avec un ordre généré aléatoirement, et nous retirons un agent à la fois. La liste ainsi construite peut être parcourue plusieurs fois. Nous retirons l'élément (i,s) de la liste lorsque la variable $x_{i,s}$ ne peut pas être réduite, parce que soit $x_{i,s} = 0$ ou la solution devient irréalisable. L'algorithme s'arrête lorsque la liste devient vide. Puisqu'il retire un agent à la fois pour toutes les variables $x_{i,s}$, le temps d'exécution est dans $\mathcal{O}(\sum_{i=1}^{I}\sum_{s=1}^{S} x_{i,s})$.

Phase 3. L'algorithme 5.7 réduit le coût total en déplaçant, un agent à la fois, d'un quart de travail (ou groupe) vers un autre quart (ou groupe) de même coût ou moins dispendieux. Contrairement à la phase 2, le nombre total d'agents demeure inchangé durant cette phase. Dans notre article [17], nous implémentons l'algorithme 5.7, qui est une méthode aléatoire très simple, pour sélectionner les groupes et les quarts de travail à échanger. Nous cherchons à déplacer un agent affecté à $x_{i,s} \geq 1$ vers $x_{h,r}$ tel que $c_{i,s} \geq c_{h,r}$. Les indices i,s,h et r sont choisis aléatoirement parmi $\{1,\ldots,I\}$ et $\{1,\ldots,S\}$ jusqu'à ce que ces conditions $x_{i,s} \geq 1$ et $c_{i,s} \geq c_{h,r}$ soient vérifiées. Notons qu'il est possible d'avoir $i = h$ ou $s = r$. La méthode HASARD choisit aléatoirement et uniformément un élément d'un ensemble. Le nombre de combinaisons d'indices est gigantesque, et notre méthode finit toujours par trouver une combinaison valide dans nos exemples (mais pas nécessairement réalisable après simulation). Remarquons que nous déplaçons un agent indépendamment

Entrées : $\mathbf{x}, \mathbf{z}, n^{(t)}$
Sorties : $\mathbf{x}$

$\mathbf{y} \leftarrow$ TROUVEREFFECTIF $(\mathbf{x}, \mathbf{z})$
Simuler $\mathbf{y}$ avec $n^{(t)}$ réplications.
tant que $\mathbf{y}$ est irréalisable pour (SP1$_{n^{(t)}}$) **faire**
  $\mathcal{V} \leftarrow \{(k,p) : \hat{g}_{k,p}(\mathbf{y}) < l_{k,p}\}$
  si $\mathcal{V} = \emptyset$ **alors** $\mathcal{V} \leftarrow \{(k,\bullet) : \hat{g}_k(\mathbf{y}) < l_k\}$

  si $\mathcal{V} = \emptyset$ **alors** $\mathcal{V} \leftarrow \{(\bullet,p) : \hat{g}_p(\mathbf{y}) < l_p\}$

  si $\mathcal{V} = \emptyset$ **alors**
    $o \leftarrow 0.9 \max_{i,p} \{\mathsf{Occupation}(\mathbf{y}, i, p)\}$
    $i^* \leftarrow \arg\max_i \left\{ \sum_{p=1}^{P} \mathbb{I}\Big[\mathsf{Occupation}(\mathbf{y}, i, p) \geq o\Big] \right\}$
  sinon
    $o \leftarrow 0.9 \max_{i,p} \{\mathsf{Occupation}(\mathbf{y}, i, p) : (k,p) \in \mathcal{V} \text{ et } k \in \mathcal{S}_i\}$
    $i^* \leftarrow \arg\max_i \left\{ \sum_{k=1}^{K} \sum_{p=1}^{P} \mathbb{I}\Big[\mathsf{Occupation}(\mathbf{y}, i, p) \geq o : (k,p) \in \mathcal{V} \text{ et } k \in \mathcal{S}_i\Big] \right\}$
  si $\mathcal{V} = \emptyset$ **ou** $(p = \bullet, \forall (k,p) \in \mathcal{V})$ **alors**
    $s^* \leftarrow \arg\max_s \left\{ \sum_{p=1}^{P} \bar{a}(p,s) \mathbb{I}\Big[\mathsf{Occupation}(\mathbf{y}, i^*, p) \geq o\Big] \right\}$
  sinon
    $s^* \leftarrow \arg\max_s \left\{ \sum_{p=1}^{P} \bar{a}(p,s) : (\bullet,p) \in \mathcal{V} \right\}$
  $x_{i^*,s^*} \leftarrow x_{i^*,s^*} + 1$
  $\mathbf{y} \leftarrow$ TROUVEREFFECTIF $(\mathbf{x}, \mathbf{z})$
  Simuler $\mathbf{y}$ avec $n^{(t)}$ réplications.

Algorithme 5.5 : PHASE1. Ajouter des agents jusqu'à ce que la solution soit réalisable pour (SP1$_{n^{(t)}}$).

de ses habiletés, ce qui est différent de l'idée des variables de transfert. Cette étape se termine lorsque le coût ne se réduit pas pour un nombre d'essais consécutifs maxEssais ou si la limite de temps est atteinte. Nous imposons une limite maxEssais = 40 dans nos expériences numériques.

Entrées : $\mathbf{x}, \mathbf{z}, n^{(t)}$
Sorties : $\mathbf{x}$

$\mathcal{C} \leftarrow \emptyset$ (Initialiser la liste de candidats.)
pour *groupe* $i = 1$ **à** I **faire**
  pour *quart* $s = 1$ **à** S **faire**
    si $x_{i,s} \geq 1$ **alors** $\mathcal{C} \leftarrow \mathcal{C} \cup (i,s)$
$\mathcal{L} \leftarrow$ PERMUTER $(\mathcal{C})$ (Créer une liste en permutant aléatoirement $\mathcal{C}$.)
tant que $\mathcal{L} \neq \emptyset$ **faire**
  Sortir l'élément (i,s) à la tête de la liste $\mathcal{L}$.
  $x_{i,s} \leftarrow x_{i,s} - 1$
  $\mathbf{y} \leftarrow$ TROUVEREFFECTIF $(\mathbf{x}, \mathbf{z})$
  Simuler $\mathbf{y}$ avec $n^{(t)}$ réplications.
  si $\mathbf{y}$ est irréalisable pour $(\text{SP1}_{n^{(t)}})$ **alors**
    $x_{i,s} \leftarrow x_{i,s} + 1$.
  sinon
    si $x_{i,s} \geq 1$ **alors**
      $\mathcal{L} \leftarrow \mathcal{L} \cup (i,s)$ (Remettre (i,s) à la fin de la liste.)

Algorithme 5.6 : PHASE2. Réduire itérativement un agent à la fois.

Nous avons également implémenté une version un peu plus sophistiquée de la phase 3 avec l'algorithme 5.8. Au lieu de piger deux variables complètement au hasard parmi IS variables $x_{i,s}$, nous favorisons certaines combinaisons plus que d'autres. À chaque itération, nous choisissons une variable $x_{i,s} \geq 1$, et nous appliquons une de ces trois modifications :

1. Changer seulement son quart de travail de s à r, tel que $c_{i,r} \leq c_{i,s}$.
2. Changer seulement son groupe de i à h, tel que $c_{h,s} \leq c_{i,s}$.
3. Changer son groupe de i à h et son quart de travail de s à r, tels que $c_{h,r} \leq c_{i,s}$.

La méthode de modification est choisie aléatoirement avec les probabilités $q_1, q_2, q_3 \geq 0$, respectivement, et $q_1 + q_2 + q_3 = 1$. Nous utilisons les valeurs $q_1 = q_2 = 0.35$ et $q_3 = 0.3$ dans nos expériences numériques. Notons que s'il n'y a qu'un seul groupe ($I = 1$), alors nous ajustons $q_1 = 0$ et nous ajustons $q_2 = q_2/(q_2 + q_3)$ et $q_3 = 1 - q_2$. Nous ajustons de manière similaire lorsqu'il n'y qu'un seul quart de travail ($S = 1$) en fixant cette fois-ci $q_2 = 0$. La méthode HASARD2 retourne la valeur j avec probabilité q_j.

Lorsque nous choisissons le groupe i, nous nous basons sur l'idée qu'un agent moins occupé est possiblement moins important qu'un agent plus occupé. Remarquons que ceci n'est pas nécessairement vrai, par exemple si l'AWT d'un petit type d'appel est proche de 0 seconde, alors les agents servant ce type devront souvent être inoccupés. La méthode CHOISIRGROUPE choisit au hasard un groupe i en attribuant de plus grandes masses de probabilité aux groupes avec de faibles taux d'occupation. Les taux d'occupation sont triés par ordre croissant, et le premier groupe a $(1+d)$ fois plus de chance d'être choisi que le deuxième groupe, ainsi de suite. Nous utilisons les taux d'occupation agrégés sur toutes les périodes et nous ignorons les groupes j sans agent tels que $\sum_{s=1}^{S} x_{j,s} = 0$. Soit $\hat{d} > 0$ la probabilité de sélectionner le groupe le plus occupé. Nous trouvons $\hat{d}$ en résolvant l'équation $\sum_{j=0}^{\hat{I}-1}(1+d)^j\hat{d} = 1$ par une recherche binaire, où $\hat{I}$ est le nombre de groupes ayant au moins un agent affecté.

Une fois le groupe i choisi, nous sélectionnons le quart de travail s avec la méthode CHOISIRQUART. Similairement à CHOISIRGROUPE, la méthode CHOISIRQUART choisit le quart de travail s en donnant plus de chance aux quarts les plus dispendieux. Les coûts $\{c_{i,j} : x_{i,j} \geq 1\}$ sont triés par ordre décroissant, et le quart le plus coûteux a $(1+d)$ fois plus de chance d'être choisi que le quart suivant, ainsi de suite. Nous utilisons le même paramètre $d = 0.05$ pour les deux méthodes dans nos expériences.

Les méthodes de modification correspondent aux trois méthodes suivantes. DÉPLACERQUART choisit aléatoirement un quart de travail r tel que $c_{i,r} \leq c_{i,s}$. DÉPLACERGROUPE choisit aléatoirement un groupe h tel que $c_{h,s} \leq c_{i,s}$. Finalement, DÉPLACERGROUPEQUART choisit au hasard un groupe h et un quart de travail r tels que $c_{h,r} \leq c_{i,s}$. Dans notre implémentation, nous arrêtons la phase 3 si le coût n'est pas réduit durant maxEssais = 50 itérations consécutives.

Entrées : $\mathbf{x}, \mathbf{z}, n^{(t)}$
Sorties : $\mathbf{x}$

$m \leftarrow 0$
maxEssais $\leftarrow 40$
tant que $m <$ maxEssais **faire**
  faire
    $i \leftarrow$ HASARD $(\{1,2,\ldots,I\})$
    $s \leftarrow$ HASARD $(\{1,2,\ldots,S\})$
    $h \leftarrow$ HASARD $(\{1,2,\ldots,I\})$
    $r \leftarrow$ HASARD $(\{1,2,\ldots,S\})$
  tant que $c_{i,s} < c_{h,r}$ *ou* $x_{i,s} = 0$
  $x_{i,s} = x_{i,s} - 1$
  $x_{h,r} = x_{h,r} + 1$
  $\mathbf{y} \leftarrow$ TROUVEREFFECTIF $(\mathbf{x}, \mathbf{z})$
  Simuler $\mathbf{y}$ avec $n^{(t)}$ réplications.
  si $\mathbf{y}$ est irréalisable pour $(\mathrm{SP1}_{n^{(t)}})$ **alors**
    $x_{i,s} = x_{i,s} + 1$
    $x_{h,r} = x_{h,r} - 1$
    $m \leftarrow m + 1$
  sinon
    $m \leftarrow 0$

Algorithme 5.7 : PHASE3. Échanger les agents de groupes ou de quarts de travail pour réduire le coût tout en gardant le même nombre total d'agents.

Entrées : $\mathbf{x}, \mathbf{z}, n^{(t)}$, q_1, q_2, q_3, d
Sorties : $\mathbf{x}$

$m \leftarrow 0$
maxEssais $\leftarrow 50$
tant que $m <$ maxEssais **faire**
  $\mathbf{y} \leftarrow$ TROUVEREFFECTIF $(\mathbf{x}, \mathbf{z})$
  Simuler $\mathbf{y}$ avec $n^{(t)}$ réplications.
  $i \leftarrow$ CHOISIRGROUPE (Occupation ($\mathbf{y}$), d)
  $s \leftarrow$ CHOISIRQUART (i, d)
  méthode $\leftarrow$ HASARD2 (q_1, q_2, q_3)
  si méthode $= 1$ **alors**
    $r \leftarrow$ DÉPLACERQUART (i, s)
    $h \leftarrow i$
  si méthode $= 2$ **alors**
    $r \leftarrow s$
    $h \leftarrow$ DÉPLACERGROUPE (i, s)
  si méthode $= 3$ **alors**
    $(h, r) \leftarrow$ DÉPLACERGROUPEQUART (i, s)
  $x_{i,s} = x_{i,s} - 1$
  $x_{h,r} = x_{h,r} + 1$
  $\mathbf{y} \leftarrow$ TROUVEREFFECTIF $(\mathbf{x}, \mathbf{z})$
  Simuler $\mathbf{y}$ avec $n^{(t)}$ réplications.
  si $\mathbf{y}$ est irréalisable pour $(\mathrm{SP1}_{n^{(t)}})$ **alors**
    $x_{i,s} = x_{i,s} + 1$
    $x_{h,r} = x_{h,r} - 1$
    $m \leftarrow m + 1$
  sinon
    $m \leftarrow 0$

Algorithme 5.8 : PHASE3B. Algorithme plus sophistiqué pour déplacer un agent dans un autre groupe ou quart de travail.

5.3.5 Réduction du nombre de simulations

Les opérations les plus coûteuses dans l'algorithme SCP sont les appels au simulateur (en supposant que nous appliquons la relaxation continue aux grands problèmes). Pour économiser du temps de simulation, nous avons essayé de garder en mémoire, dans une table de hachage, toutes les simulations exécutées par l'algorithme. Les clés de la table de hachage sont les vecteurs entiers $\mathbf{y}$ simulés, et les valeurs sont les mesures de performance obtenues. Nous gardons aussi en mémoire les simulations exécutées lors de la recherche locale. Cependant, nous utilisons une nouvelle table de hachage lorsque nous changeons le nombre de réplications de simulation.

La probabilité que nous revisitons une solution plus qu'une fois est petite puisque nous utilisons un algorithme de coupes et l'espace de solutions est très vaste. D'après nos expériences numériques sur un exemple avec $K = 5$ types d'appels, $I = 15$ groupes d'agents et $S = 285$ types de quarts de travail, l'utilisation de la table de hachage réduit le nombre de simulations d'environ 1%. La table de hachage n'était quelque peu utile que durant les dernières itérations de l'algorithme de coupes et lors de la recherche locale. Nous concluons qu'il n'est pas vraiment utile de garder les résultats des simulations en mémoire.

5.4 Exemples numériques

Dans la section numérique, nous comparons les performances d'optimisation de l'algorithme de coupes basées sur les sous-gradients (SCP) et la méthode à deux étapes (TS). Nous testons les algorithmes sur différents exemples de centres d'appels, inspirés sur des données réelles.

Voici une description générale du mode d'opération des centres d'appels dans nos exemples. Ces informations s'appliquent sur tous les exemples à moins d'une mention contraire. Un centre d'appels ouvre à 8:00 le matin et ferme à 17:00 le soir. Une journée est divisée en $P = 36$ périodes d'une durée de 15 minutes par période. Chaque quart de travail inclut une pause de 30 minutes pour le dîner (repas du midi), une pause de 15 minutes avant le dîner et une autre pause de 15 minutes après le dîner. La durée totale (incluant les pauses) varie de 6.5 heures (26 périodes) à 9 heures (36 périodes). Un quart de travail

peut alors être défini par 5 attributs : (1) la durée totale du quart, (2) l'heure de début, (3) le temps de travail entre l'heure de début et la première pause, (4) l'heure du dîner et (5) le temps de travail après le dîner et la deuxième pause. Le tableau 5.I décrit les différentes configurations possibles, où le type du quart de travail est déterminé par sa durée totale. Il y a au total 285 différents quarts de travail répartis comme suit : 105 quarts de type 1, 45 quarts de type 6, et 27 quarts pour chacun des 5 autres types.

Type	Durée	Début	Délai pause 1	Heure de dîner	Délai pause 2
1	7:30	8:00 8:00 8:30, 9:00, 9:30	1:30, 1:45, 2:00 1:30, 1:45, 2:00 1:30, 1:45, 2:00	12:00, 12:30 13:00 12:00, 12:30, 13:00	1:30, 1:45, 2:00 1:30, 1:45 1:30, 1:45, 2:00
2	7:45	9:15	1:30, 1:45, 2:00	12:00, 12:30, 13:00	1:30, 1:45, 2:00
3	8:00	9:00	1:30, 1:45, 2:00	12:00, 12:30, 13:00	1:30, 1:45, 2:00
4	8:15	8:45	1:30, 1:45, 2:00	12:00, 12:30, 13:00	1:30, 1:45, 2:00
5	8:30	8:30	1:30, 1:45, 2:00	12:00, 12:30, 13:00	2:00, 2:15, 2:30
6	9:00	8:00	3:00, 3:15, 3:30 3:45, 4:00	12:00, 13:30, 14:00	1:15, 1:30, 1:45
7	6:30	10:00	1:30, 1:45, 2:00	13:00, 13:30, 14:00	0:45, 1:00, 1:15

Tableau 5.I – Description des 285 quarts de travail possibles dans nos exemples. Délai pause 1 indique le temps de travail à effectuer avant le début de la première pause. Délai pause 2 indique le temps de travail après le dîner et avant la deuxième pause.

Les appels de chaque type arrivent en suivant un processus de Poisson stationnaire par période, et ce processus est indépendant des autres types d'appels. Les taux d'arrivée sont inspirés de données de centres d'appels réels, et ils varient entre les périodes et les types d'appels. Comme la quantité de données est trop élevée pour énumérer textuellement tous les taux d'arrivée, nous les présenterons par des figures lors des descriptions des exemples. Les durées de service sont des variables exponentielles avec des taux identiques de $\mu_{k,i} =$ 8 appels par heure pour tous les types d'appels et groupes. Les temps de patience ont une distribution mixte et identique pour tous les types d'appels. La probabilité d'abandon immédiat est $\eta_k = 0.001$ (durée de patience 0), et avec une probabilité de 0.999, la durée de patience est une variable exponentielle de moyenne $\nu_k^{-1} = 6$ minutes, sauf pour le petit exemple où la moyenne de patience est de 3 minutes. Par manque de données, nous avons simplifié les durées de service et de patience, ce qui réduit le réalisme des exemples. Mais

puisque ce sont des paramètres du modèle de simulation, ces simplifications ne devraient pas affecter le fonctionnement des algorithmes. Tous les exemples utilisent la politique de routage par priorités P définie à la section 2.2.2, page 18. Les listes type-vers-groupe ont toutes des priorités différentes, alors que les listes groupe-vers-type ont toutes les priorités égales. C'est-à-dire que lorsqu'un appel arrive, le routeur va chercher le premier agent libre capable de le servir en visitant son ordre de préférence des groupes. Quand un agent se libère, il prend l'appel en attente, dont il est capable de servir, ayant attendu le plus longtemps.

Dans nos exemples, nous considérons seulement des contraintes sur les SL (dans la réalité, il pourrait avoir des contraintes sur les abandons, les temps d'attente moyen, etc.). L'AWT est de 20 secondes pour toutes les contraintes de SL. Pour tous les exemples, nous imposons des contraintes sur les SL agrégés globalement g et par période g_p avec des seuils $l = l_p = 0.8$. Pour quelques exemples, nous ajoutons des contraintes sur les SL agrégés par type d'appels g_k et non agrégés $g_{k,p}$ avec des seuils $l_k = l_{k,p} = 0.5$. En pratique, les contraintes sont souvent imposées sur les SL agrégés, et les contraintes sur les SL non agrégés servent plutôt à éviter d'avoir des écarts trop importants entre les différents types d'appels ou périodes.

Le coût $c_{i,s}$ d'un agent du groupe i affecté au quart s dépend du nombre d'habiletés $|\mathcal{S}_i|$ du groupe et de la durée du quart de travail. Le coût $c_{i,s}$ est déterminé par

$$c_{i,s} = \frac{(1 + (|\mathcal{S}_i| - 1)\bar{c})Q_s}{30}, \tag{5.9}$$

où $\bar{c}$ représente le coût par habileté additionnelle, et Q_s est la longueur du quart s en nombre de périodes. Nous "normalisons" le coût en divisant par 30, ce qui correspond à peu près à une durée moyenne de travail de 7.5 heures.

Nous considérons trois exemples de centres d'appels de tailles différentes : (1) un petit centre avec 2 types et 2 groupes, (2) un centre moyen avec 5 types et 15 groupes, et (3) un grand centre composé de 20 types et 35 groupes. De plus, nous considérons deux versions du centre d'appels moyen : M_1 qui n'inclut que les contraintes l et l_p, et M_2 pour lequel nous ajoutons les contraintes l_k et $l_{k,p}$ à M_1. L'exemple M_2 a donc tous les types de

contraintes sur les SL. Nous expérimentons également deux versions du grand exemple : l'exemple L_{36} opère sur 36 périodes comme les autres exemples plus petits, alors que nous augmentons les heures d'ouverture de l'exemple L_{52} à 52 périodes (de 8:00 à 21:00).

Les algorithmes SCP et TS utilisent la librairie CPLEX version 9.0 pour optimiser les programmes linéaires continus et entiers. Nous ajoutons le suffixe IP ou LP pour signifier si l'algorithme de coupes optimise le problème à nombres entiers (SCP-IP) ou le problème relaxé (SCP-LP). Les simulations sont exécutées grâce à la librairie ContactCenters [31] en Java. Nous comparons les algorithmes SCP et TS en allouant des budgets de temps "identiques". Cependant, au lieu d'imposer une limite de temps dure, nous préférons que les algorithmes puissent terminer afin de comparer leurs solutions finales. L'arrêt d'exécution causé par l'expiration du budget de temps ne s'applique que durant la recherche locale de SCP. Nous contrôlons principalement les temps d'exécution par le nombre de réplications de simulation n. Nous comparons les performances d'optimisation pour différents budgets de temps. Plus le budget alloué est grand, plus nous pouvons utiliser un grand n et la solution a plus de chance d'être réalisable. À cause des bruits de simulation et d'optimisation, nous exécutons r réplications de l'optimisation pour chaque test. Nous exécutons les algorithmes SCP et TS avec diverses valeurs de n, et nous comparons les résultats dont les temps moyens d'exécution sont similaires. Toutes les expériences numériques sont exécutées sur un processeur AMD Opteron 2.0Ghz.

Pour résoudre les P sous-problèmes (SP2a-p) de la méthode TS, nous préférons utiliser l'algorithme de recherche d'Avramidis et al. [18], car les P solutions ont tendance à être plus "proches", ce qui aide la résolution du problème unifié (SP2b). Résoudre indépendamment les P sous-problèmes avec la méthode de coupes linéaires de Ceẓik et L'Ecuyer [37] a l'inconvénient que les solutions risquent d'être plus dispersées entre les périodes. Pour chaque sous-problème (SP2a-p), nous simulons uniquement la période p du centre d'appels à l'état stationnaire avec la méthode des moyennes par lots (ou "*batch means*") [95]. C'est-à-dire que nous estimons les performances du centre d'appels à la période p non pas pour la durée prévue de 15 minutes, mais pour un horizon de temps infini. Chaque simulation exécute 15 ou 30 lots (selon le budget de temps) avec 2 lots additionnels de réchauffement où les données statistiques ne sont pas recueillies. Pour traduire la longueur

de simulation par lots en nombre de réplications n, nous avons la relation :

$$n = n_{\mathrm{b}} L_{\mathrm{b}} / 15, \tag{5.10}$$

où n_{b} est le nombre de lots à simuler, L_{b} est la durée d'un lot en minutes, et 15 est la durée d'une période en minutes. Nous utilisons toujours CPLEX pour résoudre le problème à nombres entiers à l'étape 2, (SP2b). Pour les grands exemples, nous imposons une limite de temps à CPLEX.

Les solutions finales de SCP et TS devraient être réalisables pour le problème (SP1$_n$), à moins d'un arrêt prématuré de la recherche locale. Afin de vérifier leurs réalisabilités pour le problème originale (SP1), nous simulons les solutions finales avec beaucoup plus de précision, soit $n_* = 50\ 000$ réplications (ou jours). Pour chaque exemple, nous présentons les résultats dans un tableau avec les colonnes suivantes :

1. Budget : le temps d'exécution alloué.
2. n : le nombre de réplications pour vérifier la réalisabilité dans SCP, ou l'équivalence pour la simulation par lots dans TS, calculée par (5.10).
3. n_2 : le nombre initial de réplications pour la recherche locale, applicable seulement pour SCP.
4. CPU : le temps d'exécution moyen des r réplications d'optimisation.
5. Min : le coût minimal parmi les solutions finales réalisables avec n_* réplications.
6. Méd : le coût médian parmi les solutions finales réalisables avec n_* réplications.
7. P_1^* : le pourcentage des r solutions qui sont réalisables et à 1% du meilleur coût connu du problème (SP1$_{n_*}$).
8. P^* : le pourcentage des r solutions qui sont réalisables avec n_* réplications.
9. G_{p} : la moyenne des écarts relatifs maximaux sur la violation des contraintes agrégées par période : $\mathbb{E}\left[\max_p \left\{100\dfrac{l_p - g_p(\mathbf{y})}{l_p}, 0\right\}\right]$, moyenne sur les r solutions.
10. $G_{\mathrm{t,p}}$: mesure similaire à G_{p} mais sur les contraintes non agrégées $l_{k,p}$.

5.4.1 Un petit centre d'appels : modèle N

Ce centre d'appels est un modèle N (voir la figure 2.1 à la page 16) avec $K = 2$ types d'appels et $I = 2$ groupes d'agents. Le groupe 1 ne peut servir que les appels de type 1, alors que le groupe 2 est capable de servir les deux types. Lorsqu'un appel de type 1 arrive, il donne priorité à un agent libre du groupe 1. La figure 5.2 présente les taux d'arrivée des types d'appels 1 et 2 pour les 36 périodes. Il y a environ quatre fois plus d'appels de type 1 que de type 2. Les taux d'arrivée (en unité d'heure) varient de 84 à 160 pour le type 1, et de 21 à 40 pour le type 2. Comme chaque période dure 0.25 heure, il faudrait diviser les taux par 4 pour obtenir les taux d'appels par période. Les taux d'arrivée sont les plus bas à l'ouverture de la journée, mais augmentent très rapidement durant les deux premières heures, puis ils baissent graduellement par la suite. Le coût supplémentaire pour la seconde habileté du groupe 2 est $\bar{c} = 0.2$ dans l'équation 5.9. Nous imposons des contraintes sur tous les SL avec les seuils : $l = l_p = 0.8$ et $l_k = l_{k,p} = 0.5$ pour tout k et p. Nous considérons quatre budgets de temps différents : 3, 15, 30 et 60 minutes. Pour chaque test, nous exécutons $r = 32$ réplications des algorithmes d'optimisation.

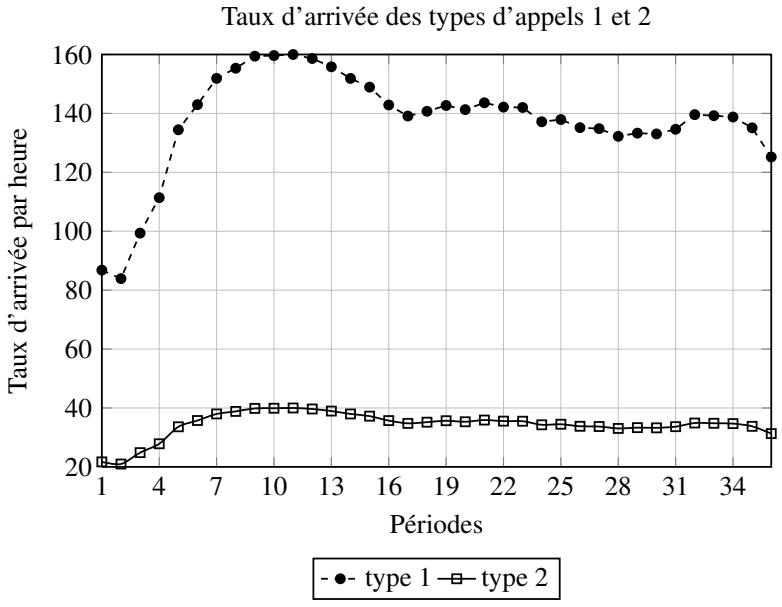


Figure 5.2 – Exemple N : taux d'arrivée (par unité d'heure) des types 1 et 2 pour les 36 périodes. Diviser les taux par 4 pour obtenir les taux par période.

La solution empirique optimale, trouvée parmi toutes les exécutions effectuées par les algorithmes SCP-IP et TS (incluant celles qui ne sont pas montrées dans les résultats), est présentée dans le tableau 5.II. Il y a 23 agents du groupe 1 et 8 agents du groupe 2 pour un total de 31 agents et un coût de 34.873. Dans cet exemple, la mesure P_1^* présente la proportion des solutions réalisables pour ($SP1_{n_*}$) avec des coûts égaux ou inférieurs à 35.222.

Quarts de travail					Groupe 1	Groupe 2
Durée	Début	Délai pause 1	Heure de dîner	Délai pause 2		
7:30	8:00	1:30	12:30	1:45	2	0
7:30	8:00	1:45	13:00	1:30	2	0
7:30	8:00	1:45	12:00	2:00	1	0
7:30	9:30	1:30	12:00	2:00	2	0
7:30	9:30	2:00	12:00	2:00	0	2
7:45	9:15	1:30	12:00	2:00	1	0
7:45	9:15	1:30	12:30	1:45	0	1
8:00	9:00	1:30	12:30	1:45	1	0
8:00	9:00	1:30	13:00	1:30	1	0
8:00	9:00	1:45	13:00	1:30	1	0
8:15	8:45	1:30	13:00	1:30	2	0
8:30	8:30	1:30	12:30	1:45	0	1
8:30	8:30	1:30	12:30	2:00	1	0
9:00	8:00	3:00	14:00	1:45	0	1
9:00	8:00	3:15	13:30	1:15	0	1
9:00	8:00	3:15	14:00	1:15	0	1
9:00	8:00	3:15	14:00	1:30	2	0
9:00	8:00	3:30	13:30	1:30	2	0
9:00	8:00	3:45	13:30	1:15	2	1
6:30	10:00	1:45	14:00	0:45	2	0
6:30	10:00	2:00	14:00	0:45	1	0
Autres					0	0

Tableau 5.II – Exemple N: solution optimale empirique avec 31 agents au total et un coût de 34.873.

Le tableau 5.III compare les résultats pour les différents budgets de temps. Puisque ce problème est petit, nous utilisons l'algorithme SCP-IP pour résoudre avec la programmation en nombres entiers. À l'exception d'un budget de temps très court (3 minutes), l'algorithme SCP-IP trouve des solutions réalisables moins coûteuses beaucoup plus souvent que

l'algorithme TS. Les chances de trouver de bonnes solutions avec SCP-IP augmentent avec le budget de temps et la qualité des simulations. En incrémentant le budget de temps, les colonnes Méd, P_1^* et P^* augmentent tandis que les colonnes G_{p} et $G_{\mathrm{t,p}}$ diminuent. Il est encourageant de voir que la colonne Min ne varie pas beaucoup, ce qui veut dire que SCP-IP est capable de trouver des solutions réalisables proches du coût optimal empirique même lorsque le budget de temps est petit. L'algorithme SCP-IP trouve également beaucoup plus de solutions réalisables que TS. Avec un budget de 60 minutes, 78% des solutions trouvées par SCP-IP sont réalisables contre seulement 12% pour TS. Nous observons que la qualité des solutions de TS ne s'améliore pas significativement quand nous augmentons les budgets d'exécution. TS trouve très peu de solutions réalisables même si les longueurs de simulations sont plus élevées que SCP-IP. Ceci motiverait à ajouter une recherche locale par TS. Pour les deux algorithmes, les solutions irréalisables ont généralement des violations bénignes des contraintes (souvent inférieures à 1%). Étant donné les bruits stochastiques présents dans la réalité, un gestionnaire de centre d'appels pourrait accepter des solutions quasiment réalisables.

Budget (min)	Algo	n	n_2	CPU (sec)	Min	Méd	P_1^*	P^*	G_{p}	$G_{\mathrm{t,p}}$
3	SCP-IP	120	50	191	36.47	37.67	0	34	2.65	3.22
	TS	600		205	35.57	35.57	0	3	1.46	0
15	SCP-IP	1500	800	916	35.13	36.10	9	66	0.36	0.47
	TS	2800		831	35.59	35.64	0	12	0.73	0
30	SCP-IP	2000	1000	1751	35.19	35.99	3	69	0.36	0.07
	TS	5500		1739	35.59	35.67	0	9	0.56	0
60	SCP-IP	3000	1500	3031	35.17	35.77	22	78	0.42	0
	TS	9000		2810	35.51	35.66	0	12	0.58	0

Tableau 5.III – Exemple N : résultats des algorithmes SCP-IP et TS pour divers budgets de temps, basés sur $r = 32$ réplications d'optimisation.

Le tableau 5.IV compare les meilleures solutions trouvées par SCP-IP et TS, parmi toutes les solutions réalisables du tableau 5.III. Les solutions sont présentées sous la forme agrégée par type de quarts de travail, défini dans le tableau 5.I. Les coûts des meilleures solutions sont 35.13 pour SCP-IP, et 35.51 pour TS. Les deux solutions présentent plusieurs

similitudes avec au total 31 agents chacune, mais SCP-IP emploie 23 spécialistes (groupe 1) comparé à 22 pour TS. La solution de SCP-IP est moins coûteuse, car elle emploie un généraliste (groupe 2) de moins et les quarts de travail sont légèrement plus courts. Nous avons expérimenté une variante du modèle N en ajoutant un groupe de spécialistes pour le type d'appel 2. Cette variante s'appelle le modèle M, voir la figure 2.1 à la page 16. La structure de coût de ce nouveau groupe 3 est identique à celle du groupe 1. Le coût optimal empirique de cette variante est 34.45. Le coût est plus bas que le modèle N, parce que la solution emploie maintenant 26 spécialistes (23 agents du groupe 1 et 3 agents du groupe 3) et 5 généralistes pour un total inchangé de 31 agents. Cette expérience montre qu'il est important d'avoir suffisamment de flexibilité sur les ensembles d'habiletés.

La figure 5.3 présente les SL du type 1, du type 2 et agrégé sur les deux types, pour les 36 périodes de la solution SCP-IP du tableau 5.IV. Les seuils des SL agrégés sont $l_p = 0.8$ et de ceux non agrégés sont $l_{k,p} = 0.5$. Ces seuils permettent un certain jeux sur les SL des types 1 et 2 afin de satisfaire les contraintes agrégées. Effectivement, nous observons une grande variation des SL tout au long de la journée. Le type d'appel 1 a des SL constamment meilleurs que ceux du type d'appel 2. Ce déséquilibre s'explique par le fait que les appels du type 1 peuvent être servis par des agents spécialistes moins dispendieux, alors que les appels du type 2 doivent être servis par des généralistes plus coûteux. Les faibles SL du type 2 sont compensés par de hauts SL du type 1 afin d'atteindre les seuils agrégés de 0.8. Ces observations montrent qu'il serait important en pratique d'imposer des minimums de SL non agrégés $l_{k,p}$ même si les contraintes officielles sont sur les performances agrégées (l_k, l_p ou l) afin d'éviter des SL excessivement faibles pour certains types d'appels.

5.4.2 Centre d'appels moyen : 5 types et 15 groupes

Ce centre d'appels possède $K = 5$ types d'appels et $I = 15$ groupes d'agents. Les groupes 1 à 5 sont des spécialistes, avec un groupe pour chaque type d'appel. Les 10 autres groupes sont des agents généralistes pouvant servir entre 2 et 5 types d'appels. Le tableau 5.V présente les ensembles de groupes $\mathcal{T}_k$ pouvant servir chaque type d'appel k. Les appels du type 3 sont plus contraignants, car ils ne peuvent être servis que par 4 groupes d'agents, alors que les appels du type 5 peuvent être répondus par 9 groupes. Les agents du groupe

Algorithme	Groupe d'agents	Type de quarts de travail						
		1	2	3	4	5	6	7
SCP-IP	1	7	1	3	2	1	6	3
	2	2	1	0	0	1	4	0
TS	1	7	1	2	2	1	7	2
	2	1	1	1	0	1	4	1

Tableau 5.IV – Exemple N : meilleures solutions trouvées par SCP-IP et TS, parmi toutes les solutions réalisables du tableau 5.III.

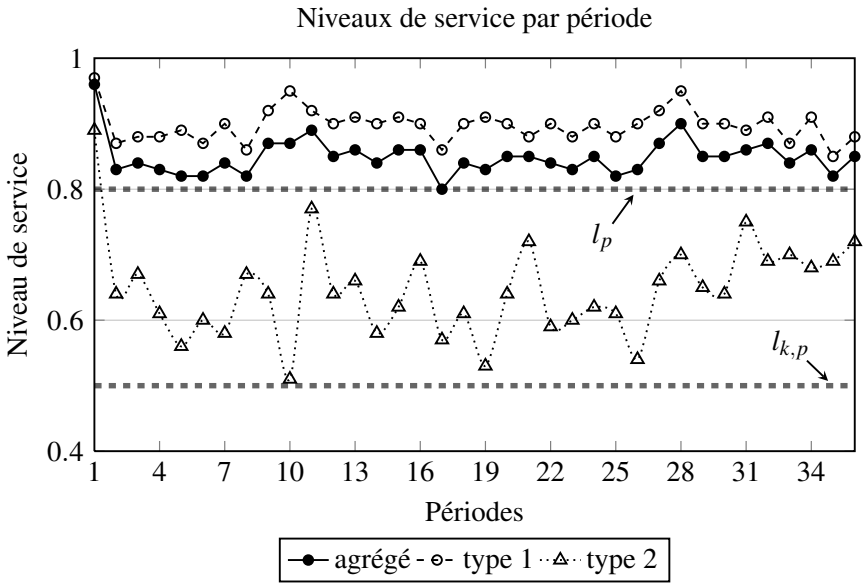


Figure 5.3 – Exemple N : les SL de la meilleure solution trouvée par SCP-IP.

15 peuvent servir tous les appels, et ils sont les plus coûteux par conséquent. Le coût pour chaque habileté additionnelle est $\bar{c} = 0.1$ dans la formule (5.9). Les taux d'arrivée (par unité d'heure) des 5 types d'appels pour les 36 périodes sont présentés dans la figure 5.4. Notons que nous avons ré-numéroté les types d'appels et les groupes d'agents dans cette thèse afin qu'ils soient identiques aux fichiers de paramètres. La numérotation utilisée dans notre article [17] est basée selon l'ordre croissant des taux d'arrivée. Ce changement ne change aucunement la suite des discussions.

En pratique, un gestionnaire de centre d'appels s'intéresse davantage aux SL agrégés. Nous allons de comparer les résultats des variantes M_1 et M_2, où M_1 considère seulement

Type d'appel	Groupes
1	1, 6, 7, 8, 10, 11, 12, 14, 15
2	2, 6, 9, 10, 11, 14, 15
3	3, 7, 9, 10, 12, 13, 14, 15
4	4, 8, 13, 15
5	5, 11, 12, 13, 14, 15

Tableau 5.V – Exemple moyen : les groupes d'agents pouvant répondre à chaque type d'appel.

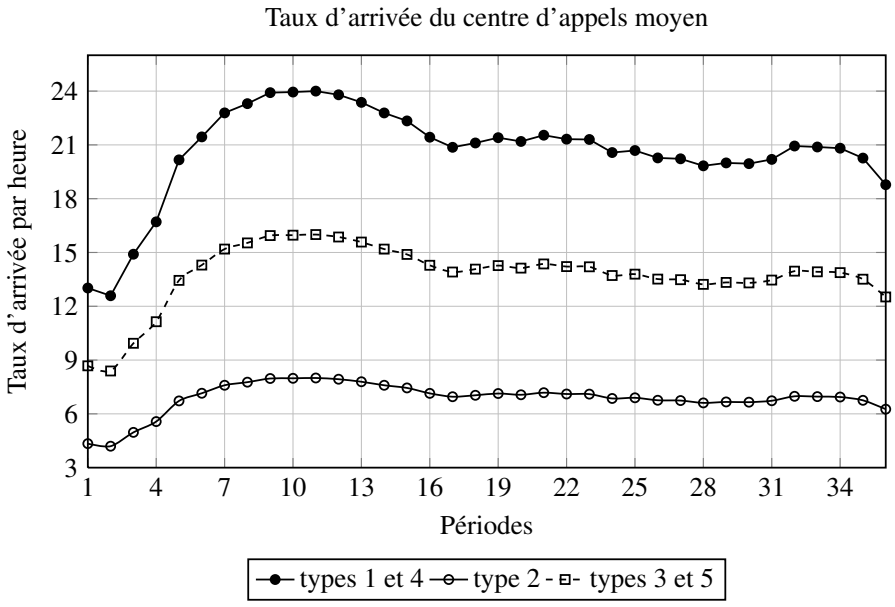


Figure 5.4 – Exemple moyen : taux d'arrivée (par unité d'heure) des 5 types d'appels pour les 36 périodes. Diviser les taux par 4 pour obtenir les taux par période.

les contraintes agrégées $l = l_p = 0.8$, et M_2 ajoute les contraintes additionnelles $l_k = l_{k,p} = 0.5$ à M_1.

Nous expérimentons avec les budgets de temps suivants : 15, 30 et 60 minutes. Pour chaque test, nous exécutons $r = 8$ réplications d'optimisation. Nous ne résolvons pas les problèmes linéaires à nombres entiers jusqu'à l'optimum, car ceci nécessiterait trop de temps. Pour réduire le temps d'exécution, nous essayons de relaxer la condition d'intégralité des variables entières ou de limiter le temps alloué à CPLEX. Nous comparons deux versions de SCP : (1) SCP-LP avec la relaxation continue et (2) SCP-IP qui impose une

limite de temps d'exécution entre 10 et 90 secondes, selon le budget de CPU, pour chaque appel à CPLEX. CPLEX retourne la meilleure solution entière courante lorsque la limite de temps est atteinte. SCP-IP requiert généralement moins d'itérations (donc moins de simulations) que SCP-LP, ce qui compense pour le temps additionnel occupé par CPLEX.

Les tableaux 5.VI et 5.VII présentent les résultats d'optimisation des exemples M_1 et M_2. Le tableau 5.VI n'a pas de colonne $G_{\mathrm{t,p}}$ puisqu'il n'y a pas de contrainte $l_{k,p}$ dans M_1. Les coûts des solutions au problème M_2 sont marginalement supérieurs à ceux de M_1. Les coûts optimaux empiriques (trouvés parmi toutes les exécutions d'optimisation incluant celles non rapportées ici) sont 20.043 pour M_1 et 20.033 pour M_2. Même si ces coûts optimaux sont empiriques, il est surprenant que le coût optimal de M_2 soit comparable à celui de M_1. Cependant, les contraintes additionnelles $l_k = l_{k,p} = 0.5$ dans M_2 sont possiblement trop souples.

Budget (min)	Algo	n	n_2	CPU (sec)	Min	Méd	P_1^*	P^*	G_{p}
15	SCP-IP	200	80	764	20.54	20.99	0	62	0.85
	SCP-LP	200	80	972	20.29	20.96	0	50	2.33
	TS	1000		864	21.47	21.60	0	100	0
30	SCP-IP	600	100	1776	20.18	20.51	12	75	0.88
	SCP-LP	600	100	1947	20.36	20.88	0	62	0.74
	TS	1333		1494	21.43	21.61	0	100	0
60	SCP-IP	1000	400	2981	20.45	20.90	0	62	0.46
	SCP-LP	1000	400	2987	20.51	21.26	0	75	0.25
	TS	4000		3603	21.49	21.57	0	100	0

Tableau 5.VI – M_1 : résultats des algorithmes SCP-IP, SCP-LP et TS pour différents budgets de temps, basés sur $r = 8$ réplications.

Les résultats montrent que les performances de SCP-IP et SCP-LP sont très similaires. SCP-IP est légèrement meilleur que SCP-LP lorsque le budget est grand. Les algorithmes SCP-IP et SCP-LP exécutent plus d'itérations pour optimiser M_2 que M_1, parce que M_2 a beaucoup plus de contraintes à satisfaire. Les solutions réalisables trouvées par SCP-IP et SCP-LP sont environ 5% moins chères que celles trouvées par TS, mais toutes les solutions de TS sont réalisables. La colonne P_1^* montre qu'il est difficile pour les trois algorithmes

Budget (min)	Algo	n	n_2	CPU (sec)	Min	Méd	P_1^*	P^*	G_{p}	$G_{\text{t,p}}$
15	SCP-IP	200	80	664	20.80	21.53	0	37	0.43	0.21
	SCP-LP	200	80	1146	20.57	21.81	0	62	0.98	2.28
	TS	1000		903	21.47	21.59	0	100	0	0
30	SCP-IP	300	80	2188	20.92	21.89	0	62	0.32	0
	SCP-LP	300	80	1827	20.65	21.22	0	62	1.02	0
	TS	1333		1804	21.47	21.62	0	100	0	0
60	SCP-IP	400	100	3106	20.59	21.26	0	62	0.32	3.37
	SCP-LP	400	100	4597	20.71	21.47	0	50	0.17	2.21
	TS	4000		3604	21.53	21.59	0	100	0	0

Tableau 5.VII – M_2 : résultats des algorithmes SCP-IP, SCP-LP et TS pour différents budgets de temps, basés sur $r = 8$ réplications.

de trouver une solution à la fois réalisable et proche du coût optimal empirique. Il est intéressant de voir que les performances de TS ne s'améliorent pas avec l'augmentation du budget de temps.

5.4.3 Grand centre d'appels : 20 types et 35 groupes

Ce grand centre d'appels est composé de $K = 20$ types d'appels et $I = 35$ groupes d'agents. Il y a un groupe de spécialistes par type d'appel et les 15 autres groupes de généralistes peuvent servir entre 4 et 9 types d'appels. Le tableau 5.VIII présente l'ensemble d'habiletés $\mathcal{S}_i$ de chaque groupe d'agents i. Le coût pour chaque habileté additionnelle est $\bar{c} = 0.1$ dans la formule (5.9). Ce centre n'impose que les contraintes agrégées $l = l_p = 0.8$. Ce problème est trop grand pour être optimisé avec SCP-IP (même en limitant le temps occupé par CPLEX), donc nous comparons seulement les algorithmes SCP-LP et TS. Nous testons ce grand exemple avec des budgets de temps de 5 et 10 heures. Pour chaque test, nous exécutons $r = 8$ réplications d'optimisation.

Pour cet exemple, nous voulons expérimenter et comparer les performances de l'algorithme SCP-LP pour deux différentes structures de quarts de travail. La première variante L_{36} est ouverte durant 9 heures par jour et possède les mêmes 285 quarts de travail que les exemples précédents. La deuxième variante L_{52} est ouverte de 8:00 à 21:00 (donc 52

Groupe	Ensemble d'habiletés	Groupe	Ensemble d'habiletés
1	1,2,5,11,16	19	4
2	3,5,9,13,15,17,19	20	5
3	1,6,18,20	21	6
4	3,4,9,12,18	22	7
5	1,5,7,8,9,10,11,13,16	23	8
6	3,8,9,10,15,17,20	24	9
7	1,6,13,15	25	10
8	3,6,11,14,19,20	26	11
9	1,7,10,12,14	27	12
10	3,6,8,9,11,13,16	28	13
11	2,5,12,17	29	14
12	4,7,8,11,16,19,20	30	15
13	2,7,10,14,18,20	31	16
14	4,8,10,12,15,18,20	32	17
15	2,8,12,13,14,19	33	18
16	1	34	19
17	2	35	20
18	3		

Tableau 5.VIII – Grand exemple : ensembles d'habiletés des 35 groupes d'agents.

périodes de 15 minutes) et les quarts de travail sont limités à une durée de 7.5 heures. Ces quarts consistent des quarts de type 1 du tableau 5.I auxquels nous ajoutons la possibilité de débuter un quart à 13:00 ou 13:30 pour couvrir les périodes du soir. Pour limiter le nombre de variables, nous n'offrons pas la possibilité de commencer la journée de travail entre 9:45 et 12:45 (inclusivement). Il y au total $S = 123$ quarts de travail dans L_{52}.

Les taux d'arrivée dans L_{36} et les taux des 36 premières périodes dans L_{52} suivent des courbes similaires aux taux d'arrivée du centre d'appels moyen de la figure 5.4, mais avec différents facteurs de taille. Puis les taux d'arrivée des 16 dernières périodes dans L_{52} diminuent lentement jusqu'à la fermeture du centre. La figure 5.5 donne quelques courbes représentatives des taux d'arrivée (par unité d'heure) de L_{52}. Le type d'appel 18 est le plus petit avec un taux d'arrivée qui varie entre 1.5 et 3 appels à l'heure. Le type d'appel 13 est le plus fréquent avec un taux d'arrivée qui varie entre 14 et 27 appels à l'heure. Il faut diviser ces taux par 4 pour obtenir les taux d'appels par périodes.

Le tableau 5.IX présente les résultats de l'exemple L_{36}. Le coût optimal empirique est

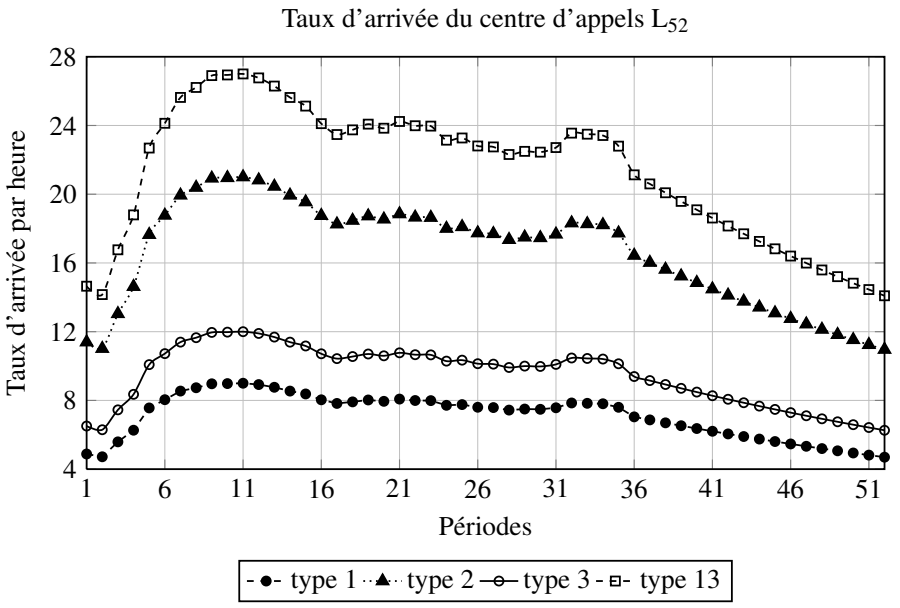


Figure 5.5 – L_{52} : taux d'arrivée (par unité d'heure) des types 1, 2, 3 et 13 pour les 52 périodes. Diviser les taux par 4 pour obtenir les taux par période.

78.38, trouvé par SCP-LP. La simulation occupe quasiment la totalité des temps d'exécution de SCP-LP et TS. Pour ce grand exemple, il faut exécuter au plus $IP + 1 = 1261$ simulations par itération de SCP-LP. Une exécution complète de SCP-LP effectue au total environ 20 000 simulations. Même si les budgets de temps de 5 et 10 heures semblent longs, le temps moyen revient à 0.7 et 1.3 secondes par simulation, ce qui est très court ! SCP-LP a de la difficulté à trouver des solutions réalisables, mais les violations des contraintes sont généralement très faibles. TS trouve toujours des solutions réalisables, cependant ces solutions sont nettement sous-optimales avec des coûts environ 25% plus élevés que les solutions de SCP-LP. Encore une fois, la méthode TS ne semble pas s'améliorer quand nous augmentons le budget de temps et la qualité des simulations. Lorsque nous comparons les meilleures solutions de SCP-LP et TS, nous trouvons que la principale cause de l'écart de coûts est que SCP-LP emploie 10 agents de moins, soit 52 agents contre 62 pour TS.

Le tableau 5.X présente les résultats de l'exemple L_{52}. Le coût optimal empirique est 133.0, trouvé par SCP-LP cette fois-ci aussi. Comme il y a 16 périodes additionnelles, il faut exécuter plus de simulations pour estimer les sous-gradients, soit au plus 1821 si-

Budget (h)	Algo	n	n_2	CPU (min)	Min	Méd	P_1^*	P^*	G_p
5	SCP-LP	400	50	261	79.17	80.05	0	50	0.77
	TS	4000		268	95.99	100.46	0	100	0
10	SCP-LP	500	80	506	78.38	79.44	12	38	0.65
	TS	6000		602	96.64	100.30	0	100	0

Tableau 5.IX – L_{36} : résultats des algorithmes SCP-LP et TS, basés sur $r = 8$ réplications.

mulations par itération de SCP-LP. Une exécution complète de SCP-LP requiert au total environ 35 000 simulations. Étant donné que la durée d'une simulation doit être très courte (moins qu'une seconde), il y a alors plus de chance que la solution finale soit irréalisable pour ($SP1_{n_*}$). Les violations de contraintes diminuent cependant en augmentant le budget d'exécution. Également pour L_{52}, TS trouve toujours des solutions réalisables, mais elles sont environ 20% plus coûteuses que les solutions de SCP-LP. Plusieurs raisons expliquent les coûts élevés des solutions de TS. Quand nous comparons les meilleures solutions de SCP-LP et TS, nous trouvons que la solution de SCP-LP contient 94 agents contre 104 pour celle de TS. La solution de SCP-LP contient 6 agents spécialistes alors que celle de TS n'en contient qu'un seul. De plus, la solution de TS requiert 32 agents généralistes ayant 7 habiletés et plus, tandis que celle de SCP-LP n'en demande que 22. L'exemple L_{52} montre que l'algorithme SCP performe bien pour différentes structures de quarts de travail.

Budget (h)	Algo	n	n_2	CPU (min)	Min	Méd	P_1^*	P^*	G_p
5	SCP-LP	200	50	300	133.00	133.40	25	37	1.31
	TS	3000		257	161.50	167.85	0	100	0
10	SCP-LP	400	50	696	133.70	134.15	12	25	0.48
	TS	4400		561	158.60	163.40	0	100	0

Tableau 5.X – L_{52} : résultats des algorithmes SCP-LP et TS, basés sur $r = 8$ réplications.

5.4.4 Obtenir plus de solutions réalisables

Lors des expériences numériques, une des difficultés majeures de l'algorithme SCP est de trouver des solutions réalisables pour ($SP1_{n_*}$) lorsque le problème est grand. Dans cette section, nous essayons d'augmenter les chances d'obtenir des solutions réalisables avec SCP-LP pour l'exemple L_{52}.

Puisque les violations des contraintes sont souvent petites, une idée est d'incrémenter faiblement les seuils de $+1\%$ à $l = l_p = 0.81$ durant l'optimisation. Le test de réalisabilité final avec n_* réplications se fait avec les seuils originaux $l = l_p = 0.8$. De plus, nous voulons vérifier si le paramètre δ de la méthode d'arrondissement par seuil (voir la section 5.3.3.1, page 113) a un impact sur la solution finale. Nous exécutons une recherche binaire sur δ lors de la dernière itération de la procédure de coupes, mais nous arrondissons naïvement les solutions intermédiaires avec un paramètre δ fixe.

Dans ces expériences, nous optimisons le problème L_{52} à l'aide de l'algorithme relaxé SCP-LP avec trois différentes valeurs de $\delta \in \{0.5, 0.6, 0.7\}$ avec les seuils $l = l_p = 0.81$ durant l'optimisation. Nous considérons des budgets de temps de 5 et 10 heures, et nous exécutons $r = 8$ réplications de SCP-LP pour chaque test. Les résultats sont présentés dans le tableau 5.XI. Nous observons que les chances de trouver une solution réalisable pour ($SP1_{n_*}$) se sont grandement améliorées, et les coûts ne sont que légèrement supérieurs à ceux du tableau 5.X. Le choix de δ semble avoir un léger impact sur la solution finale. Comme mentionné à la section 5.3.3.1, page 113, SCP-LP tend à arrondir les solutions au plancher plus souvent lorsque δ est proche de 1. Ceci peut créer une attrition importante sur le nombre d'agents lorsque les valeurs à arrondir sont petites. Par conséquent, la solution risque davantage d'être dans une région non concave, et l'algorithme SCP-LP produira une mauvaise coupe ; c'est ce que nous observons dans le cas $\delta = 0.7$.

Nous expérimentons également avec $\delta \in \{0.5, 0.6, 0.7\}$, mais en gardant les seuils originaux $l = l_p = 0.8$ cette fois-ci. Les résultats obtenus démontrent que le choix de δ n'a pas un impact important sur la réalisabilité des solutions pour ($SP1_{n_*}$), car plus de 50% des solutions sont irréalisables.

Par ailleurs, nous essayons d'optimiser les autres exemples avec SCP-LP avec les seuils

δ	CPU (h)	Min	Méd	P^*
0.5	5	133.90	135.35	50
	10	133.70	134.85	100
0.6	5	133.70	135.00	87
	10	134.50	135.15	100
0.7	5	139.00	140.20	87
	10	136.70	139.15	75

Tableau 5.XI – L_{52} : résultats obtenus avec différents paramètres de δ et en incrémentant les seuils à $l = l_p = 0.81$ durant l'optimisation.

$l = l_p = 0.81$, mais en fixant $\delta = 0.5$. Voici un résumé des résultats :

- Petit centre d'appels modèle N : toutes les solutions obtenues sont réalisables sauf avec le plus petit budget de temps de 3 minutes. Plusieurs solutions sont légèrement plus chères que celles du tableau 5.III, mais nous trouvons aussi une meilleure solution avec un coût de 35.11.
- Centres d'appels moyens M_1 et M_2 : quasiment toutes les solutions sont réalisables, mais nous ne trouvons pas de meilleure solution.
- Grand centre d'appels L_{36} : toutes les solutions sont réalisables et nous trouvons une meilleure solution avec un coût de 78.27.

En conclusion, il est utile d'incrémenter légèrement les seuils des SL, même si le coût de la solution augmente légèrement, lorsque le centre d'appels doit absolument satisfaire les contraintes des SL. Cependant, la variabilité de nos résultats numériques montre qu'il n'y a aucune certitude que la solution sera réalisable ou optimale à cause de la nature stochastique du problème et de l'algorithme.

5.4.5 Qualité des solutions entières dans SCP-IP et TS

Les problèmes à nombres entiers de SCP-IP et TS sont résolus à l'aide de CPLEX 9.0 en utilisant des méthodes de branchement. Pour des raisons pratiques, nous devons généralement imposer une limite de temps sur l'exécution de CPLEX. Il est connu que ces méthodes passent la majorité du temps à vérifier l'optimalité de la solution. Il est possible que la solution retournée par CPLEX soit optimal (ou presque), même si nous l'arrêtons

prématurément. Dans cette section, nous offrons une brève discussion sur la qualité des solutions entières retournées par CPLEX.

Nous mesurons la qualité de la solution entière selon l'écart MIP en pourcentage (%) calculé comme suit :

$$\frac{c_{\text{entier}} - c_{\text{nœud}}}{c_{\text{entier}} + \varepsilon} \times 100,$$

où c_{entier} est le coût de la solution entière retournée, $c_{\text{nœud}}$ est le coût du meilleur nœud dans les branches restantes (une borne inférieure) et ε est un petit nombre positif. L'écart MIP mesure le potentiel d'amélioration du coût optimal de la solution, mais ce potentiel n'est pas nécessairement atteignable. Par exemple, une solution avec un écart MIP de 1% pourrait bien être optimale, sauf que CPLEX n'a pas eu le temps de valider son optimalité en purgeant toutes les branches. En résumé,

- Petit centre d'appels modèle N : les solutions de TS ont des écarts MIP plus petits que 0.3%, et les solutions de SCP-IP ont généralement des écarts inférieurs à 1.5%.
- Centres d'appels moyens M_1 et M_2 : les solutions de TS ont des écarts MIP inférieurs à 1.5%, mais les solutions de SCP-IP ont parfois des écarts jusqu'à 10% à cause des courtes limites de temps imposées à CPLEX.
- Grands centres d'appels L_{36} et L_{52} : les solutions de TS ont des écarts MIP en-dessous de 0.03% pour L_{36} et de 0.001% pour L_{52}. Ces problèmes sont trop grands pour être résolus avec SCP-IP.

5.4.6 Comparaison des méthodes d'arrondissement de SCP-LP

Pour arrondir les variables continues des solutions relaxées dans SCP-LP, nous avons présenté les méthodes par seuil (Seuil) et par fixation progressive des variables (FPV) à la section 5.3.3, page 112. Nous avons mentionné que nous avons choisi la méthode par seuil plutôt que FPV dans notre implémentation. Cette section présente des résultats numériques sur quelques exemples qui justifient ce choix. Notons que ces expériences supplémentaires ont été exécutées après la soumission de notre article [17].

Nous considérons aussi une variante de la méthode d'arrondissement par seuil, que nous appelons Seuil2. Cette variante de base sur l'hypothèse suivante : "Est-ce que la

cherche binaire sur δ à la fin du processus de coupes incrémente des variables $x_{i,s}$ inutilement ?" Puisque la méthode arrondit le nombre d'agents $x_{i,s}$ uniquement en fonction de sa valeur fractionnaire, il se pourrait que la méthode ajoute des agents qui n'ont aucun impact sur les contraintes violées.

Dans la variante Seuil2, nous remplaçons la recherche binaire par une méthode itérative. Premièrement, nous calculons $f_{i,s} = x_{i,s} - \lfloor x_{i,s} \rfloor$, la partie fractionnaire de $x_{i,s}$. Puis nous arrondissons toutes les variables $x_{i,s}$ au plancher. Cette méthode incrémente $N \geq 1$ variables $x_{i,s}$ (changer pour la valeur plafond) itérativement en suivant la liste triée par ordre décroissant des $f_{i,s}$ et conditionnellement aux contraintes insatisfaites. Nous incrémentons la variables $x_{i,s}$ seulement si la contrainte globale l est violée ou s'il existe une contrainte l_p, l_k ou $l_{k,p}$ violée telle que $k \in \mathcal{S}_i$ ou $\bar{a}(p,s) = 1$. Chaque variable $x_{i,s}$ ne peut être incrémentée qu'une seule fois. Nous arrêtons lorsque toutes les contraintes sont satisfaites ou s'il ne reste plus de candidat à incrémenter. Nous choisissons $N = 2$ dans nos expériences.

Nous testons sur deux exemples de centres d'appels : M_3 et L_{36}. L'exemple M_3 est identique à M_1, mais les taux d'arrivée sont multipliés par 5 et nous ne considérons que les quarts de travail d'une durée de 7.5 à 8 heures, pour un total de 159 quarts de travail au lieu de 285. Le tableau 5.XII présente les résultats avec $r = 8$ réplications d'optimisation pour chaque test. Il n'y a pas d'avantage net entre Seuil et Seuil2. Par contre, la méthode FPV est légèrement plus coûteuse que les deux autres et demande plus de temps d'exécution à cause des problèmes linéaires additionnels à résoudre. La raison des coûts plus élevés de FPV est que la méthode arrondit de manière à ce que la solution entière soit réalisable (selon le programme linéaire continu). Ceci peut causer une sur-affectation d'agents, et comme cette méthode est appelée à chaque itération de SCP-LP, alors les erreurs s'accumulent. Ainsi, nous conseillons d'utiliser plutôt les méthodes d'arrondissement par seuil.

5.5 Conclusion

Dans ce chapitre, nous avons présenté un algorithme de planification des quarts de travail dans un centre d'appels multi-compétences. Notre méthode est une extension des algorithmes de coupes par sous-gradients et la simulation de Ceẓik et L'Ecuyer [37] et

Exemple	Méthode	CPU (min)	Min	Méd	P^*
M_3	Seuil	130	95.87	98.33	38
	Seuil2	151	97.07	97.21	25
	FPV	184	102.43	102.89	38
L_{36}	Seuil	288	77.58	78.90	38
	Seuil2	286	77.61	78.05	75
	FPV	314	78.59	79.04	50

Tableau 5.XII – Comparaison des méthodes d'arrondissement de SCP-LP basées sur $r = 8$ réplications.

Atlason et al. [12]. Un agent est déterminé par son groupe et le quart de travail auquel il est affecté. Un quart de travail est défini par les périodes de travail et les périodes de pause. Nous utilisons le mécanisme de transfert d'agents proposé par Bhulai et al. [23] où un agent peut être transféré temporairement vers un groupe ayant moins d'habiletés.

Nous avons présenté notre méthode heuristique dans la section 5.3. Étant donné le nombre élevé de variables, les problèmes linéaires doivent généralement être résolus en relaxant les critères d'intégralité des variables entières. Puisque l'algorithme d'optimisation est itératif et le simulateur requiert un vecteur de paramètres entiers pour les agents, il est important de bien arrondir les solutions continues. Nous avons présenté deux méthodes heuristiques d'arrondissement dans la section 5.3.3. La première méthode arrondit chaque variable indépendamment en fonction d'un seuil fractionnaire. La deuxième méthode, plus complexe, est un processus itératif qui fixe quelques variables à des valeurs entières à la fois, puis résout le problème linéaire et répète ce processus jusqu'à ce que toutes les variables soient fixées. La deuxième méthode doit alors optimiser le problème linéaire beaucoup souvent. Les expériences numériques suggèrent que la méthode par seuil est meilleure en générale.

Nous complétons l'optimisation par une recherche locale utilisant la simulation, décrite à la section 5.3.4. Étant donné que les simulations doivent être courtes durant l'estimation des sous-gradients, la solution de l'algorithme de coupes peut contenir beaucoup de bruits, et la solution finale risque de ne pas être véritablement réalisable. Pour éviter ceci, la recherche locale commence par rendre la solution réalisable, pour des simulations

plus longues, en ajoutant des agents. Une fois la solution devenue réalisable, la recherche locale tente de réduire le coût total en retirant des agents ou en déplaçant des agents vers des groupes ou quarts de travail moins dispendieux. Ce processus peut être répété plusieurs fois en augmentant la longueur de simulation à chaque fois.

Dans la section numérique 5.4, nous avons comparé notre algorithme avec une approche traditionnelle à deux étapes, qui a été présentée à la section 5.2. Les exemples numériques montrent que notre algorithme trouve de meilleures solutions, qui sont aussi plus fréquemment réalisables, qu'avec l'approche à deux étapes. La différence de qualité des solutions est beaucoup plus frappante lorsqu'il y a davantage de types d'appels et groupes d'agents.

CHAPITRE 6

OPTIMISATION DES POLITIQUES DE ROUTAGE

Dans ce chapitre, nous étudions exclusivement les politiques de routage pour les centres d'appels avec plusieurs types d'appels et groupes d'agents. Nous supposons que les paramètres des divers processus du centre d'appels ainsi que l'affectation du personnel sont fixes. Le problème consiste uniquement à optimiser le routage des appels. Nous cherchons à maximiser les mesures de performance et d'équité tant au niveau des clients que des agents. Les contraintes sur les qualités de service sont traduites en coûts de pénalités dans la fonction objectif. Le problème d'optimisation ne contient aucune contrainte dure. Ce chapitre se base sur notre article sur le routage [38].

L'importance d'avoir un bon routage a déjà été mentionnée dans le chapitre d'introduction. Le problème de routage consiste à décider ce que le routeur doit faire lorsque : (1) un nouvel appel arrive ou (2) un agent termine un service et devient libre. Lorsqu'un appel arrive, le routeur doit choisir l'agent qui répondra à cet appel, parmi tous ceux qui sont libres et capables de servir cet appel. S'il n'y a aucun agent disponible, alors l'appel est placé à la fin de la file d'attente correspondant à son type d'appel. Quand un agent devient libre, le routeur décide du prochain appel à assigner à cet agent ou de laisser l'agent inoccupé.

Une politique de routage est dite *conservatrice de travail* si un agent ne peut pas être libre tant et aussi longtemps qu'il y a des appels en attente qu'il peut servir. Du point de vue

de la maximisation des qualités de service et de la productivité, nous serions porté à croire que la conservation du travail serait optimale, comme dans le modèle de base d'un seul type d'appel et un groupe d'agents. Cependant, les centres d'appels multi-compétences sont plus complexes, et les politiques de routage non conservatrices de travail sont plus flexibles et produisent parfois de meilleures solutions, comme nous le rapportons dans la section numérique. Voici deux exemples de règles non conservatrices de travail : (1) un agent ne peut pas répondre à un appel avant que le temps de délai depuis l'arrivée de cet appel soit expiré, et (2) réduire temporairement l'ensemble d'habiletés des agents quand le nombre d'agents libres est bas.

Optimiser le routage d'un centre d'appels avec plusieurs types et groupes d'agents est maintes fois plus dur qu'un centre avec un seul type d'appel et un groupe, à cause des nombreux choix de routage possibles. En principe, un gestionnaire devrait optimiser simultanément les ensembles d'habiletés des agents, les horaires des agents et la politique de routage en se basant sur les prévisions des appels. Par contre, ceci donne un problème d'optimisation très complexe, et les problèmes sont alors optimisés séparément en général. En pratique, la politique de routage est rarement changée une fois qu'elle est implantée. Elle est souvent considérée un paramètre fixe du modèle dans l'optimisation des horaires. Ironiquement, il n'est pas rare dans la pratique qu'un gestionnaire modifie le routage temporairement de manière ad hoc durant une journée pour faire face à une demande imprévue d'appels. Ceci montre l'importance d'optimiser le routage plus régulièrement.

Comme mentionné dans la revue de littérature à la section 3.4, page 61, il est théoriquement possible de trouver le routage optimale à l'aide de la programmation dynamique, mais plusieurs difficultés, dont la taille du problème, rendent cette approche peu utilisable en pratique. Plusieurs études dans la littérature proposent des règles de routage basées sur des conditions asymptotiques d'un système à trafic intense et imposent généralement des simplifications importantes au modèle du centre d'appels. Par contre, ces solutions ne sont pas nécessairement optimales pour les centres d'appels réels.

Nos travaux constituent une première étude sur des politiques de routage pragmatiques basées sur des modèles de centres d'appels réalistes. Notre contribution principale est une nouvelle politique de routage basé sur des poids, les temps d'attente des appels et les temps

d'inactivité des agents. Le nombre de paramètres considérés est relativement faible afin de faciliter l'implantation et l'optimisation. Le routage est optimisé à l'aide d'algorithmes heuristiques pour des fonctions de type *boîte noire* et la simulation. L'algorithme d'optimisation ne peut que passer des paramètres en entrée à la fonction boîte noire et de retirer les résultats à la sortie. En particulier, il ne connaît ni la formule, ni les dérivées de la fonction. Cette approche permet d'optimiser divers choix de fonctions objectifs avec un minimum de modifications aux algorithmes. De plus, nous pouvons optimiser les routages d'une plus grande diversité de centres d'appels qu'avec les méthodes basées sur des approximations analytiques. Nous comparons les performances de notre politique avec celles que nous retrouvons dans l'industrie et la littérature. Les exemples numériques montrent que notre nouvelle politique de routage donne des résultats supérieurs ou égaux aux meilleures politiques existantes.

Le reste du chapitre est organisé comme suit. La prochaine section décrit le modèle de centre d'appels auquel nous optimisons le routage. Nous présentons les fonctions objectifs considérées dans les exemples numériques. La section 6.2 présente notre nouvelle politique de routage basé sur des poids et quelques politiques de routage que nous incluons dans nos exemples numériques et qui n'ont pas été décrites à la section 2.2, page 17. La section 6.3 présente les algorithmes d'optimisation que nous avons implémentés, soient une méthode de descente du gradient et un algorithme génétique modifié. Nous détaillons sur la manière d'adapter ces algorithmes afin d'optimiser les paramètres des différentes politiques de routage. Finalement, la section 6.4 compare les performances de notre politique de routage avec les autres politiques, et présente quelques résultats numériques.

6.1 Modèle du centre d'appels

Nous considérons un centre d'appels avec K types d'appels, numérotés de 1 à K, et I groupes d'agents, numérotés de 1 à I. L'ensemble d'habiletés du groupe d'agents i est donné par $\mathcal{S}_i \subseteq \{1,\ldots,K\}$, et l'ensemble des groupes pouvant servir le type d'appel k par $\mathcal{T}_k = \{i : k \in \mathcal{S}_i, \forall i\}$. Étant donné que l'optimisation est basée sur la simulation, nous ne supposons aucun modèle spécifique pour les processus stochastiques (les temps d'arri-

vée, les durées de service, etc.). L'avantage de notre approche par rapport aux méthodes analytiques est que nous pouvons optimiser n'importe quel centre d'appels pouvant être modélisé par un simulateur.

Nous supposons qu'il y a une file d'attente par type d'appel. Lorsqu'un nouvel appel ne peut pas être servi immédiatement, il est placé à la fin de sa file d'attente. Nous supposons que chaque appel ne demande qu'un seul type de service, et qu'il ne peut quitter le système que s'il a été répondu par un agent ou par abandon. Nous supposons que le service est non préemptif, c'est-à-dire qu'un agent ne peut interrompre un service en cours pour répondre à un autre client. Nous simplifions les problèmes en ne considérant qu'une seule période. Le vecteur d'affectation des agents $\mathbf{y} = (y_1, \ldots, y_I)^{\mathrm{T}}$ est fixé au problème, et nous optimisons uniquement les paramètres de routage.

Un autre avantage d'utiliser la simulation est que nous pouvons optimiser n'importe quelle mesure de performance implémentable par un simulateur. Dans nos exemples numériques, nous considérons les mesures du niveau de service $f_{\mathrm{S}}(\pi, \tau)$, du taux d'abandons $f_{\mathrm{A}}(\pi)$, du temps d'attente moyen $f_{\mathrm{W}}(\pi)$, et du taux d'occupation des groupes d'agents $f_{\mathrm{O},i}(\pi)$, qui sont définies à la section 2.3, page 20. Nous ajoutons l'argument π aux fonctions des mesures de performance afin d'indiquer qu'elles dépendent de la politique du routage π.

Les mesures de performance sont souvent formulées par des contraintes dures dans les problèmes d'optimisation trouvés dans la littérature. Dans notre étude, nous relaxons ces contraintes en les transformant en coûts de pénalités. Le problème n'a alors aucune contrainte, et le but est de minimiser les coûts de pénalités. La fonction objectif peut inclure un nombre infini de mesures de performance ; il suffit de bien pondérer les différents coûts de pénalités. L'utilisation de coûts de pénalités au lieu de contraintes dures convient mieux aux problèmes réels des centres d'appels. Rater un objectif de performance par une faible marge est souvent acceptable en pratique. De plus, ces mesures sont souvent des variables aléatoires qu'il faut estimer à l'aide de la simulation. Déterminer si une solution est réalisable ou non (si toutes les contraintes sont satisfaites ou non) devient difficile à cause du bruit de la simulation. Minimiser ce bruit peut demander de très longues simulations puisque les solutions optimales sont généralement sur la frontière de réalisabilité.

La fonction de pénalité peut prendre n'importe quelle formulation, mais nous optons pour des fonctions polynomiales dans nos exemples numériques. Puisque nous n'optimisons que la politique de routage (les autres des paramètres sont fixés), la fonction de pénalité prend en argument le routage π. Nous présentons maintenant les fonctions objectifs qui seront utilisées dans la section numérique.

1. La fonction de pénalité sur les SL est :

$$F_{\mathrm{S}}(\pi) = \sum_{k=1}^{K} c_{\mathrm{S},k} \max(l_k - f_{\mathrm{S},k}(\pi, \tau_k), 0)^{e_{\mathrm{S},k}}, \tag{6.1}$$

   où pour chaque type d'appel k, τ_k est le temps d'attente acceptable, l_k est le seuil minimal du SL, $c_{\mathrm{S},k}$ est le poids de pénalité, et $e_{\mathrm{S},k} \geq 0$ est le degré du polynôme. Cette forme polynomiale permet de pénaliser davantage lorsqu'un SL est substantiellement inférieur à son seuil. Ceci représente mieux les problèmes réels où il est habituellement acceptable de rater l_k par une petite fraction, mais qu'il est inadmissible de le rater par une grande marge. Cette forme est intéressante aussi, parce qu'elle permet d'avoir un critère d'équité entre les types d'appels. Il est souvent préférable d'avoir deux types d'appels qui ratent leur cible de $p\%$ que d'avoir un type qui rate de $2p\%$ et l'autre type parfaitement sur sa cible. Les coefficients $c_{\mathrm{S},k}$ permettent de contrôler l'importance des types d'appels. Par exemple, ils pourraient être proportionnels aux taux d'arrivée, mais un type d'appel ayant un faible volume pourrait devenir insignifiant au problème. Nous pourrions facilement inclure la mesure de performance agrégée sur tous les types d'appels dans l'équation.

2. La fonction de pénalité sur les abandons est :

$$F_{\mathrm{A}}(\pi) = \sum_{k=1}^{K} c_{\mathrm{A},k} \max(f_{\mathrm{A},k}(\pi) - u_{\mathrm{A},k}, 0)^{e_{\mathrm{A},k}}, \tag{6.2}$$

   où pour chaque type d'appel k, $c_{\mathrm{A},k}$ est le poids de pénalité, $e_{\mathrm{A},k} \geq 0$ est le degré du polynôme, et $u_{\mathrm{A},k}$ est le seuil acceptable d'abandons. Comme la fonction $F_{\mathrm{S}}(\pi)$, nous contrôlons les pénalités grâce aux paramètres $c_{\mathrm{A},k}$, $u_{\mathrm{A},k}$ et $e_{\mathrm{A},k}$. Il est connu qu'un peu d'abandons peut améliorer significativement les qualités de service d'un centre

d'appels. Ainsi, les gestionnaires tolèrent habituellement un faible taux d'abandons, disons moins de 2%. Nous choisissons néanmoins les seuils $u_{\mathrm{A},k} = 0$ dans nos exemples numériques.

3. La fonction de pénalité sur les temps d'attente moyens est :

$$F_{\mathrm{W}}(\pi) = \sum_{k=1}^{K} c_{\mathrm{W},k} \max(f_{\mathrm{W},k}(\pi) - u_{\mathrm{W},k}, 0)^{e_{\mathrm{W},k}}, \tag{6.3}$$

où pour chaque type d'appel k, $f_{\mathrm{W},k}$ est le temps d'attente moyen, $c_{\mathrm{W},k}$ est le poids de pénalité, $u_{\mathrm{W},k}$ est le seuil minimal, et $e_{\mathrm{W},k} \geq 0$ est le degré du polynôme.

4. Pour le taux d'occupation des agents, nous considérons juste le critère d'équité. L'équation serait similaire à $F_{\mathrm{S}}(\pi)$ s'il y avait des seuils minimaux d'occupation. La fonction de pénalité pour une occupation équitable est :

$$F_{\mathrm{O}}(\pi) = \sum_{i=1}^{I} c_{\mathrm{O},i} \left| f_{\mathrm{O},i}(\pi) - \bar{O} \right|^{e_{\mathrm{O},i}}, \tag{6.4}$$

où pour chaque groupe i, $c_{\mathrm{O},i}$ est le poids de pénalité, $e_{\mathrm{O},i}$ est le degré du polynôme, et $\bar{O}$ est le taux moyen d'occupation des groupes, défini par $\bar{O} = \sum_{i=1}^{I} f_{\mathrm{O},i}(\pi)/I$ dans nos exemples numériques. L'opérateur $|x|$ retourne la valeur absolue de x. Nous pourrions remplacer $\bar{O}$ par le taux moyen d'occupation pondéré par le nombre d'agents $\bar{O}_{\mathrm{p}} = \dfrac{\sum_{i=1}^{I} y_i f_{\mathrm{O},i}(\pi)}{\sum_{i=1}^{I} y_i}$. Notons que $\bar{O} = \bar{O}_{\mathrm{p}}$ dans le cas optimal où tous les taux d'occupation sont égaux. Nous pourrions également utiliser cette formulation pour pénaliser l'iniquité des types d'appels.

Notre approche d'optimisation permet une grande flexibilité sur la modélisation du problème. Dans la section numérique, nous combinons les différentes fonctions objectifs définies ci-haut pour obtenir une fonction générale :

$$F_{\mathrm{SAWO}}(\pi) = F_{\mathrm{S}}(\pi) + F_{\mathrm{A}}(\pi) + F_{\mathrm{W}}(\pi) + F_{\mathrm{O}}(\pi). \tag{6.5}$$

Les poids $c_{\mathrm{S},k}, c_{\mathrm{A},k}, c_{\mathrm{W},k}$ et $c_{\mathrm{O},i}$ permettent de déterminer l'importance de chaque élément dans $F_{\mathrm{SAWO}}(\pi)$. Pour retirer un terme, il suffit d'assigner les poids correspondants à zéro.

Pour simplifier les notations, nous cachons l'indice $*$ dans F_{SAWO} lorsque $c_{*,j} = 0$ pour tout j, où j peut être un type d'appel ou un groupe d'agent. Par exemple, nous dénotons la fonction F_{SAWO} par F_{SA} lorsque $c_{\mathrm{W},k} = c_{\mathrm{O},i} = 0$ pour tout k et tout i.

Remarquons que si $F_{\mathrm{SAWO}}(\pi) = F_{\mathrm{S}}(\pi)$, c'est-à-dire que tous les $c_{\mathrm{A},k} = c_{\mathrm{W},k} = c_{\mathrm{O},i} = 0$, alors le centre d'appels est assujetti à satisfaire seulement les seuils des SL. Il n'y a aucun gain à répondre aux clients ayant attendu plus que les AWT τ_k. Un routage optimal ne devrait pas servir les appels ayant attendu plus que τ_k (si cette discrimination est permise). La politique de routage populaire du *premier arrivé, premier servi* ne serait pas optimale dans cette situation.

Rappelons que nos méthodes d'optimisation sont conçues indépendamment du choix de la fonction objectif qui est considérée comme une fonction boîte noire. Nous pouvons facilement remplacer (6.5) par une autre fonction complètement différente. Un grand avantage de l'approche d'optimisation boîte noire est la facilité de changer la fonction objectif avec un minimum de modifications aux programmes. Nous supposons aussi que le problème ne contient aucune contrainte dure. Nous utilisons la simulation pour évaluer les coûts des solutions. Les inconvénients sont que les coûts retournés contiennent des bruits de simulation et que la simulation est généralement le goulot d'étranglement des algorithmes d'optimisation.

6.2 Politiques de routage

Nous présentons les politiques de routage considérées dans cette étude. Plusieurs de ces politiques peuvent être retrouvées dans la pratique (avec quelques variations possibles). Nous introduisons aussi notre nouvelle politique de routage basé sur des poids. Toutes les politiques de routage considérées respectent les deux règles d'équité de base définies à la section 2.2 : (1) les appels du même type sont toujours *premier arrivé, premier servi* (FCFS), et (2) parmi les agents du même groupe, *le premier agent inactif est choisi avant* (LISF).

Trois politiques de routage ont déjà été présentées à la section 2.2, page 17 :

1. Routage à priorités égales (E),

2. Routage par priorités (P),

3. Routage par priorités et temps de délai (PD).

Nous présentons maintenant quatre autres politiques.

6.2.1 Routage par priorités et seuils d'agents libres (PS)

Cette politique de routage est basée sur la gestion du routage dans certains centres d'appels réels. Il arrive qu'un gestionnaire essaie d'optimiser indirectement le routage en restreignant temporairement, et de manière ad hoc, les habiletés d'un groupe lorsque le nombre d'agents libres devient faible. Cette stratégie de routage est utilisée pour prévenir une pénurie d'agents libres ayant une habileté particulière, pour favoriser des types d'appels plus importants à des moments critiques, ou pour réduire le taux d'occupation de certains agents par exemple. La politique que nous proposons ici essaie de simuler ce type de routage en ajoutant des conditions sur le nombre d'agents libres à la politique P.

Nous implémentons l'idée, proposée par Gans et al. [58], de représenter les seuils d'agents libres par des paramètres continus. Il y a un paramètre continu $m_{k,i} \geq 0$ représentant le seuil d'agents libres pour chaque paire de groupe i et type d'appel $k \in \mathcal{S}_i$. Supposons que le routage par priorités choisit un agent libre du groupe i pour répondre à un appel de type k. Soient u_i le nombre d'agents libres du groupe i et $\delta_{k,i} = m_{k,i} - \lfloor m_{k,i} \rfloor$ la partie fractionnaire de $m_{k,i}$. Si $u_i > \lceil m_{k,i} \rceil$, alors l'agent répond à l'appel. Si $u_i = \lceil m_{k,i} \rceil$, alors l'agent répond à l'appel avec une probabilité de $1 - \delta_{k,i}$. Autrement, le groupe i ne peut pas répondre à l'appel, et le routeur continue avec le prochain groupe dans la liste des priorités. À cause de la supposition du service non préemptif, cette condition n'a aucun effet sur les agents déjà en service, c'est-à-dire qu'un agent occupé du groupe i n'arrêtera pas le service à un client de type k même si le nombre d'agents libres dans son groupe devient inférieur ou égal à $m_{k,i}$.

6.2.2 Routage par priorités, délais et seuils d'agents libres (PDS)

Nous combinons les politiques des routage PD et PS pour avoir une politique plus générale avec des priorités, temps de délai et seuils d'agents libres. Cette politique est plus

flexible, et elle devrait en principe offrir des performances égales ou supérieures aux autres politiques avec des règles plus contraignantes : E, P, PD et PS. Cependant, cette flexibilité vient avec le prix d'un plus grand nombre de paramètres à optimiser. Les listes de priorités sont des solutions combinatoires, puis les temps de délai et les seuils d'agents libres sont des variables continues. Optimiser toutes ces différentes variables simultanément peut devenir très difficile quand il y a beaucoup de type d'appels et de groupes d'agents. Notre approche est d'optimiser itérativement une ou deux composantes à la fois.

6.2.3 Adaptation linéaire de la règle $c\mu$ généralisée (LG$c\mu$)

Supposons que nous voulons minimiser

$$\sum_{k=1}^{K}\sum_{n=1}^{N_k} C_k(w_{k,n}), \tag{6.6}$$

où N_k est le nombre total d'appels du type k, $w_{k,n}$ est le temps d'attente du n-ième appel du type k, et C_k est une fonction de coût convexe non décroissante qui a pour dérivée première C'_k. Lorsqu'un agent du groupe i devient libre, le routeur choisit l'appel en attente de type k^* qui maximise le taux de coût instantané :

$$k^* = \arg\max_{k\in\mathcal{S}_i} C'_k(w_k)\mu_{k,i}, \tag{6.7}$$

où w_k est le temps d'attente de l'appel à la tête de la file k, et $\mu_{k,i}$ est le taux de service du type k par un agent du groupe i. Cette politique s'appelle *la règle $c\mu$ généralisée* (G$c\mu$), et elle a été prouvée asymptotiquement optimale pour la fonction de coût (6.6) pour certains régimes à trafic intense. Nous retrouvons plusieurs études [11, 46, 102, 127, 130, 131] sur la règle G$c\mu$ dans la littérature, voir la section 3.4, page 61.

Dans tous les cas, la condition d'optimalité asymptotique ne tient que pour des types de fonctions objectifs bien spécifiques, avec des hypothèses de simplification. La politique G$c\mu$ ne marche pas nécessairement bien dans les centres d'appels réels, et les fonctions objectifs ne s'expriment pas toujours sous la forme de (6.6), comme dans notre étude.

Puisque les fonctions objectifs (6.1) à (6.5) ne sont pas sous la forme (6.6), nous ne

pouvons pas utiliser directement la politique G$c\mu$. Nous proposons une variante de la règle G$c\mu$ en remplaçant la dérivée C'_k dans (6.7) par une fonction linéaire de la forme $C'_k(w) = a_k + b_k w$, d'où le nom de la *règle cμ généralisée linéaire* (LG$c\mu$). Contrairement à la politique G$c\mu$, nous devons optimiser les paramètres $a_k, b_k \geq 0$ dans la version LG$c\mu$. Remarquons que si $C_k(w) = a_k w$ dans (6.6), alors la dérivée $C'_k(w) = a_k$, et ce cas est couvert par LG$c\mu$. La fonction (6.7) pour sélectionner le type d'appel k^* devient :

$$k^* = \arg\max_{k \in \mathcal{S}_i} (a_k + b_k w_k)\mu_{k,i}.$$

À cause des hypothèses de régime à trafic intense, la politique G$c\mu$ originale n'a aucune règle pour choisir un agent libre lorsqu'un nouvel appel arrive. Dans la réalité et dans nos exemples numériques, le routeur doit faire ce choix. Dans la politique LG$c\mu$, nous utilisons une autre fonction linéaire pour choisir un agent libre du groupe i^* quand un nouvel appel du type k arrive :

$$i^* = \arg\max_{i \in \mathcal{T}_k} (e_i + f_i v_i)\mu_{k,i},$$

où v_i est le temps d'inactivité de l'agent le plus longtemps inactif dans le groupe i. Ajouter le temps d'inactivité des agents dans LG$c\mu$ permet de mieux optimiser le taux d'occupation des agents. En cas d'égalité, nous choisissons le groupe i^* aléatoirement avec des probabilités proportionnelles au nombre d'agents libres.

Si nous choisissons $a_k = 1$ et $b_k = 0$ pour tout k, et $e_i = 1$ et $f_i = 0$ pour tout i, la règle LG$c\mu$ devient la politique du *serveur le plus rapide en premier* ou "*fastest-server-first*" (FSF). Tezcan et Dai [127] montrent que la politique FSF est asymptotiquement optimale pour le cas spécial d'un modèle N (voir la figure 2.1, page 16), sous un régime à trafic intense, avec des coûts linéaires sur les temps d'attente et les abandons, et des taux de service qui dépendent uniquement des groupes d'agents.

Toutes les politiques de type G$c\mu$ sont *conservatrices de travail*, ce qui peut être sous-optimal pour certains centres d'appels. Elles ont cependant l'avantage d'être des politiques faciles et simples à implanter, et elles sont robustes aux taux d'arrivée stochastiques sous certaines conditions. Notons que nous devons optimiser les $2(K+I)$ paramètres de la politique LG$c\mu$, alors que la règle G$c\mu$ originale prend ses paramètres directement de la

dérivée de C_k (donc aucune optimisation requise).

6.2.4 Routage basé sur des poids (BP)

Nous présentons notre nouvelle politique de routage basé sur des poids, définie en fonction des temps d'attente des clients et des durées d'inactivité des agents. Nous affectons un poids $c_{k,i} \in \mathbb{R}$ à chaque paire de groupe d'agents i et type d'appel $k \in \mathcal{S}_i$, et $c_{k,i}$ peut être interprété comme étant le niveau de priorité de cette paire. Dans cette étude, nous définissons $c_{k,i}$ par une fonction linéaire :

$$c_{k,i} = q_{k,i} + a_{k,i} w_k + b_{k,i} v_i, \tag{6.8}$$

où w_k est le temps d'attente de l'appel à la tête de la file k, v_i est le plus long temps d'inactivité parmi les agents libres du groupe i, et $a_{k,i}, b_{k,i} \geq 0$ et $q_{k,i} \in \mathbb{R}$ sont des paramètres continus à optimiser. Seulement les paramètres $q_{k,i}$ peuvent prendre des valeurs négatives. Nous ne permettons pas d'avoir des $a_{k,i} < 0$, car ceci signifierait que la priorité d'un appel diminue lorsque le temps d'attente augmente. Ceci pourrait augmenter le SL en forçant les appels ayant attendu longtemps à abandonner, mais cette stratégie n'est pas équitable et irréaliste en pratique. Les clients ayant attendu longtemps risquent d'être furieux ! D'autre part, il ne semble pas logique d'avoir des $b_{k,i} < 0$, parce que la priorité de l'agent diminue avec son temps d'inoccupation, jusqu'à possiblement ne plus servir d'appel si $c_{k,i} < 0$. Néanmoins, nous présenterons et expérimenterons plus loin une variante de BP où tous les paramètres $q_{k,i}, a_{k,i}$ et $b_{k,i}$ peuvent être négatifs.

La politique de routage BP fonctionne comme suit. S'il n'y a aucun agent libre dans le groupe i ou aucun appel dans la file d'attente k, alors nous attribuons $c_{k,i} = -\infty$. Cette règle suppose que tous les nouveaux appels doivent obligatoirement entrer dans une file d'attente. Si un appel est répondu immédiatement à son arrivée par un agent libre, alors il sortira aussitôt de la file et son temps d'attente sera zéro. Si tous les $c_{k,i} < 0$, alors aucune action n'est exécutée par le routeur. S'il y a au moins un $c_{k,i} \geq 0$, alors le routeur choisit la paire $(k^*, i^*) = \arg\max_{k,i} c_{k,i}$, et assigne l'appel à la tête de la file d'attente k^* à l'agent le plus longtemps inactif du groupe i^*.

Contrairement à G$c\mu$ et LG$c\mu$, la politique BP peut être non conservatrice de travail. Un paramètre $q_{k,i} < 0$ peut être utilisé pour créer une condition de délai sur la paire (k,i). Le routeur ne peut pas affecter un agent libre du groupe i à un appel en attente du type k tant et aussi longtemps que $a_{k,i}w_k + b_{k,i}v_i < -q_{k,i}$, autrement dit $c_{k,i} < 0$. La possibilité de délai peut être utile dans certaines situations, en particulier lorsque le SL est une composante importante de la fonction objectif. Dans un tel cas, nous voudrions que le temps de délai soit inférieur au AWT.

Le routeur doit mettre à jour régulièrement les poids $c_{k,i}$, car les variables w_k et v_i sont dynamiques, et elles évoluent avec le temps et l'état du centre d'appels. Si les paramètres $a_{k,i}, b_{k,i}$ et $q_{k,i}$ sont contraints à être non négatifs, alors le routeur peut mettre à jour les poids $c_{k,i}$ uniquement aux instants d'arrivée ou aux moments de fin de service des appels. Quand il y a des $q_{k,i}$ négatifs, il faut recalculer les poids $c_{k,i}$ plus fréquemment. Il est cependant facile de prédire le prochain moment d'expiration d'un délai lorsque les poids sont exprimés sous la forme linéaire (6.8). Mais pour des formulations plus complexes ou qui dépendent de l'état du système, il serait plus simple de mettre à jour les poids régulièrement et rapidement, à chaque seconde par exemple, au lieu de prédire les moments d'expiration des délais.

La politique BP définie par l'équation (6.8) contient $3\sum_{i=1}^{I} |\mathcal{S}_i|$ paramètres à optimiser, où $|\mathcal{S}_i|$ est le nombre d'habiletés du groupe i. Nous considérons quelques variantes de BP dans la section numérique avec différents calculs des poids $c_{k,i}$:

BP-S : $c_{k,i} = q_{k,i} + a_k w_k + b_i v_i$,

BP-S2 : $c_{k,i} = q_k + r_i + a_k w_k + b_i v_i$,

BP-A : $c_{k,i} = q_{k,i} + a_{k,i} w_k + b_{k,i} u_i$, où u_i est le nombre d'agents libres,

BP-AS : $c_{k,i} = q_{k,i} + a_{k,i} w_k + b_{k,i} v_i$, et conditionnel au seuil $m_{k,i}$,

BP-N : BP avec $a_{k,i}, b_{k,i} \in \mathbb{R}$ (peuvent avoir des valeurs négatives).

Dans BP-S, nous réduisons le nombre de paramètres en retirant l'indice i de $a_{k,i}$ et en retirant l'indice k de $b_{k,i}$. Nous réduisons davantage le nombre de paramètres dans BP-S2 en remplaçant le poids $q_{k,i}$ par $q_k + r_i$, où q_k ne dépend que du type k et r_i ne dépend que du groupe i. Comme BP-S et BP-S2 sont des versions restreintes de BP, elles ne devraient pas réussir mieux que BP. Cependant, puisqu'elles ont moins de paramètres à optimiser que BP,

soient $K+I+\sum_{i=1}^{I}|\mathcal{S}_i|$ et $2(K+I)$ paramètres respectivement contre $3\sum_{i=1}^{I}|\mathcal{S}_i|$ pour BP, il est possible que ces variantes réussissent mieux en pratique. Notons que nous pourrions prendre la solution de BP-S ou BP-S2 comme solution de départ dans l'optimisation de BP (mais nous n'avons pas fait ceci dans nos expériences numériques).

Dans BP-A, nous remplaçons la variable v_i par u_i qui représente le nombre d'agents libres du groupe i. Mettre à jour les poids dans BP-A devient plus simple. Dans BP-AS, nous gardons v_i et nous ajoutons un seuil $m_{k,i} \geq 0$ sur u_i semblable à la politique PS. La condition du seuil d'agents libres $m_{k,i}$ est vérifiée avant le calcul des poids $c_{k,i}$. Le fonctionnement du seuil d'agents libres est exactement pareil à la politique PS. Le poids $c_{k,i} = -\infty$ si le seuil, qui est aléatoire si $m_{k,i}$ n'est pas entier, est insatisfait. Ces variantes semblent mieux performer lorsqu'il y a des types d'appels avec de faibles volumes. BP-A a souvent été une des meilleures politiques de routage dans nos expériences numériques.

Finalement, la variante BP-N permet à tous les paramètres de prendre des valeurs négatives. Il est intéressant d'observer ce qui se passe lorsque nous permettons à tous les paramètres d'être négatifs. Par exemple, quand la fonction objectif est F_{S}, il y a moins d'intérêt à répondre aux appels ayant attendu plus que AWT (sauf pour accéder aux appels à la fin de la file qui ont attendu moins que AWT). Pour approximer ceci grossièrement, la politique BP-N choisit souvent des coefficients $a_{k,i} < 0$ tels que le poids (priorité) d'un appel diminue lorsque son temps d'attente augmente. La priorité est donnée aux appels ayant attendu moins que AWT en attribuant des $q_{k,i} > 0$ suffisamment grands. Toutefois, ceci ne mène pas à une politique optimale pour F_{S}, car les appels du même type doivent être répondus en respectant la règle FCFS. Mettre $b_{k,i} < 0$ ne semble pas logique, et nous observons ces cas très rarement dans nos solutions numériques.

Notons que si $a_{k,i}, b_{k,i}, q_{k,i} < 0$, alors un agent du groupe i ne servira jamais un appel du type k. Si tous les paramètres $a_{k,i}, b_{k,i}, q_{k,i} = 0$, alors l'assignation d'un appel est choisie *aléatoirement* parmi les agents libres capables de le servir (tout en respectant les règles de base FCFS et LISF). Remarquons qu'un routage complètement aléatoire peut donner des SL supérieurs que ceux des routages traditionnels lorsque le système est surchargé. Un appel (disons à la tête de la file k_1) arrivé récemment a autant de chance d'être sélectionné qu'un appel (disons à la tête de la file k_2) en attente depuis très longtemps. Par contre, le

nombre d'abandons risque fort d'augmenter. Il n'est pas certain non plus qu'une politique de routage totalement aléatoire soit considérée équitable en pratique. Néanmoins, ce type de routage aléatoire est utilisé dans des domaines tels que les réseaux de télécommunication, où un paquet d'information peut être envoyé des dizaines ou des centaines de fois à la seconde, et que les abandons sont moins pénalisants.

Pour décrire la flexibilité du routage basé sur des poids, nous examinons à présent comment approximer les politiques E, P et PD à l'aide de la politique BP.

Politique E : La politique de routage BP peut approximer la politique de routage à priorités égales en choisissant $q_{k,i} = 0, a_{k,i} = 1$ et $b_{k,i} = \varepsilon$ pour toute paire (k,i), où $\varepsilon > 0$ est un petit nombre. Quand un agent devient libre, le choix du prochain appel à servir dépend quasi uniquement des temps d'attente des appels. Lorsqu'un nouvel appel arrive, l'agent le plus longtemps inactif et capable de servir cet appel sera sélectionné.

Politique P : Si la politique P utilise seulement un des deux types de priorités, soient type-vers-groupe ou groupe-vers-type, ou si les listes de priorités sont symétriques (par exemple lorsque le type primaire d'un groupe et le groupe primaire d'un type sont la même paire, ce qui arrive souvent en pratique), alors il y a deux possibilités : (1) attribuer $q_{k,i} \geq 0$ par ordre décroissant en suivant les priorités et $a_{k,i} = 1$, ou (2) attribuer $a_{k,i} > 0$ par ordre décroissant en suivant les priorités et $q_{k,i} = 0$. Puis choisir $b_{k,i} = \varepsilon$ dans les deux cas.

Si les listes de priorités ne sont pas symétriques, alors nous pouvons utiliser les paramètres $q_{k,i}$ pour reproduire les priorités type-vers-groupe, et choisir les $a_{k,i}$ de manière à refléter les priorités groupe-vers-type, puis $b_{k,i} = \varepsilon$. En effet, quand un appel arrive, son temps d'attente est zéro, alors le paramètre $a_{k,i}$ n'a aucune influence sur le poids $c_{k,i}$. Les paramètres $a_{k,i}$ doivent être suffisamment grands afin que les $q_{k,i}$ deviennent négligeables lorsqu'un agent libre cherche un appel en attente à servir.

Politique PD : Pour approximer la politique PD, nous pouvons utiliser la même approche que pour la politique P. Remarquons qu'un appel du type k ne peut pas avoir simultanément une priorité type-vers-groupe au groupe i et un temps de délai $d_{k,i} > 0$. C'est-à-dire que si le groupe i impose un temps de délai $d_{k,i}$ aux appels du type k,

alors la priorité type-vers-groupe du type k vers le groupe i ne sert à rien. Ainsi, nous pouvons utiliser le paramètre $q_{k,i}$ pour imposer soit le temps de délai ou bien la priorité type-vers-groupe. Il faut choisir les $a_{k,i}$ en fonction des priorités groupe-vers-type et des délais, et $b_{k,i} = \varepsilon$.

En principe, les priorités groupe-vers-type vont dominer lorsque le centre d'appels est surchargé (la majorité des appels doivent attendre), et les priorités type-vers-groupe vont dominer si le centre est sous-chargé (la plupart des appels sont répondus à leur arrivée). Dans ces cas, il est possible d'avoir une bonne approximation en convertissant uniquement les priorités groupe-vers-type lorsque le centre d'appels est surchargé et uniquement les priorités type-vers-groupe quand le centre est sous-chargé. Nous pouvons simplifier la politique BP en approximant les priorités par les paramètres $q_{k,i}$ avec la variante BP-S.

Étant donné que la politique de routage BP peut approximer les politiques E, P et PD, nous nous attendons à ce que la politique BP donne des performances égales ou supérieures à ces trois politiques.

Comparons maintenant les politiques BP et LG$c\mu$. Remarquons que la politique BP serait une généralisation de LG$c\mu$ si nous remplacions $q_{k,i}$ par les paramètres $q^{\mathrm{A}}_{k,i}$ et $q^{\mathrm{C}}_{k,i}$, où A représente la sélection d'un agent libre et C, la sélection d'un appel en attente. La formule de poids serait :

$$c'_{k,i} = q^{\mathrm{A}}_{k,i}\mathbb{I}[v_i > 0] + q^{\mathrm{C}}_{k,i}\mathbb{I}[w_k > 0] + a_{k,i}w_k + b_{k,i}v_i.$$

Nous supposons que la probabilité qu'un appel arrive et qu'un agent devient libre exactement au même moment soit nulle, c'est-à-dire $w_k = v_i = 0$. Il n'y a aucun problème quand il n'y a pas de possibilité de délai ($q^{\mathrm{A}}_{k,i}, q^{\mathrm{C}}_{k,i} \geq 0$). Au lieu d'avoir une seule formule de poids $c_{k,i}$, nous pouvons décomposer $c'_{k,i}$ en : (1) poids $c^{\mathrm{A}}_{k,i} = q^{\mathrm{A}}_{k,i} + b_{k,i}v_i$ pour sélectionner un agent libre quand un appel arrive, et (2) poids $c^{\mathrm{C}}_{k,i} = q^{\mathrm{C}}_{k,i} + a_{k,i}w_k$ pour sélectionner un appel en attente quand un agent devient libre. Ceci est une simple généralisation de la politique LG$c\mu$ avec des paramètres qui dépendent de k et i.

Par contre, il n'est pas évident de gérer les délais ($q^{\mathrm{A}}_{k,i}, q^{\mathrm{C}}_{k,i} \in \mathbb{R}$) avec la formule de poids $c'_{k,i}$, car les variables w_k et v_i pourraient être positives en même temps. La règle de

délai pourrait avoir des effets non désirés. Par exemple, si $q^{\mathrm{A}}_{k,i} > 0$ et $q^{\mathrm{C}}_{k,i} < 0$, alors il n'y pas de délai quand un nouvel appel arrive, mais le délai peut être activé quand l'appel est déjà en attente ! Pour cette raison, nous n'avons pas implémenté la politique BP avec les paramètres $q^{\mathrm{A}}_{k,i}$ et $q^{\mathrm{C}}_{k,i}$. Notons que la règle de délai est cohérente dans la politique PD, car il n'y a qu'un seul paramètre de délai $d_{k,i}$ pour chaque paire (k,i), et aussi dans la politique BP parce qu'il n'y a qu'un seul $q_{k,i}$ pour chaque paire (k,i) dans la formule de poids $c_{k,i}$.

La fonction linéaire dans (6.8) est une manière de calculer les poids $c_{k,i}$, mais nous pourrions certainement calculer ces poids différemment. Les poids pourraient être conditionnels au taux d'occupation des agents, à la taille des files d'attente, si le temps d'attente d'un appel est supérieur au AWT, etc. La formule du poids pourrait être beaucoup plus complexe qu'une fonction linéaire, mais l'implantation du routage en pratique et l'optimisation seront plus difficiles.

6.3 Optimisation du routage

Dans cette section, nous présentons les algorithmes utilisés pour optimiser les paramètres des politiques de routage. L'optimisation du routage est un problème difficile pour plusieurs raisons. La complexité des mécanismes de routage et la nature stochastique du problème nécessitent l'usage de la simulation afin d'avoir des évaluations précises des solutions. Le mélange de paramètres combinatoires et continus, ainsi que les dimensions élevées du problème ajoutent à la difficulté de l'optimisation. Nous ne connaissons aucun algorithme d'optimisation rapide et efficace pour optimiser les politiques de routage avec des fonctions objectifs de type boîte noire.

Tout d'abord, nous avons essayé le programme NOMAD version 3.4 [1, 96] qui implémente l'algorithme "*Mesh adaptive direct search*" (MADS) [14] pour l'optimisation de fonctions boîtes noires. Nous avons utilisé le programme par défaut, sans modification. NOMAD pouvait optimiser des petits problèmes avec peu de paramètres, mais l'algorithme ne trouvait pas de bons résultats pour nos exemples numériques. NOMAD permet à l'utilisateur de combiner un algorithme externe afin de guider l'algorithme MADS. Ceci augmenterait possiblement les chances de NOMAD de trouver de bonnes solutions, mais nous

n'avons pas poursuivi dans cette direction.

Nous choisissons d'adapter et implémenter deux métaheuristiques connues basées sur la simulation : une *descente du gradient* (DG) et un *algorithme génétique modifié* (AGM). Chaque méthode a ses avantages et ses inconvénients. La DG peut être plus rapide pour les petits problèmes, mais elle dépend fortement du point initial et ne peut pas optimiser les problèmes combinatoires. L'AGM est une méthode plus générale que DG. Elle peut optimiser les problèmes combinatoires, entiers ainsi que continus, et elle est moins dépendante du point initial que la DG. Par contre, l'efficacité de l'AGM diminue pour les très grands problèmes, car elle doit générer de grandes populations de solutions (ce qui demande beaucoup de temps de simulation). Nous avons également implémenté une variante de la DG qui est une méthode *quasi-Newton* (QN).

Remarquons que nous ne présentons que la méthode AGM dans notre article [38], car elle est plus simple à implémenter et plus générale, et elle peut optimiser tous les nos exemples numériques. Néanmoins, nous présentons nos implémentations de DG et QN dans cette thèse, sans inclure les résultats numériques. Les résultats de la DG sont souvent compétitifs avec ceux de l'AGM pour les problèmes continus, mais la DG est plus difficile à configurer, car l'algorithme est plus sensible aux paramètres initiaux. Les algorithmes sont décrits en supposant un problème de minimisation. Nous détaillons aussi les modifications et les adaptations apportées à l'AGM pour optimiser les politiques P, PD, PS, PDS, LG$c\mu$ et BP. Notre recherche constitue une première étude numérique sur l'optimisation heuristique du routage pour des modèles de centres d'appels réalistes.

6.3.1 Descente du gradient (DG)

Pour la minimisation des variables continues, nous avons implémenté une méthode de descente bien connue basée sur les gradients [56, 57, 99], appelée aussi la *méthode de la plus forte pente*. À chaque itération, l'algorithme se déplace en fonction du gradient de la solution courante. Le gradient est estimé par différences finies à l'aide de la simulation, avec une longueur de pas qui diminue avec le nombre d'itérations.

Supposons qu'il y a d variables continues à optimiser, $\mathbf{e}_j$ représente un vecteur unitaire avec l'élément 1 à la position j et 0 ailleurs, et f est la fonction objectif à minimiser. Soient

$\mathbf{x} \in \mathbb{R}^d$ la solution courante et $\delta > 0$ la longueur du pas. La j-ième composante du gradient $\mathbf{g}(\mathbf{x})$ de la solution courante peut être approximée par la différence finie centrale :

$$g_j(\mathbf{x}) = \frac{f(\mathbf{x} + \delta \mathbf{e}_j) - f(\mathbf{x} - \delta \mathbf{e}_j)}{2\delta}, \tag{6.9}$$

pour $j = 1, \ldots, d$. Dans notre cas, nous ne pouvons pas calculer exactement la fonction f, mais nous pouvons l'approximer par une fonction $\hat{f}$ obtenue par la simulation. Dans notre algorithme, nous remplaçons la fonction originale f par une estimation $\hat{f}$.

Nous utilisons toujours les variables aléatoires communes (VAC) [9, 98] lors de l'estimation des solutions $\mathbf{x} \in \mathbb{R}^d$ et ce, durant toute l'optimisation. Avec les VAC, toutes les simulations, sur un temps d'horizon identique, sont synchronisées afin de simuler exactement le même historique des clients. Le nombre de clients, leur type, leur temps d'arrivée, leur durée de service (si répondu) et leur durée de patience sont identiques dans toutes les simulations. Les autres flux aléatoires (ou *random streams*), tels que pour les bris d'égalité par le routeur, sont aussi identiques dans toutes les simulations. Notons que le temps de service requis et le temps de patience sont déterminés aléatoirement par le simulateur dès l'arrivée d'un appel, mais ces informations sont cachées à l'optimiseur. Si le temps de service dépend à la fois du type d'appel et du groupe d'agents, alors le simulateur doit générer le temps de service requis pour chaque groupe puisqu'il ne peut pas prédire quel groupe servira l'appel. Utiliser les VAC durant l'optimisation signifie résoudre un problème de moyenne d'échantillons (ou "*sample average problem*"). La solution optimale trouvée pour $\hat{f}$ sera alors biaisée, et elle ne sera pas nécessairement optimale pour f, mais ce biais peut être réduit en augmentant la longueur de la simulation. Nous pourrions utiliser les VAC uniquement pour l'estimation d'un gradient, et générer des suites de nombres différents à chaque itération de la DG, comme dans le chapitre 4, mais nous n'avons pas fait cela. Étant donné que nous sommes principalement intéressés à comparer les performances des différentes politiques de routage, nous devrions appliquer les VAC à travers l'optimisation de toutes les politiques de routage.

Dans notre algorithme DG, la longueur du pas δ est identique pour toutes les variables. Ceci n'est généralement pas un problème si les variables ont des échelles de taille simi-

laires, par exemple si toutes les variables sont des temps de délai. Par contre, lorsque les variables ont différentes échelles de taille, il faudrait appliquer un facteur d'échelle. Prenons par exemple la politique BP, si les variables de temps w_k et v_i sont mesurées par unité d'*heure* comme dans notre implémentation, alors $q_{k,i}$ devrait être beaucoup plus petit que $a_{k,i}$ et $b_{k,i}$. Les paramètres $q_{k,i} = -1$ et $a_{k,i} = b_{k,i} = 1$ donneraient ici un délai d'environ 1 heure ! Ces facteurs d'échelle sont déterminés grossièrement et d'une manière ad hoc. D'après nos expériences, nous choisissons un facteur de 1 pour $q_{k,i}$ et un facteur de 1000 pour $a_{k,i}$ et $b_{k,i}$. D'autre part, si w_k et v_i sont mesurées à la seconde, alors $q_{k,i}$ devrait être plus grand que $a_{k,i}$ et $b_{k,i}$. Nous inversons les facteurs d'échelle dans ce cas.

À chaque itération de la DG, l'algorithme estime le gradient de la solution courante et se déplace vers la prochaine solution en suivant la direction opposée au gradient. La distance à parcourir est déterminée par une recherche linéaire telle la méthode du nombre d'or (ou "*golden section search*"). Exécuter cette recherche nécessite des simulations supplémentaires.

Quand la solution converge vers un minimum local, nous ré-initialisons la longueur du pas δ à une grande valeur, et nous recommençons la méthode DG à partir de la solution courante. La DG est répétée un certain nombre de fois ou jusqu'à ce qu'il n'y ait aucune amélioration de la meilleure solution entre deux exécutions successives de la DG. Le but de relancer la DG est pour éviter les minimums locaux.

L'algorithme 6.1 décrit la méthode DG. Les paramètres d'entrée sont : une solution initiale $\mathbf{x}^{(0)}$, la longueur de pas initiale δ_0, le nombre maximal d'itérations maxIt, et le nombre maximal de démarrages maxDG. Chaque variable peut être restreinte dans un intervalle donné. Une variable est tronquée à sa borne la plus proche si l'algorithme tente de se déplacer à l'extérieur de son intervalle réalisable. Les paramètres $\varepsilon_1, \varepsilon_2$ et ε_3 sont utilisés pour contrôler les critères d'arrêt de l'algorithme. L'opérateur $|\mathbf{g}|$ retourne la longueur euclidienne du vecteur du gradient $\mathbf{g}$.

Méthode quasi-Newton (QN)

Nous avons aussi implémenté une méthode quasi-Newton (QN)[52, 84] qui est une variante de la DG. Au lieu d'utiliser directement le gradient (ou sous-gradient) pour dé-

Entrées : $\mathbf{x}^{(0)}$, δ_0, maxIt, maxDG
Sorties : la meilleure solution trouvée $\mathbf{x}^*$

pour $m = 1$ **à** maxDG **faire**
  $f' \leftarrow \hat{f}(\mathbf{x}^{(0)})$
  pour $n = 1$ **à** maxIt **faire**
    $\delta \leftarrow \delta_0 / n$
    $\mathbf{g}^{(n-1)} \leftarrow$ gradient estimé à $\mathbf{x}^{(n-1)}$ par différences finies avec un pas δ.
    $\bar{\theta} \leftarrow \arg\min_{\theta \geq 0} \hat{f}(\mathbf{x}^{(n-1)} - \theta \mathbf{g}^{(n-1)})$, trouvé avec la méthode du nombre d'or.
    $\mathbf{x}^{(n)} \leftarrow \mathbf{x}^{(n-1)} - \bar{\theta}\mathbf{g}^{(n-1)}$ (Déplacer la solution.)
    si $\hat{f}(\mathbf{x}^{(n)}) < \hat{f}(\mathbf{x}^*)$ **alors**
      $\mathbf{x}^* \leftarrow \mathbf{x}^{(n)}$
    si $\left(\left| \frac{\mathbf{x}_i^{(n)} - \mathbf{x}_i^{(n-1)}}{\mathbf{x}_i^{(n-1)}} \right| < \varepsilon_1$ pour tout i, **et** $|\mathbf{g}| < \varepsilon_2$, **et** $d < \varepsilon_3 \right)$ **alors**
      $n \leftarrow$ maxIt $+1$ (Sortir de la boucle.)
  si $\hat{f}(\mathbf{x}^*) \geq f'$ **alors**
    Terminer. (Si aucune amélioration.)
  $\mathbf{x}^{(0)} \leftarrow \mathbf{x}^*$ (Relancer la méthode.)

Algorithme 6.1 : DESCENTEGRADIENT. La méthode de descente du gradient (DG).

terminer la direction de recherche, le gradient est utilisé pour approximer l'inverse de la matrice hessienne à l'aide de la formule de Broyden-Fletcher-Goldfarb-Shanno (BFGS). Soient $\mathbf{x}^{(n)}$ la solution à l'itération n, $\mathbf{g}^{(n)}$ le gradient de $\hat{f}$ au point $\mathbf{x}^{(n)}$, $\mathbf{Z}^{(n)}$ l'inverse de la matrice hessienne, $\alpha^{(n)} = \mathbf{x}^{(n)} - \mathbf{x}^{(n-1)}$, et $\gamma^{(n)} = \mathbf{g}^{(n)} - \mathbf{g}^{(n-1)}$. La formule BFGS est :

$$\mathbf{Z}^{(n)} = \mathbf{Z}^{(n-1)} + \left(1 + \frac{(\gamma^{(n)})^{\mathrm{T}} \mathbf{Z}^{(n-1)} \gamma^{(n)}}{(\alpha^{(n)})^{\mathrm{T}} \gamma^{(n)}} \right) \frac{\alpha^{(n)} (\alpha^{(n)})^{\mathrm{T}}}{(\alpha^{(n)})^{\mathrm{T}} \gamma^{(n)}} - \frac{\alpha^{(n)} (\gamma^{(n)})^{\mathrm{T}} \mathbf{Z}^{(n-1)} + \mathbf{Z}^{(n-1)} \gamma^{(n)} (\alpha^{(n)})^{\mathrm{T}}}{(\alpha^{(n)})^{\mathrm{T}} \gamma^{(n)}}. \quad (6.10)$$

La direction de recherche à l'itération n est $\mathbf{s}^{(n)} = -\mathbf{Z}^{(n-1)}\mathbf{g}^{(n-1)}$. Dans notre implémentation, $\mathbf{g}^{(n)}$ est une estimation bruitée du gradient de $\hat{f}$ comme dans la méthode DG. Nous utilisons aussi la méthode du nombre d'or pour effectuer la recherche linéaire.

L'algorithme 6.2 présente les modifications apportées à l'algorithme 6.1 afin d'implé-

pour $n = 1$ **à** maxIt **faire**
- ...
- $\mathbf{g}^{(n-1)} \leftarrow$ gradient estimé à $\mathbf{x}^{(n-1)}$ par différences finies avec un pas δ.
- $\mathbf{Z}^{(n-1)} \leftarrow$ mettre à jour l'inverse de la hessienne avec la formule BFGS (6.10).
- $\mathbf{s}^{(n)} \leftarrow -\mathbf{Z}^{(n-1)}\mathbf{g}^{(n-1)}$ (nouvelle direction de recherche)
- $\bar{\theta} \leftarrow \arg\min_{\theta \geq 0} \hat{f}(\mathbf{x}^{(n-1)} + \theta \mathbf{s}^{(n)})$, trouvé avec la méthode du nombre d'or.
- $\mathbf{x}^{(n)} \leftarrow \mathbf{x}^{(n-1)} + \bar{\theta}\mathbf{s}^{(n)}$
- ...

Algorithme 6.2 : QUASINEWTON. La méthode quasi-Newton (QN) est une variante de DG ; remplacer les lignes 5 à 7 de l'algorithme 6.1.

menter la méthode QN. Pour l'initialisation, nous assignons des vecteurs nuls (0 partout) à $\mathbf{x}^{(-1)}$ et $\mathbf{z}^{(-1)}$, et la matrice identité à $\mathbf{Z}^{(-1)} = \mathbf{I}$. La méthode QN ne fonctionne pas toujours mieux que la DG, parce qu'elle est plus sensible au bruit de simulation et à la structure du problème. Nous avons essayé les méthodes DG et QN pour optimiser les poids de la politique BP et pour optimiser les temps de délai des politiques PD et PDS. Nous ne présentons pas les résultats des expérimentations dans la section numérique. Malgré des résultats souvent similaires à ceux de l'AGM, nous préférons la méthode AGM, car elle est plus simple à implémenter, et elle peut optimiser des problèmes plus généraux.

6.3.2 Algorithme génétique modifié (AGM)

Les algorithmes génétiques classiques [64] utilisent habituellement des opérateurs de croisements et de mutations pour générer une nouvelle population de solutions. La définition de ces opérateurs est fortement liée au modèle du problème à optimiser. Dans notre étude, nous choisissons plutôt une variante de l'algorithme génétique, qui opère sur une loi de probabilité paramétrique $\Phi = \Phi(\boldsymbol{\theta})$ sur l'ensemble des solutions du problème. Cette variante optimise et change le vecteur de paramètres $\boldsymbol{\theta}$ de la distribution Φ au lieu d'altérer directement les solutions. Cet algorithme peut être vu comme une version simplifiée des *algorithmes d'estimation de distribution* [94, 106] ou de la méthode d'optimisation *entropie croisée* [25, 49, 116].

L'algorithme génétique modifié (AGM) est décrit dans l'algorithme 6.3. Il commence

Entrées : maxIt, maxItQ, P, $\hat{P}$, ζ, ε, $\boldsymbol{\theta}^{(0)}$ (avec $\hat{P} \leq P$)
Sorties : la meilleure solution trouvée $\mathbf{x}^*$ et son coût estimé f^*

$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}^{(0)}$, $t \leftarrow 0$, $f^* \leftarrow \infty$, $f_{\mathrm{Q}} \leftarrow \infty$
pour $i = 1$ **à** maxIt **faire**
  pour $p = 1$ **à** P **faire**
    Générer une solution $\mathbf{x}^{(p)}$ selon la loi de probabilité $\Phi(\boldsymbol{\theta})$.
    $f^{(p)} \leftarrow \hat{f}(\mathbf{x}^{(p)})$, coût estimé par simulation.
  Trier et indexer $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(P)}$ par ordre croissant des coûts $f^{(1)}, \ldots, f^{(P)}$.
  si $f^{(1)} < f^*$ **alors**
    $\mathbf{x}^* \leftarrow \mathbf{x}^{(1)}$ **et** $f^* \leftarrow f^{(1)}$
  si $f^{(\hat{P})} < f_{\mathrm{Q}}$ **alors**
    $f_{\mathrm{Q}} \leftarrow f^{(\hat{P})}$ **et** $t \leftarrow 0$
  sinon
    $t \leftarrow t + 1$
  Calculer l'estimateur $\tilde{\boldsymbol{\theta}}$ (par maximum de vraisemblance par exemple) à partir de l'ensemble des solutions élites $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(\hat{P})}$.
  $\boldsymbol{\theta} \leftarrow \zeta \tilde{\boldsymbol{\theta}} + (1 - \zeta)\boldsymbol{\theta}$
  si la variance de $\Phi(\boldsymbol{\theta})$ est suffisamment petite, disons $< \varepsilon$, **alors**
    Terminer.
  si $t =$ maxItQ **alors**
    Terminer.

Algorithme 6.3 : ALGOGÉNÉTIQUE. Algorithme génétique modifié.

avec un vecteur de paramètres initiaux $\boldsymbol{\theta} = \boldsymbol{\theta}^{(0)}$ pour la loi de probabilité Φ. À chaque itération, il génère aléatoirement une *population* de P solutions admissibles à partir de $\Phi(\boldsymbol{\theta})$. Il estime les coûts de ces P solutions par simulation, et garde les $\hat{P}$ solutions ayant les plus petits coûts, avec $\hat{P} \leq P$. Ces $\hat{P}$ solutions forment la *population élite*. Puis, l'algorithme estime un nouveau vecteur de paramètres $\tilde{\boldsymbol{\theta}}$ à l'aide de la méthode du maximum de vraisemblance sur la population élite. Le vecteur $\boldsymbol{\theta}$ est ensuite mis à jour par une combinaison convexe de sa valeur précédente et du vecteur $\tilde{\boldsymbol{\theta}}$, avec un paramètre de lissage $\zeta \in [0,1]$. Nous utilisons $\zeta = 0.5$ dans nos expériences numériques. Une nouvelle population de P solutions est générée à partir de la nouvelle loi de probabilité $\Phi(\boldsymbol{\theta})$ à l'itération suivante, et ainsi de suite. L'idée de cette méthode est qu'en calibrant le vecteur de paramètres $\boldsymbol{\theta}$ en

fonction des solutions élites à chaque itération, la densité (ou masse) de probabilité de $\Phi(\boldsymbol{\theta})$ devrait converger progressivement autour d'une solution optimale. L'algorithme s'arrête si (1) la trace (ou l'élément maximal) de la matrice de covariance de $\Phi(\boldsymbol{\theta})$ est inférieure à ε, (2) le nombre maximal d'itérations maxIt est atteint, ou (3) le coût du $(\hat{P}/P)$-quantile, qui correspond au plus grand coût dans la population élite, n'a pas diminué depuis les maxItQ dernières itérations.

Pour simplifier l'algorithme et son implémentation, nous choisissons la fonction de densité de $\Phi(\boldsymbol{\theta})$ comme le produit des densités de d lois normales, où d est le nombre de paramètres de routage. Ainsi, chaque paramètre de routage est représenté par une loi normale indépendante. Le vecteur $\boldsymbol{\theta}$ a une taille $2d$, puisque chaque loi normale possède deux paramètres (la moyenne et la variance). Nous aurions pu choisir une loi de probabilité multivariée plus générale, mais les paramètres de cette loi seraient plus difficiles à estimer. Toutefois, nous avons expérimenté une loi normale multivariée pour optimiser le routage BP. Les résultats montrent une légère amélioration pour les petits problèmes (avec peu de paramètres), mais de pires performances pour les grands problèmes.

Le vecteur initial $\boldsymbol{\theta}^{(0)}$ est choisi tel que la densité de probabilité initiale serait assez large pour couvrir (hypothétiquement) une région de solutions contenant une solution optimale. Grossièrement, la loi de probabilité initiale possède une grande variance, et cette variance décroît généralement à chaque itération.

Nous optimisons une fonction d'échantillons $\hat{f}$ au lieu de f, avec les VAC, similairement aux méthodes DG et QN. Comme nous utilisons la simulation pour estimer les coûts et que l'AGM est une méthode stochastique, nous nous attendons à des solutions bruitées. Dans la section numérique, nous optimisons les politiques de routage sur plusieurs fonctions d'échantillons, puis nous comparons leurs performances sur l'ensemble de ces échantillons.

Les choix des tailles de populations P et $\hat{P}$ ont aussi un impact sur la vitesse de convergence et sur la qualité de la solution finale. Quand P est grand et $\hat{P}$ est petit, l'AGM converge habituellement plus rapidement vers un minimum local. Par contre, si $\hat{P}$ est relativement grand, alors la convergence sera plus lente, mais ceci peut mener vers une meilleure solution. Nous suggérons le choix $\hat{P} = P/10$ ou plus petit si le problème est très grand.

Quelques heuristiques peuvent être appliquées à l'AGM afin d'augmenter son efficacité. L'AGM peut converger lentement vers un minimum local, comme il n'est pas une méthode de recherche directe. Pour augmenter la vitesse de convergence, une heuristique très simple est d'augmenter la pondération de la meilleure solution élite dans la population élite en dupliquant cette solution un certain nombre de fois. Ceci augmente souvent la vitesse de convergence vers un minimum local, ce qui peut être utile quand le budget d'exécution est court. Nous ajoutons $\lceil 0.3\hat{P} \rceil - 1$ copies de la meilleure solution élite dans nos expériences numériques.

En se basant sur nos expérimentations, nous recommandons un bon équilibre entre la taille de la population P et le nombre maximal d'itérations maxIt. Un grand P augmente les chances de générer une bonne solution, mais exécuter plus d'itérations permet de générer des solutions à partir de lois de probabilité plus raffinées.

Par ailleurs, il y a souvent des solutions qui peuvent être éliminées rapidement sans exécuter toute la simulation. Dans notre implémentation, nous exécutons ce filtrage en évaluant premièrement toutes les P solutions par de courtes simulations, disons avec r_0 réplications de simulation. Puis, nous gardons les $P' = \min\{2\hat{P}, P\}$ meilleures solutions pour la simulation complète, disons avec $r_1 > r_0$ réplications. Nous sélectionnons ensuite les $\hat{P}$ meilleures solutions parmi les P' solutions pour former la population élite. Il existe des *méthodes de classement et de sélection* plus sophistiquées [22, 65], mais comme une sélection optimale n'est pas cruciale à chaque itération de l'AGM, ces méthodes plus complexes ne sont pas nécessairement utiles.

6.3.3 Optimisation des politiques de routage

Nous expliquons à présent comment optimiser plus en détails les différentes politiques de routage avec l'AGM.

Pour la politique BP, nous utilisons une loi normale pour les paramètres continus non bornés, et une loi normale tronquée (remplacer les valeurs négatives par 0) lorsque les paramètres sont contraints à être non négatifs. L'estimateur du maximum de vraisemblance d'une loi normale est facile à calculer. Chaque paramètre de poids a deux éléments dans $\boldsymbol{\theta}$: la moyenne et la variance de sa loi normale. Pour les paramètres initiaux $\boldsymbol{\theta}^{(0)}$, nous

commençons avec des moyennes de 0 et de grandes variances pour tous les paramètres. Selon l'unité de temps utilisée pour les variables w_k et v_i, choisir de bonnes échelles de grandeur pour les $q_{k,i}, a_{k,i}$ et $b_{k,i}$ améliore les solutions de départ. Par exemple, si w_k et v_i sont mesurées en unité seconde, alors nous choisissons les variances initiales des $q_{k,i}$ à être 100 fois plus grande que celles de $a_{k,i}$ et $b_{k,i}$. Par contre, si le temps est mesuré en unité d'heure, alors nous choisissons les variances des $q_{k,i}$ à être 100 fois plus petites.

Pour BP-N, toute solution ayant une paire (k', i') telle que $a_{k',i'} < 0$ et $b_{k',i'} < 0$ est considérée inadmissible, et une nouvelle solution est générée pour la remplacer. À chaque itération de l'AGM, des solutions sont générées aléatoirement selon la loi de probabilité $\Phi(\boldsymbol{\theta})$ jusqu'à l'obtention de P solutions admissibles. Cette procédure est particulièrement utile lors des premières itérations de l'algorithme, où plusieurs mauvaises solutions candidates sont rejetées sans être simulées.

Nous utilisons aussi la loi normale tronquée pour générer les paramètres de la politique LG$c\mu$, les temps de délai de PD et PDS, et les seuils d'agents libres de PS et PDS. Les valeurs négatives générées pour les temps de délai et les seuils d'agents sont remplacées par 0. Pour ces politiques de routage, nous attribuons de petits nombres positifs aux paramètres initiaux des moyennes et variances. Dans nos exemples numériques, nous fixons les moyennes initiales à 10 secondes de délai et un agent pour les seuils, et les mêmes valeurs pour les *écarts types* initiaux.

Optimiser les listes de priorités est plus complexe avec l'AGM, car ceci est un problème combinatoire. Les listes de priorités sont définies dans la description de la politique P à la section 2.2.2, page 18. Pour construire une solution, nous générons indépendamment les listes de priorités type-vers-groupe et groupe-vers-type. Nous décrivons à présent comment générer la liste de priorités groupe-vers-type $\mathcal{L}_i$ du groupe i avec l'ensemble d'habiletés $\mathcal{S}_i$. La procédure pour générer la liste de priorités type-vers-groupe $\mathcal{G}_k$ du type d'appel k est similaire. Notons que même si un centre d'appels est très grand (avec beaucoup de listes de priorités), ces listes sont souvent courtes.

Soit $h_i = |\mathcal{S}_i|$ le nombre d'habiletés du groupe i. Si $h_i = 1$ (cas trivial), alors la liste de priorités contient un seul élément, et il n'y a aucune optimisation à faire. Supposons maintenant que $h_i > 1$. La loi de probabilité utilisée pour générer $\mathcal{L}_i$ possède les paramètres

$\alpha_{i,k} > 0$ pour $k \in \mathcal{S}_i$ et $\beta_{i,n} \in [0.05, 0.95]$ pour $n = 1, 2, \ldots, h_i - 1$. Nous initialisons $\alpha_{i,k} = 1$ pour tout k et $\beta_{i,n} = 0.5$ pour tout n. La liste de priorités $\mathcal{L}_i$ est générée en deux phases indépendantes :

Phase 1 : Nous générons au hasard une permutation $\mathcal{P} = (p_1, p_2, \ldots, p_{h_i})$ telle que $p_j \in \mathcal{S}_i$ pour tout j, et $p_j \neq p_l$ pour $j \neq l$. Pour obtenir ceci, nous générons une variable aléatoire $z_{i,k}$ distribuée uniformément sur l'intervalle $[0, \alpha_{i,k}]$ pour chaque $k \in \mathcal{S}_i$, et la permutation $\mathcal{P}$ est déterminée en triant ces $z_{i,k}$ par ordre croissant. Remarquons que les $h_i!$ permutations possibles n'ont pas toutes les probabilités égales. Les types d'appels k avec de petits $\alpha_{i,k}$ ont plus de chances d'être placés au début de la liste et d'avoir une plus grande priorité.

Phase 2 : À partir de la permutation obtenue $\mathcal{P}$, nous devons maintenant décider des sous-ensembles $\mathcal{L}_i^{(\cdot)}$ de types d'appels ayant les priorités égales. C'est-à-dire que nous devons choisir entre une priorité *égale* ou *plus grande que* pour chaque relation de priorité entre deux positions successives dans la permutation $\mathcal{P}$. Il y a $h_i - 1$ relations à déterminer, et la n-ième relation est choisie *égale* avec probabilité $\beta_{i,n}$, pour $n = 1, 2, \ldots, h_i - 1$. C'est-à-dire que les types d'appels p_n et p_{n+1} ont la même priorité avec probabilité $\beta_{i,n}$, autrement le type p_n a une plus grande priorité. Les paramètres $\beta_{i,n}$ sont bornés entre 0.05 et 0.95 afin de permettre un minimum de diversification. Mentionnons que pour simplifier la procédure, ces choix de priorités ne dépendent que des positions des éléments dans la liste, et non des types d'appels. La phase 2 est donc indépendante de la phase 1.

Les paramètres de distribution $\alpha_{i,k}$ et $\beta_{i,n}$ sont mis à jour à chaque itération comme suit. Pour $\alpha_{i,k}$, nous prenons simplement la moyenne des rangs de priorités du type d'appel k dans la liste de priorités $\mathcal{L}_i$ du groupe i parmi les $\hat{P}$ solutions élites. Les types d'appels ayant des priorités égales possèdent le même rang. Les types ayant la plus haute priorité ont le rang 1, ceux ayant la priorité suivante ont le rang 2, et ainsi de suite. Pour mettre à jour $\beta_{i,n}$, nous calculons la proportion de types d'appels à la position n ayant la même priorité que les types d'appels à la position $n+1$ (lorsque $\mathcal{L}_i$ est exprimée sous la forme d'un vecteur comme à la phase 2) parmi les $\hat{P}$ solutions élites. Pour le critère d'arrêt basé

sur la variance de $\Phi(\boldsymbol{\theta})$ à la ligne 15 de l'algorithme 6.3, nous considérons uniquement la variance des rangs de priorités $\alpha_{i,k}$ parmi les solutions élites.

Pour la politique PD, nous optimisons les listes de priorités et les temps de délai simultanément quand le centre d'appels est petit. Pour des centres d'appels plus grands et un budget de temps limité, nous trouvons qu'une approche à deux étapes est plus efficace, malgré qu'elle soit sous-optimale. Tout d'abord, nous optimisons seulement les listes de priorités, puis nous optimisons les temps de délai en gardant les listes de priorités fixes.

Nous procédons de la même manière pour optimiser la politique PS. Nous optimisons simultanément les priorités et les seuils d'agents libres lorsque le centre d'appels est petit. Pour des centres plus grands, nous optimisons séparément en commençant par les listes de priorités, puis les seuils d'agents après.

Quand nous combinons les listes de priorités, les temps de délai et les seuils d'agents libres dans la politique PDS, le nombre de paramètres augmente rapidement, et il devient très difficile d'optimiser tous ces paramètres parallèlement. Nous proposons une approche multi-étapes qui exécute l'AGM plusieurs fois. À chaque étape, l'AGM optimise juste un sous-ensemble des paramètres. Nous débutons toujours par optimiser uniquement les listes de priorités, car, d'après nos expérimentations, les priorités ont tendance à avoir un plus grand impact sur les mesures de performance. Pour les autres itérations, jusqu'à un nombre maximal itAGM, nous exécutons l'AGM pour optimiser le sous-ensemble de paramètres choisi aléatoirement et uniformément parmi :

1. les listes de priorités,
2. les listes de priorités et les temps de délai,
3. les listes de priorités et les seuils d'agents libres,
4. les temps de délai.

Nous utilisons une liste tabou [62, 63] pour conserver les sous-ensembles de paramètres qui n'ont donné aucune amélioration, et ces sous-ensembles ne peuvent pas être choisis tant et aussi longtemps qu'ils sont présents dans cette liste. Cette version multi-étapes demande encore plus de temps d'exécution, parce que l'AGM doit être exécuté plusieurs fois. En tenant compte de ce désavantage, nous allouons un plus grand budget de temps pour

optimiser la politique PDS dans nos exemples numériques.

6.4 Exemples numériques

Nous comparons les performances des diverses politiques de routage, premièrement, sur des modèles canoniques simples avec 2 ou 3 types d'appels et 2 groupes d'agents, puis un grand modèle avec 8 types et 10 groupes, et finalement un modèle composé de 6 types et 3 groupes généralistes. Les modèles canoniques considérés sont les modèles N et W, présentés à la figure 6.1 (tirée de la figure 2.1, page 16). Rappelons que les exemples ne considèrent qu'une période d'une journée.

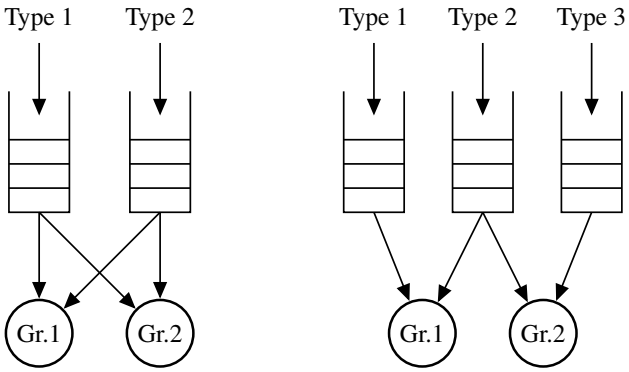


Figure 6.1 – Modèles canoniques de centres d'appels : X et W. Les arcs définissent les habiletés des groupes d'agents.

Nous avons expérimenté différents modèles de processus d'arrivée et de distributions des temps de service. Pour chaque type d'appel k, le processus d'arrivée est soit Poisson avec un taux fixe λ_k, ou bien Poisson-gamma tel que le processus d'arrivée d'une période (ou une journée) est Poisson, mais le taux est une variable aléatoire gamma. Le processus d'arrivée du type k est indépendant des processus d'arrivée des autres types d'appels. La loi de probabilité des temps de service pour le type d'appel k par un agent du groupe i est soit une loi exponentielle de moyenne $\mu_{k,i}^{-1}$, ou bien une loi log-normale de moyenne $\phi_{k,i}$ et d'écart type $\omega_{k,i}$ (les paramètres d'échelle et de forme peuvent être calculés à partir de $\phi_{k,i}$ et $\omega_{k,i}$). Le temps de patience d'un appel du type k est une variable exponentielle de moyenne ν_k^{-1}. Toutes ces variables sont indépendantes.

Dans nos exemples, toutes les valeurs des taux et des moyennes sont exprimées en *unité minute*, à moins d'une mention contraire. Afin d'éviter de rapporter de très petits résultats, les mesures de performance $f_{\mathrm{S},k}, f_{\mathrm{A},k}$ et $f_{\mathrm{O},i}$ (définies à la section 2.3, page 20) sont calculées en *pourcentage*, et les temps d'attente moyens $f_{W,k}$ sont mesurés en *secondes* dans nos formules. Pour rapporter les mesures de performance d'une solution, nous définissons les vecteurs suivants : $\mathbf{S} = (f_{\mathrm{S},1}, \ldots, f_{\mathrm{S},K}), \mathbf{A} = (f_{\mathrm{A},1}, \ldots, f_{\mathrm{A},K}), \mathbf{W} = (f_{\mathrm{W},1}, \ldots, f_{\mathrm{W},K})$ et $\mathbf{O} = (f_{\mathrm{O},1}, \ldots, f_{\mathrm{O},I})$. Dans le simulateur, toutes les mesures de temps (temps d'attente, temps d'inoccupation des agents, temps de délai, etc.) sont comptés en *secondes* pour toutes les politiques de routage.

En pratique, il n'existe aucune méthode systématique pour l'optimiser les mesures de performance des centres d'appels multi-compétences. C'est-à-dire qu'il n'y a aucun procédé établi pour comparer différents vecteurs de performance. Habituellement, un gestionnaire préfère avoir plusieurs types d'appels légèrement sous leurs cibles de performance que d'avoir un type qui soit largement sous sa cible, tandis que les autres types atteignent leurs cibles. Pour cette raison, nous choisissons de pénaliser les solutions par des fonctions quadratiques. Nous considérons une variété de fonctions objectifs différentes :

$$F_{\mathrm{S}}(\pi) = \sum_{k=1}^{K} \max(l_k - f_{S,k}(\pi, \tau_k), 0)^2, \tag{6.11}$$

$$F_{\mathrm{S}}^1(\pi) = 3\max(l_1 - f_{S,1}(\pi, \tau_1), 0)^2 + \sum_{k=2}^{K} \max(l_k - f_{S,k}(\pi, \tau_k), 0)^2, \tag{6.12}$$

$$F_{\mathrm{S}}^2(\pi) = 10\max(l_1 - f_{S,1}(\pi, \tau_1), 0)^2 + \sum_{k=2}^{K} \max(l_k - f_{S,k}(\pi, \tau_k), 0)^2, \tag{6.13}$$

$$F_{\mathrm{SA}}(\pi) = \sum_{k=1}^{K} \left(\max(l_k - f_{S,k}(\pi, \tau_k), 0)^2 + f_{A,k}(\pi)^2\right), \tag{6.14}$$

$$F_{\mathrm{SA}}^{\lambda}(\pi) = \sum_{k=1}^{K} \lambda_k \left(\max(l_k - f_{S,k}(\pi, \tau_k), 0)^2 + f_{A,k}(\pi)^2\right), \tag{6.15}$$

$$F_{\mathrm{AW}}(\pi) = \sum_{k=1}^{K} \left(f_{A,k}(\pi)^3 + f_{W,k}(\pi)^2\right), \tag{6.16}$$

$$F_{\mathrm{SO}}(\pi) = \sum_{k=1}^{K} \max(l_k - f_{S,k}(\pi, \tau_k), 0)^2 + 5 \sum_{i=1}^{I} \left|f_{O,i}(\pi) - \bar{O}\right|^2. \tag{6.17}$$

Dans les fonctions F_{S}^1 et F_{S}^2, le type d'appel 1 a plus d'importance que les autres types avec les coefficients de poids $c_{\mathrm{S},1}$ qui sont 3 et 10 fois plus grands, respectivement. Dans $F_{\mathrm{SA}}^{\lambda}$, les types d'appels sont pénalisés proportionnellement à leur taux d'arrivée. Similairement, nous aurions pu pénaliser l'occupation des agents dans F_{SO} proportionnellement aux nombres d'agents, en fixant les poids $c_{\mathrm{O},i} = y_i$. Dans la pratique, ces poids sont choisis par la direction du centre d'appels. Un inconvénient possible de la fonction $F_{\mathrm{SA}}^{\lambda}$ est que les types d'appels ayant de très faibles volumes risquent d'avoir aucune influence significative dans la fonction objectif, et d'être négligés par conséquent. Un compromis serait d'avoir des coûts de pénalités partiellement proportionnels aux volumes d'appels ou aux nombres d'agents. Dans F_{AW}, nous choisissons un plus grand exposant pour les abandons $f_{\mathrm{A},k}$, parce que les valeurs des temps d'attente (mesurés en secondes) augmentent plus rapidement. Nous avons aussi expérimenté des fonctions objectifs avec de plus grand poids sur les abandons, incluant une fonction F_{A} qui pénalise uniquement les abandons. Dans ces expériences, nous observons que les différences entre les politiques BP et LG$c\mu$ sont beaucoup plus petites, et les solutions de routage imposent rarement des temps de délai.

Rappelons que tous les résultats d'optimisation rapportés dans cette section proviennent de la méthode AGM. Nous avons testé les méthodes DG et QN, mais l'AGM est beaucoup plus versatile, plus facile à implémenter et à initialiser. Pour les problèmes où DG et QN sont applicables, les résultats d'optimisation des trois méthodes sont souvent similaires.

Comme nous désirons principalement comparer les performances des différentes politiques de routage, nous simulons toutes les solutions de routage de toutes les politiques avec les VAC telles que décrites à la section 6.3.1, page 169.

Nous approximons la fonction objectif en simulant n réplications indépendantes de T heures d'opération du centre d'appels. La longueur totale d'une simulation nT est choisie relativement courte afin de limiter le temps d'exécution de l'optimisation. La fonction d'échantillons $\hat{f}$ peut alors être fortement bruitée. Pour avoir des résultats représentatifs, nous répétons indépendamment l'algorithme d'optimisation r fois ; c'est-à-dire que nous optimisons r fonctions d'échantillons $\hat{f}$ indépendantes. Nous choisissons $r = 5$ pour les modèles canoniques X et W, et $r = 3$ pour les exemples plus grands. À la fin, les r solutions sont simulées hors-échantillon avec $100n$ (100 fois plus) nouvelles réplications indé-

pendantes, mais avec les VAC pour toutes les politiques, et nous comparons ces résultats. Ces évaluations hors-échantillon donnent des estimations avec des largeurs d'intervalles de confiance à 95% inférieures à 1% pour la majorité des mesures de performance. Les évaluations des solutions finales contiennent alors très peu de bruit de simulation (mais les solutions peuvent contenir du bruit d'optimisation).

Les exemples ont été choisis tels qu'il serait difficile d'obtenir une note parfaite de 0 pour les fonctions objectifs. Les paramètres de l'AGM sont donnés dans la description de chaque exemple. Les expériences numériques ont été exécutées sur un processeur Intel Xeon 2.40Ghz. Pour les résultats rapportés dans cette section, les temps moyens d'exécution de l'AGM sont environ 2 heures pour le modèle X, 4 heures pour le modèle W, 14 heures pour le grand exemple et 4 heures pour l'exemple moyen avec 6 types d'appels. Nous ne rapportons pas les temps d'exécution en détails, car notre but n'est pas d'étudier la vitesse des algorithmes d'optimisation, mais de comparer les politiques avec leurs paramètres (supposément) optimisés. Par ailleurs, nous avons répété les expérimentations avec la moitié du nombre d'itérations de l'AGM, et les résultats obtenus étaient presque aussi bons. Pour donner une idée de la vitesse d'optimisation de l'AGM pour la politique BP, la première itération de l'AGM trouve habituellement un coût qui est 2 fois plus grand que le coût final pour les exemples canoniques, et au moins 5 fois plus grand pour les deux exemples plus grands. À la moitié du nombre d'itérations, la politique BP est généralement égale ou meilleure que les autres politiques, et le coût hors-échantillon est presque toujours à moins de 5% du coût final, et souvent à moins de 1% pour les petits exemples. Cet écart dépend ordinairement du nombre de paramètres à optimiser et de la taille de l'espace de solutions. La politique BP-N, où tous les paramètres peuvent être négatifs, requiert souvent plus d'itérations. Nous montrons quelques exemples de la convergence de l'AGM à la section 6.4.6.

L'AGM peut être facilement adapté pour le traitement en parallèle, mais nous n'avons pas fait ceci. La méthode d'optimisation peut être améliorée davantage, par exemple en effectuant une recherche locale autour de la solution finale. Comme dans les chapitres précédents, toutes les simulations sont exécutées à l'aide de la librairie de simulation ContactCenters en Java de Buist et L'Ecuyer [31].

Nous rapportons les résultats des exemples les plus intéressants. Nous avons expérimenté plusieurs autres centres d'appels et fonctions objectifs qui ne sont pas inclus ici. En particulier, nous avons expérimenté les modèles V, N et M (voir la figure 2.1 à la page 16), ainsi qu'un centre d'appels avec 3 types et 6 groupes d'agents. Il y a aussi des exemples (que nous jugeons moins intéressants) où la plupart des politiques, incluant BP, LG$c\mu$ et PDS, obtiennent des performances très similaires. Ces exemples sont généralement ceux qui modélisent un régime à trafic intense et qui pénalisent seulement les abandons ou les temps d'attente moyens.

6.4.1 Exemple d'un modèle X

Nous considérons un modèle X avec deux types d'appels et deux groupes d'agents où chaque groupe peut servir les tous les types d'appels, voir la figure 6.1. Les ensembles d'habiletés des agents sont $\mathcal{S}_1 = \mathcal{S}_2 = \{1,2\}$. En pratique, chaque groupe possède souvent une habileté primaire et une habileté secondaire, et les agents serviraient plus rapidement les appels correspondant à leur habileté primaire. Nous considérons des processus d'arrivée Poisson avec les taux $\lambda_1 = 18$ et $\lambda_2 = 1.8$, des durées de service exponentielles avec les taux $\mu_{1,1} = 0.198, \mu_{1,2} = 0.18, \mu_{2,1} = 0.162$ et $\mu_{2,2} = 0.18$, et des durées de patience exponentielles avec les taux $\nu_1 = 0.12$ et $\nu_2 = 0.24$. Les nombres d'agents sont $y_1 = 90$ et $y_2 = 14$, pour un total de 104 agents. Les seuils de SL sont $l_k = 80\%$ avec des AWT de $\tau_k = 20$ secondes pour les deux types d'appels. Les paramètres de simulation sont $T = 100$ heures et $n = 6$.

Nous exécutons l'AGM avec les paramètres $P = 100, \hat{P} = 10$ et $\mathsf{maxIt} = 30$. Pour la politique PDS, comme le processus exécute plusieurs fois l'AGM, nous réduisons maxIt à 10 et choisissons $\mathsf{itAGM} = 10$, pour un total de 100 itérations de l'AGM pour chacune des r fonctions d'échantillons. Les temps d'optimisation des autres politiques sont similaires entre eux (car l'AGM évalue exactement le même nombre de solutions), mais optimiser PDS demande plus de temps à cause de la procédure multi-étapes. Pour toutes les politiques, nous choisissons $\mathsf{maxitQ} = 5, \zeta = 0.5$, et ε tellement petit qu'il n'arrête jamais l'AGM, à l'exception de la politique P quand toute la population élite est constituée de la même solution.

Politique	F_S	F_S^1	F_S^2	F_{SA}	F_{SA}^λ	F_{AW}	F_{SO}
E	143.2	300.3	846.6	181.0	28.73	575	143.3
P	21.3	35.6	84.9	50.7	5.05	269	36.9
PD	8.3	21.5	35.0	39.9	4.42	254	40.8
PS	13.2	37.8	77.1	42.5	5.09	239	20.0
PDS	**7.6**	23.2	41.4	36.5	**4.35**	241	**18.7**
LG$c\mu$	44.5	59.6	86.3	88.1	8.23	276	63.7
BP	8.8	22.8	**27.3**	38.2	4.46	245	19.1
BP-S	8.2	**20.1**	28.0	38.9	4.51	246	19.4
BP-S2	44.0	59.3	84.6	108.8	8.59	310	58.6
BP-A	8.8	23.8	62.6	**36.3**	4.68	**236**	21.4
BP-AS	13.5	31.2	50.8	42.9	4.71	237	18.9
BP-N	10.3	**14.6**	**3.9**	39.4	**4.23**	245	**17.7**

Tableau 6.I – Exemple X : coût moyen des 5 solutions évaluées hors-échantillon.

Le tableau 6.I présente les coûts moyens des politiques de routage et des fonctions objectifs sélectionnées, basés sur les $r = 5$ solutions évaluées hors-échantillon. Le meilleur coût moyen pour chaque fonction objectif est coloré en rouge. La variante BP-N avec des coefficients négatifs trouve parfois des meilleurs coûts (colorés en bleu). La politique BP-N n'est pas réaliste, comme mentionnée à la section 6.2.4, mais nous l'incluons dans nos tests à des fins de comparaisons seulement. En résumé, les politiques conservatrices de travail (E, P et LG$c\mu$) réussissent moins bien. La politique E (toutes les priorités égales) obtient les pires coûts de pénalités, suivie des politiques LG$c\mu$, P et BP-S2.

Les écarts de coûts entre les politiques sont plus petits quand nous pénalisons le nombre d'abandons et les temps d'attente moyens avec F_{AW}, ou lorsque les coûts de pénalités sont proportionnels aux taux d'arrivée avec F_{SA}^λ. Parmi les politiques BP, la variante simplifiée BP-S2 donne des performances décevantes, alors que les autres versions de BP performent généralement bien dans tous les cas, à part BP-AS qui marche moins bien pour les objectifs F_S, F_S^1, F_S^2 et F_{SA}. La politique PS obtient des coûts similaires à BP-AS. Les politiques PD et PS marchent bien pour certaines fonctions objectifs, alors que BP, BP-S et PDS performent bien pour toutes les fonctions objectifs.

Remarquons que notre algorithme d'optimisation AGM n'est pas parfait. Par exemple, les coûts de BP-N et BP ne devraient pas être plus grand que le coût de BP-S pour l'objectif

F_S, alors que ce n'est pas le cas dans nos résultats. Ceci s'explique par le fait que la variante BP-S a moins de variables et qu'elle est plus facile à optimiser. Parfois, l'erreur vient du bruit de la simulation. Par exemple, le coût moyen des 5 fonctions d'échantillons de BP-S évalué durant l'optimisation est plus petit que celui de PDS : soit 7.02 pour BP-S contre 7.18 pour PDS. La politique BP-S est donc meilleure que PDS selon l'AGM. Cependant, nous obtenons le résultat contraire quand nous évaluons les solutions par de longues simulations hors-échantillon. La politique PDS est meilleure que BP-S : soit un coût de 7.59 pour PDS contre 8.17 pour BP-S.

Nous donnons à présent quelques exemples sur les fonctionnements de certaines politiques de routage. Parmi les $r = 5$ solutions finales, nous considérons toujours la meilleure solution. Nous comparons tout d'abord les solutions de routage pour la fonction objectif F_S. Pour la politique P, le routage donne la priorité aux paires (groupe 1, type 1) et (groupe 2, type 2) lors de la sélection d'un agent ou d'un appel. Remarquons que ceci n'est pas équivalent à la règle du *plus-rapide-serveur-en-premier* (ou "*fastest-server-first*"), car le groupe 2 répond aux deux types d'appels à la même vitesse ($\mu_{1,2} = \mu_{2,2}$). Pour PD, la meilleure solution possède les mêmes listes de priorités groupe-vers-type que pour la politique P, et elle impose des temps de délai de $d_{1,2} = 16.1$ et $d_{2,1} = 2.7$ secondes. Notons que si $d_{k,i} > \tau_k = 20$, alors tous les appels de type k répondus par les agents du groupe i auront reçu un "mauvais service". Les mesures de performance des solutions discutées ici sont présentées dans le tableau 6.IV. La politique PD a plus d'abandons au total que P, parce que le volume du type d'appel 1 est 10 fois plus élevé que le volume du type 2. Donc, il nécessite moins de ressources (et d'agents) pour améliorer le type 2. Pour PS, les listes de priorités type-vers-groupe sont identiques à celles de P, les listes de priorités groupe-vers-type sont $\mathcal{L}_1 = (\{2\},\{1\})$ et $\mathcal{L}_2 = (\{1\},\{2\})$, et les seuils d'agents libres sont $m_{1,2} = 0.95, m_{2,1} = 1.62$ et $m_{2,2} = 0.09$. La petite valeur positive de $m_{2,2}$ est un artéfact de la convergence "incomplète" de l'optimisation AGM. Le coût de la solution est réduit par moins de 0.2 quand nous remplaçons $m_{2,2}$ par 0. La solution pour PDS est un mélange : les listes de priorités sont identiques à celles de P, les temps de délai imposés sont $d_{1,2} = 14.7$ et $d_{2,1} = 7.8$ secondes, et les seuils d'agents libres sont $m_{1,1} = 0.63, m_{1,2} = 0.80, m_{2,1} = 1.64$ et $m_{2,2} = 0.004$. Le seuil $m_{2,2}$ vaut essentiellement 0, mais pas le seuil $m_{1,1}$. Si nous ré-

duisons $m_{1,1}$ de 0.63 à 0, alors le coût passera de 4.9 à 8.5. Avec $m_{1,1} = 0.63$, le nombre d'abandons est 0.05% plus élevé et le temps d'attente moyen est 0.2 seconde plus grand, mais $f_{S,1}$ est 0.8% plus grand et $f_{S,2}$ est 0.2% plus petit. La perte du SL du type 2 est facilement compensée par le gain supérieur du SL du type 1.

Objectif	a_1	b_1	a_2	b_2	e_1	f_1	e_2	f_2
F_S	538	101	2418	0	1116	152	991	72.1
F_S^1	304	165	2759	0.244	2328	1.38	2234	6.59
F_S^2	476	203	2808	0	2088	27.3	1486	29.9
F_{SA}	1396	27.0	990	124	2735	0	2762	3.00
F_{SA}^λ	147	201	2824	0.626	1590	8.19	1512	1.79
F_{SO}	1220	142	3844	0.190	855	132	1244	107

Tableau 6.II – Exemple X : la meilleure des 5 solutions pour la politique LG$c\mu$ et les fonctions objectifs choisies.

Polit.	Obj.	$q_{1,1}$	$q_{1,2}$	$q_{2,1}$	$q_{2,2}$	$a_{1,1}$	$a_{1,2}$	$a_{2,1}$	$a_{2,2}$	$b_{1,1}$	$b_{1,2}$	$b_{2,1}$	$b_{2,2}$
BP	F_S	1129	−116	384	1213	19.3	6.48	0.95	0.83	1.25	0.79	3.68	0
BP	F_S^1	1844	−967	231	2468	106	58.5	4.81	0	44.9	3.00	39.6	58.9
BP	F_S^2	1205	−1970	426	451	66.1	99.8	1.33	0	15.4	2.49	66.4	126
BP	F_{SA}	2153	−404	−135	2337	18.3	22.4	53.8	110	62.4	4.18	20.0	50.6
BP	F_{SA}^λ	2405	−549	−350	943	172	42.0	20.3	302	31.2	7.30	0	81.3
BP	F_{AW}	404	−400	461	3066	18.5	30.5	63.4	6.80	169	10.1	4.60	64.5
BP	F_{SO}	3250	−975	188	1600	243	1.98	18.5	72.1	27.9	111	7.61	78.5
BP-S	F_S	2195	−232	355	1456	12.6	12.6	5.56	5.56	6.20	1.35	6.20	1.35
BP-S	F_{SA}^λ	1674	−1075	−84.1	2937	76.5	76.5	5.48	5.48	48.5	6.32	48.5	6.32
BP-A	F_S	1503	−323	345	443	3.64	13.8	10.4	6.01	882	95.7	87.0	951
BP-A	F_{SA}	125	−674	110	1692	2.07	20.6	29.4	1.78	1660	279	0	2770
BP-N	F_{SA}^λ	2498	−842	313	3554	190	46.6	−1.40	−24.8	89.5	7.93	44.2	230
BP-N	F_{SO}	2191	−697	−533	1079	144	−32.6	36.9	186	129	149	0.462	61.8

Tableau 6.III – Exemple X : la meilleure des 5 solutions pour les politiques BP et les fonctions objectifs sélectionnées.

Les tableaux 6.II et 6.III présentent la meilleure solution parmi les $r = 5$ solutions de LG$c\mu$ et des variantes choisies de BP. Les SL et les proportions d'abandons des meilleures solutions de certaines politiques de routage et fonctions objectifs sélectionnées sont données dans le tableau 6.IV.

Politique	Objectif	$f_{S,1}$	$f_{S,2}$	$f_{A,1}$	$f_{A,2}$
BP	F_{S}	77.3	79.1	2.4	4.9
BP	F_{S}^{1}	77.6	78.2	2.3	6.0
BP	F_{S}^{2}	79.6	75.3	2.1	10.0
BP	F_{SA}	76.2	79.7	2.5	3.9
BP	$F_{\mathrm{SA}}^{\lambda}$	78.0	75.8	2.2	6.1
BP	F_{AW}	72.5	96.6	2.8	1.6
BP-S	F_{S}	77.2	79.4	2.4	4.9
BP-S2	F_{S}	75.7	76.0	2.5	10.3
BP-A	F_{S}	77.3	79.0	2.3	4.9
BP-A	F_{SA}	76.3	80.1	2.5	3.9
LG$c\mu$	F_{S}	75.2	75.7	2.2	10.5
LG$c\mu$	F_{SA}	72.4	89.1	2.8	2.8
LG$c\mu$	$F_{\mathrm{SA}}^{\lambda}$	78.7	74.0	2.0	11.3
P	F_{S}	77.4	76.2	2.3	5.6
PD	F_{S}	77.3	79.2	2.5	5.0
PS	F_{S}	76.5	79.3	2.4	5.0
PDS	F_{S}	77.6	78.4	2.4	5.4

Tableau 6.IV – Exemple X : mesures de performance, estimées hors-échantillon, de la meilleure des 5 solutions pour les politiques et fonctions objectifs sélectionnées. Les mesures $f_{\mathrm{S},k}$ et $f_{\mathrm{A},k}$ sont exprimées en pourcentage.

La politique LG$c\mu$ est généralement plus coûteuse, car elle a souvent des SL plus bas et un nombre d'abandons plus élevé que les politiques BP et BP-S. Quand nous comparons les solutions de LG$c\mu$ pour les fonctions objectifs $F_{\mathrm{S}}, F_{\mathrm{SA}}$ et $F_{\mathrm{SA}}^{\lambda}$, nous observons que : (1) b_2 est plus grand pour F_{SA} que F_{S} afin de réduire la proportion d'abandons $f_{\mathrm{A},2}$, et (2) b_2 est plus grand pour $F_{\mathrm{SA}}^{\lambda}$ que F_{S} afin d'augmenter la qualité de service des appels du type 1. La meilleure solution de LG$c\mu$ pour F_{S} possède un coût échantillonné proche du meilleur coût échantillonné de P, soit autour de 20, mais l'évaluation hors-échantillon retourne un coût beaucoup plus élevé pour LG$c\mu$.

Pour la politique BP, les mesures de performance pour F_{S} sont légèrement plus faibles que celles de PD. Les grands $q_{k,i}$ placent davantage de poids sur les groupes plus rapides, parce que les $\mu_{k,i}$ différents ne favorisent pas le partage des agents. Le coefficient $a_{1,2}$ est relativement grand (plus grand que $a_{2,2}$) afin de compenser le paramètre $q_{1,2}$ qui est négatif, et de créer des temps de délai dynamiques. Observons dans le tableau 6.III que toutes les

solutions de BP ont les paramètres $q_{1,2}$ négatifs, et les paramètres $q_{1,1}$ et $q_{2,2}$ positifs. Les coûts estimés hors-échantillon ne varient pas beaucoup en général. Voici les coûts des 5 solutions de BP pour F_S : $8.2, 8.3, 8.6, 8.7$ et 10.5. Il y a une "mauvaise" solution avec un coût de 10.5. Ces solutions sont présentées en détails au tableau 6.V.

Coût	$q_{1,1}$	$q_{1,2}$	$q_{2,1}$	$q_{2,2}$	$a_{1,1}$	$a_{1,2}$	$a_{2,1}$	$a_{2,2}$	$b_{1,1}$	$b_{1,2}$	$b_{2,1}$	$b_{2,2}$
8.2	1129	-116	384	1213	19.3	6.48	0.952	0.831	1.25	0.787	3.68	0
8.3	1876	-123	379	1698	8.14	6.57	2.38	6.47	5.83	0.754	2.12	14.6
8.6	2356	-164	286	1055	11.3	8.63	1.97	5.69	6.86	2.40	2.92	6.17
8.7	1696	-199	383	1640	6.73	13.3	5.73	4.28	3.02	1.04	7.96	5.08
10.5	2018	-207	272	1880	10.8	9.28	5.87	9.14	3.18	5.50	3.40	7.0

Tableau 6.V – Exemple X : les 5 solutions de la politique BP pour F_S, incluant les coûts estimés hors-échantillon.

Il y a des similitudes entre les paramètres $q_{k,i}$ de BP et de BP-S. La solution de BP-S pour F_S permet d'imposer un délai sur la paire $(k = 1, i = 2)$, tout comme BP. La meilleure solution de BP-S2 est : $(q_1, q_2) = (575, 797)$, $(a_1, a_2) = (15.1, 0.028)$, $(r_1, r_2) = (374, -783)$ et $(b_1, b_2) = (12.4, 0)$. Le groupe 2 donnera la priorité au type d'appel 1 quand le temps d'attente sera supérieur à 14.7 secondes. Les mesures de performance de BP-S2 sont semblables à celles de LG$c\mu$.

Pour la variante BP-A, les paramètres $b_{k,i}$ sont beaucoup plus grands que les $a_{k,i}$ afin de compenser la différence d'échelles de taille entre les temps d'attente w_k (en secondes) et les nombres d'agents libres u_i. Par contre, les $q_{k,i}$ sont similaires à ceux de BP, à part $q_{2,2}$ qui est plus petit pour compenser le grand $b_{2,2}$. Les coefficients $b_{k,i}$ jouent un rôle plus important dans la variante BP-A. Avec les paramètres $q_{1,2} = -323, a_{1,2} = 13.8$ et $b_{1,2} = 95.7$, quand un nouvel appel du type 1 arrive, il doit y avoir 4 agents libres du groupe 2 avant que le routeur assigne un agent du groupe 2 à cet appel. En supposant qu'il y a un seul agent libre dans le groupe 2, le nouvel appel doit attendre un délai de 16 secondes avant d'être répondu par cet agent, ce qui est comparable au routage BP.

Étant donné que tous les agents peuvent servir tous les appels dans le modèle X, il n'est pas difficile d'avoir des taux d'occupation équitables ; même la politique rudimentaire E obtient des taux d'occupation $\mathbf{O} = (95.9, 96.1)$. La partie difficile est de réduire les coûts

de pénalités sur les SL dans l'objectif F_{SO}.

6.4.2 Exemple d'un modèle W et processus d'arrivée Poisson-gamma

Nous considérons un exemple un peu plus élaboré avec des processus d'arrivée Poisson-gamma et des durées de service de log-normales. Le modèle W est composé de 3 types d'appels et 2 groupes d'agents avec les ensembles d'habiletés $\mathcal{S}_1 = \{1,2\}$ et $\mathcal{S}_1 = \{2,3\}$, voir la figure 6.1. Le centre d'appels est ouvert 10 heures par jour et commence la journée avec les files d'attente vides. Le processus d'arrivée pour chaque type d'appel est Poisson-gamma, tel que le processus d'arrivée d'une journée est Poisson stationnaire, mais le taux d'arrivée (pour toute la journée) est une variable aléatoire d'une loi gamma. Pour les types d'appels 1 à 3, les variables aléatoires gammas ont des moyennes journalières de $(3000, 1000, 200)$ et des écarts types $(244.9, 223.6, 40)$. Les processus d'arrivée sont indépendants entre les types d'appels.

Les durées de service sont des variables aléatoires log-normales de moyennes $(\phi_{1,1}, \phi_{2,1}, \phi_{2,2}, \phi_{3,2}) = (8, 10, 9, 15)$ et d'écarts types $(\omega_{1,1}, \omega_{1,2}, \omega_{2,2}, \omega_{3,2}) = (8, 10, 11, 12)$, où les indices de $\phi_{k,i}$ et $\omega_{k,i}$ représentent le type d'appel k et le groupe d'agents i. Les temps de patience sont des variables exponentielles de moyennes $(\nu_1^{-1}, \nu_2^{-1}, \nu_3^{-1}) = (5, 9, 10)$. Les nombres d'agents sont $y_1 = 48$ et $y_2 = 12$, pour un total de 60 agents. Les seuils de SL sont $l_1 = l_2 = 80\%$ et $l_3 = 90\%$ avec des AWT de $\tau_1 = 60, \tau_2 = 90$ et $\tau_3 = 30$ secondes. Les paramètres de simulation sont $T = 10$ heures et $n = 300$ jours simulés. Les paramètres d'optimisation de l'AGM sont identiques à ceux utilisés pour l'exemple du modèle X.

Dans cet exemple, le type d'appel 3 a le plus petit volume, mais le groupe 2 sert plus rapidement les appels du type 2 que le groupe 1. De plus, les appels du type 3 exigent une plus grande qualité de service, alors que les appels du type 1 sont les plus impatients. La règle du *plus-rapide-serveur-en-premier* n'est pas optimale ici. Il vaut mieux "réserver" suffisamment d'agents du groupe 2 pour répondre aux appels du type 3. Les taux d'arrivée stochastiques varient considérablement : il y a des jours où le centre d'appels sera surchargé, et d'autres jours où il sera sous-chargé.

Le tableau 6.VI donne le coût moyen, estimé hors-échantillon et basé sur $r = 5$ exécutions indépendantes de l'AGM, pour chaque politique de routage et fonction objectif

Politique	F_{SA}	F_{SA}^{λ}	F_{AW}	F_{SO}
E	1188	12.39	4166	1068
P	359	8.97	3020	519
PD	230	8.95	3027	228
PS	135	**7.44**	2676	66
PDS	146	7.50	2714	**4**
LG$c\mu$	354	8.59	2896	443
BP	148	7.78	2755	60
BP-S	147	7.76	2758	119
BP-A	**126**	7.47	**2651**	20
BP-AS	131	7.49	2635	17

Tableau 6.VI – Exemple W : coût moyen des 5 solutions évaluées hors-échantillon, pour les politiques et fonctions objectifs sélectionnées.

Polit.	Obj.	$q_{1,1}$	$q_{2,1}$	$q_{2,2}$	$q_{3,2}$	$a_{1,1}$	$a_{2,1}$	$a_{2,2}$	$a_{3,2}$	$b_{1,1}$	$b_{2,1}$	$b_{2,2}$	$b_{3,2}$
BP	F_{SA}	746	244	−926	705	1.20	8.20	4.08	5.87	5.71	3.45	6.32	2.60
BP	F_{SA}^{λ}	313	316	−1337	267	192	47.5	2.31	78.9	75.3	106	25.9	47.3
BP	F_{SO}	−929	−1353	−5497	−148	16.0	18.3	0	75.9	0.31	0.82	41.4	81.1
BP	F_{AW}	1440	427	−1203	923	60.1	130	1.65	132	0	145	36.6	175
BP-A	F_{SA}	647	180	−892	1072	0.692	17.9	1.87	6.80	815	227	560	1060
BP-A	F_{SA}^{λ}	174	−85.5	−1099	395	26.8	13.0	3.57	7.15	760	376	777	464
BP-A	F_{SO}	−875	−648	−1866	365	14.4	10.6	2.85	11.7	12.8	28.4	862	623
BP-A	F_{AW}	241	414	−1444	1089	5.94	20.8	0	0	913	284	1277	956

Tableau 6.VII – Exemple W : la meilleure des 5 solutions pour les politiques BP et les fonctions objectifs sélectionnées.

sélectionnées. En résumé, les politiques utilisant des seuils donnent les meilleurs résultats : BP-A, BP-AS, PS et PDS. Les politiques BP et BP-S suivent de près, puis PD. Les politiques non conservatrices de travail (E, P et LG$c\mu$) ont les pires performances. Ces résultats concordent avec les observations de Perry et Whitt [108], où ils décrivent l'utilité des politiques avec seuils lorsqu'un centre d'appels doit faire face à des surcharges imprévus d'appels. Les tableaux 6.VII et 6.VIII présentent les meilleures solutions et les mesures de performance des politiques de routage et fonctions objectifs sélectionnées.

Les politiques P et LG$c\mu$ échouent à satisfaire le seuil l_3 de SL pour les appels du type 3, car les agents du groupe 2 passent trop de temps à servir les appels du type 2. Dans

Politique	Objectif	$f_{S,1}$	$f_{S,2}$	$f_{S,3}$	$f_{A,1}$	$f_{A,2}$	$f_{A,3}$	$f_{O,1}$	$f_{O,2}$
BP	F_{SA}	79.2	93.2	87.8	10.1	5.9	2.2	94.6	80.4
BP	F_{SO}	78.8	86.4	86.4	19.5	10.9	2.4	84.2	83.6
BP-A	F_{SA}	80.1	97.4	87.9	9.7	4.6	2.2	94.7	82.7
BP-A	F^{λ}_{SA}	86.2	80.0	78.8	7.6	8.0	3.6	94.6	84.6
BP-A	F_{SO}	79.4	90.6	89.2	18.6	7.7	2.0	85.5	85.5
LG$c\mu$	F_{SA}	76.9	100	75.7	11.1	0.8	4.1	95.6	79.2
LG$c\mu$	F^{λ}_{SA}	89.2	81.3	69.4	7.1	8.2	5.1	94.9	84.1
P	F_{SA}	76.7	100	75.6	11.1	0.8	4.1	95.7	79.3
PD	F_{SA}	76.0	93.0	83.9	11.5	6.0	2.8	96.6	68.8
PS	F_{SA}	79.6	87.3	88.3	9.7	5.6	2.1	95.7	78.2
PS	F^{λ}_{SA}	84.5	79.4	82.8	7.8	8.2	3.1	93.6	87.3
PDS	F_{SO}	84.1	81.2	88.9	19.0	8.3	2.0	85.2	85.0

Tableau 6.VIII – Exemple W : mesures de performance, estimées hors-échantillon, de la meilleure des 5 solutions pour les politiques et fonctions objectifs sélectionnées. Les mesures $f_{S,k}, f_{A,k}$ et $f_{O,i}$ sont exprimées en pourcentage.

nos meilleures solutions avec ces politiques pour F_{SA} et F_{SO}, le groupe 1 donne la priorité au type d'appel 2, et le groupe 2 donne la priorité au type d'appel 3, mais ce n'est pas suffisant. Pour F_{SA}, les mesures de performance de LG$c\mu$ sont très similaires à celles de P. Quand les pénalités sont proportionnelles aux volumes d'appels avec F^{λ}_{SA}, la pondération du type 3 est réduite par des facteurs de 15 et 5 par rapport aux types 1 et 2, respectivement. Dans ce contexte, LG$c\mu$ diminue significativement les abandons $f_{A,1}$ même si le SL $f_{S,1}$ atteint 89.2% (soit 9.2% au-dessus du seuil l_1), car les pénalités causées par les abandons $f_{A,1}$ sont plus importantes que la piètre performance du SL $f_{S,3}$ qui est 20.6% en dessous du seuil l_3. Nous avons aussi expérimenté des pondérations plus modérées, et les résultats se situent entre F_{SA} et F^{λ}_{SA}.

Pour BP-A et F_{SA}, comme $a_{2,2}$ est petit, il faudrait vraisemblablement 2 agents libres du groupe 2 avant que ces agents puissent répondre à un appel du type 2, à moins que l'appel ait attendu plus que 178 secondes. Les mesures de performance de BP-A sont meilleures que celles de BP. Avec BP, le groupe 2 répond au type d'appel 2 seulement si le temps d'attente ou le temps d'inoccupation est élevé, mais il serait une meilleure idée de ne pas attendre l'expiration du délai quand il y a beaucoup d'agents libres. Pour F^{λ}_{SA},

la meilleure solution pour BP-A est mieux que celle de LG$c\mu$, parce que BP-A améliore considérablement le SL $f_{\mathrm{S},3}$, et ce gain l'emporte sur la faible hausse d'abandons $f_{\mathrm{A},1}$. Pour F_{AW}, la meilleure solution de BP-A procure les mesures de performance suivantes : $\mathbf{S} = (77.7, 98.5, 85.9), \mathbf{A} = (10.2, 3.4, 2.5)$ et $\mathbf{W} = (30.5, 18.4, 15.0)$ secondes. Le groupe 2 peut répondre aux appels du type 2 à la condition qu'il y a 2 ou plus d'agents libres du groupe 2, car $a_{2,2} = 0$ et $\min\{v_2 \in \mathbb{N} : q_{2,2} + b_{2,2}v_2 \geq 0\} = 2$. Notons que même si le SL $f_{\mathrm{S},2}$ est près de 100%, ceci n'implique pas nécessairement que le temps d'attente moyen $f_{\mathrm{W},2}$ est proche de 0 (comme τ_2 est grand).

Les coûts des $r = 5$ solutions, estimés hors-échantillon, ne varient pas beaucoup en général pour chaque politique de routage et fonction objectif. Par contre, les solutions peuvent varier beaucoup. Voici un exemple typique, les coûts de la politique BP-A pour $F_{\mathrm{SA}}^{\lambda}$ sont : 7.31, 7.36, 7.48, 7.57 et 7.64. Ces 5 solutions sont présentées plus en détails dans le tableau 6.IX.

Coût	$q_{1,1}$	$q_{2,1}$	$q_{2,2}$	$q_{3,2}$	$a_{1,1}$	$a_{2,1}$	$a_{2,2}$	$a_{3,2}$	$b_{1,1}$	$b_{2,1}$	$b_{2,2}$	$b_{3,2}$
7.31	174	−85.5	−1099	395	26.8	13.0	3.57	7.15	760	376	777	464
7.36	332	32.4	−1292	1183	11.7	6.57	4.07	9.00	207	306	847	466
7.48	1000	704	−1508	1108	7.68	5.34	3.52	15.1	500	324	1191	528
7.57	508	15.3	−958	839	23.0	11.9	5.19	5.81	737	104	511	831
7.64	2199	187	−1079	728	6.56	12.1	5.62	1.44	356	190	610	1053

Tableau 6.IX – Exemple W : les 5 solutions de la politique BP-A pour $F_{\mathrm{SA}}^{\lambda}$, incluant les coûts estimés hors-échantillon.

6.4.3 Grand exemple : 8 types d'appels et 10 groupes d'agents

Nous considérons un grand centre d'appels composé de 8 types d'appels et 10 groupes d'agents. Les processus d'arrivée sont Poisson, et les temps de service et de patience sont des variables exponentielles avec les taux par *heure* suivants : $(\lambda_1, \ldots, \lambda_8) = (250, 200, 100, 80, 50, 20, 15, 10)$, $(\mu_1, \ldots, \mu_8) = (10, 6, 6, 10, 6, 6, 8, 10)$ et $(\nu_1, \ldots, \nu_8) = (10, 8, 10, 12, 6, 10, 12, 10)$. Notons que les temps de service sont indépendants des groupes d'agents dans cet exemple. Le vecteur d'agents est $(y_1, \ldots, y_{10})^{\mathrm{T}} = (21, 12, 14, 8, 16, 5, 3, 7, 8, 9)^{\mathrm{T}}$ pour un total de 103 agents. Les ensembles d'habiletés sont : $\mathcal{S}_1 = \{1, 4\}, \mathcal{S}_2 = \{2, 5\}, \mathcal{S}_3 =$

$\{3,4,7\}$, $\mathcal{S}_4 = \{4,6,8\}, \mathcal{S}_5 = \{2,5\}, \mathcal{S}_6 = \{6,7,8\}, \mathcal{S}_7 = \{1,3,7\}$, $\mathcal{S}_8 = \{2,4,8\}, \mathcal{S}_9 = \{1,3,4,8\}$ et $\mathcal{S}_{10} = \{2,7,8\}$. Remarquons que les groupes 2 et 5 possèdent les mêmes habiletés, mais leurs paramètres de routage peuvent être différents. Les seuils de SL sont $l_k = 80\%$ avec $\tau_k = 20$ secondes pour tous les types d'appels. Les paramètres de simulation sont $T = 100$ heures et $n = 10$ réplications. Ce modèle stochastique est plus simple que les exemples X et W précédents, mais il y a beaucoup plus de paramètres de routage à optimiser.

Nous exécutons l'AGM avec les paramètres $P = 400, \hat{P} = 20$ et $\mathsf{maxIt} = 40$. Pour PDS, nous choisissons $P = 200, \hat{P} = 10, \mathsf{maxIt} = 20$ et le paramètre multi-étapes $\mathsf{itAGM} = 10$. Les autres paramètres de l'AGM sont identiques à ceux utilisés pour l'exemple du modèle X. Il est difficile d'optimiser simultanément tous les paramètres des politiques PD et PS. Nous utilisons une approche d'optimisation à deux étapes. La première étape consiste à optimiser les listes de priorités avec l'AGM et maxIt=30, puis la deuxième étape optimise les temps de délai ou les seuils d'agents libres avec maxIt=10. Cette approche à deux étapes est sous-optimale, mais elle trouve de meilleures solutions pour les budgets de temps alloués.

Politique	F_{S}	F_{SA}	$F_{\mathrm{SA}}^{\lambda}$	F_{AW}
E	616	722	26.3	2562
P	231	334	14.4	1896
PD*	192	318	13.3	1896
PS*	147	275	11.3	1894
PDS	165	312	11.9	1898
LG$c\mu$	209	304	13.6	1811
BP	107	241	9.4	1794
BP-S	108	242	9.1	1794
BP-A	**96**	**233**	**8.6**	**1761**
BP-AS	120	263	9.7	1860
BP-N	**61**	**230**	**7.7**	1802

Tableau 6.X – Grand exemple : coût moyen des 3 solutions évaluées hors-échantillon, pour les politiques de routage et fonctions objectifs sélectionnées.

Le tableau 6.X présente le coût moyen des $r = 3$ solutions évaluées hors-échantillon, pour les politiques de routage et fonctions objectifs sélectionnées. Les politiques PD* et PS* dans le tableau sont les résultats de la variante AGM à deux étapes. Les résultats

de l'AGM original à une étape pour PD et PS sont pires que les résultats pour P. Parmi les politiques considérées dans cette étude, la politique BP-A est clairement la meilleure, avec les coûts colorés en rouge, et elle est suivie de BP et BP-S. Si nous permettons des coefficients négatifs, alors la politique BP-N obtient des coûts encore plus bas quand les SL sont pénalisés (coûts colorés en bleu). La politique LG$c\mu$ bat seulement les politiques E et P quand les SL sont pénalisés, mais elle marche relativement bien sous l'objectif F_{AW}.

Les coûts estimés hors-échantillon des 3 solutions ne varient pas considérablement. Pour BP-A, les coûts sont 95.19, 95.37 et 96.56 pour F_{S}, et les coûts sont 232.27, 233.04 et 234.25 pour F_{SA}. La meilleure solution de BP-A pour F_{SA} donne les mesures de performance suivantes : $\mathbf{S} = (75.3, 70.8, 76.3, 81.2, 78.9, 84.6, 89.7, 95.5)$, $\mathbf{A} = (4.1, 5.2, 5.0, 4.6, 2.0, 3.6, 2.1, 0.7)$, les taux d'occupation maximal et minimal $f_{\mathrm{O},10} = 93.7\%$ et $f_{\mathrm{O},6} = 78.0\%$, les temps d'attente moyens maximal et minimal $f_{\mathrm{W},2} = 23.2$ et $f_{\mathrm{W},8} = 2.6$ secondes. Pour l'objectif $F_{\mathrm{SA}}^{\lambda}$, la meilleure solution de BP-A donne : $\mathbf{S} = (77.6, 73.0, 75.3, 78.6, 72.7, 77.9, 77.2, 82.8)$, $\mathbf{A} = (3.5, 4.6, 5.6, 5.1, 3.4, 6.1, 6.3, 3.7)$, les taux d'occupation maximal et minimal $f_{\mathrm{O},5} = 94.4\%$ et $f_{\mathrm{O},4} = 85.8\%$, et les temps d'attente moyens maximal et minimal $f_{\mathrm{W},6} = 21.9$ et $f_{\mathrm{W},1} = 12.8$ secondes. Nous voyons des différences sur les mesures de performance lorsque nous changeons la pondération des pénalités entre F_{SA} et $F_{\mathrm{SA}}^{\lambda}$. Avec F_{SA}, il est plus avantageux d'améliorer les qualités de service des types d'appels ayant de faibles volumes. Tandis qu'il est plus avantageux de favoriser les types d'appels avec de forts volumes sous l'objectif $F_{\mathrm{SA}}^{\lambda}$. Quatre types d'appels ont des SL supérieurs aux seuils de 80% sous F_{SA}, dont les trois types ayant les plus faibles volumes d'appels. Avec $F_{\mathrm{SA}}^{\lambda}$, seulement le type ayant le plus faible volume obtient un SL au-dessus de 80%, mais les proportions d'abandons sont plus bas pour les types d'appels avec de fort volumes.

6.4.4 Moyen exemple : 6 types d'appels et 3 groupes d'agents

Cet exemple est un modèle simple composé de 6 types d'appels et 3 groupes d'agents, avec des processus d'arrivée Poisson, et des durées de service et de patience exponentielles. Les particularités de cet exemple sont que : (1) tous les agents possèdent toutes les habiletés, et (2) les durées de service dépendent à la fois aux types d'appels et aux groupes d'agents. L'ensemble d'habiletés de chaque groupe est donc relativement grand, et ceci

donne beaucoup de paramètres de routage à optimiser.

Les taux d'arrivée par minute sont $(\lambda_1,\ldots,\lambda_6) = (0.6, 0.48, 0.3, 0.9, 0.06, 0.12)$. La rapidité (des temps de service) des groupes est modélisée afin d'avoir toutes les permutations d'un ensemble de 3 possibles (donc 6 configurations au total). Chaque groupe est le plus rapide, moyen et le plus lent pour exactement deux types d'appels. Pour chaque type d'appel, il y a un ordre distinct des taux de service, c'est-à-dire $\mu_{k,i} \neq \mu_{k,j}$ pour tout k, i, j. Les taux de service par minute sont $(\mu_{1,1},\ldots,\mu_{1,6}) = (0.09, 0.132, 0.156, 0.096, 0.18, 0.042)$, $(\mu_{2,1},\ldots,\mu_{2,6}) = (0.078,\ 0.096,\ 0.18,\ 0.12,\ 0.12,\ 0.048)$, et $(\mu_{3,1},\ldots,\mu_{3,6}) = (0.072, 0.12,\ 0.132, 0.102, 0.24, 0.06)$. La durée moyenne de service la plus rapide est de 4.2 minutes, et la plus lente est de 23.8 minutes. Les temps de patience ont des taux de $(\nu_1,\ldots,\nu_6) = (0.12, 0.18, 0.06, 0.048, 0.072, 0.042)$ par minute. La durée moyenne de patience la plus longue est de 23.8 minutes, et la plus courte est de 5.6 minutes. Les seuils des SL sont $(l_1,\ldots,l_6) = (80\%, 90\%, 90\%, 80\%, 80\%, 80\%)$ et les AWT sont $(\tau_1,\ldots,\tau_6) = (10, 40, 20,\ 60, 30, 30)$ secondes. Le type d'appel 3 nécessite la plus haute qualité de service, tandis que le type 4 en requiert le moins. Les nombres d'agents sont $y_1 = 9, y_2 = 10$ et $y_3 = 7$, pour un total de 26 agents.

Les paramètres de simulation sont $T = 10$ heures et $n = 200$ réplications. Nous exécutons l'AGM avec les paramètres $P = 300, \hat{P} = 20$ et $\mathsf{maxIt} = 40$. Pour la politique PDS, nous choisissons $P = 150, \hat{P} = 10, \mathsf{maxIt} = 10$ et le paramètre multi-étapes $\mathsf{itAGM} = 8$. Les autres paramètres de l'AGM sont identiques à ceux utilisés pour l'exemple du modèle X. Comme pour le grand exemple, les résultats de PD* et PS* sont obtenues par une version à deux étapes de l'AGM, où nous fixons $\mathsf{maxIt} = 20$ pour chaque étape (pour un total de 40 itérations de l'AGM). Rappelons que l'approche à deux étapes est sous-optimale, mais elle trouve des meilleures solutions que l'AGM original avec nos budgets de temps limités.

Le tableau 6.XI présente le coût moyen de $r = 3$ solutions évaluées hors-échantillon, pour les politiques de routage et les fonctions objectifs choisies. Parmi toutes les politiques considérées, BP-A est clairement la meilleure pour cet exemple en obtenant les plus bas coûts pour toutes les quatre fonctions objectifs. D'autre part, les politiques E, P, LG$c\mu$ et BP-S2 sont les plus dispendieuses. La politique BP est meilleure que la version simplifiée BP-S, sauf pour $F^{\lambda}_{\mathrm{SA}}$. BP-N est mieux que BP-A seulement si nous ne pénalisons pas les

Politique	F_{S}	F_{SA}	$F_{\mathrm{SA}}^{\lambda}$	F_{SO}
E	1978.4	2205	13.79	1978.8
P	78.4	155	1.37	144.2
PD*	18.0	71	0.69	70.0
PS*	12.2	65	0.56	34.7
PDS	18.2	69	0.69	40.3
LG$c\mu$	231.1	340	2.57	241.2
BP	5.7	54	0.49	8.0
BP-S	6.2	58	0.47	30.4
BP-S2	468.4	545	5.95	470.9
BP-A	**0.4**	**45**	**0.41**	**2.0**
BP-AS	7.3	60	0.51	13.9
BP-N	**0.2**	52	0.47	**0.3**

Tableau 6.XI – Moyen exemple : coût moyen des 3 solutions évaluées hors-échantillon, pour les politiques de routage et fonctions objectifs sélectionnées.

abandons. Les solutions de BP et BP-AS ne sont pas mauvaises non plus. Les solutions de BP-AS et PDS sont visiblement non optimales, car autrement, elles auraient des coûts égaux ou inférieurs à ceux de BP et PS*, respectivement. Le tableau 6.XII présente les meilleures solutions des politiques sélectionnées pour la fonction objectif F_{SA} avec les coûts estimés hors-échantillon. Les variables $d_{k,i} = 0$ et $m_{k,i} = 0$ sont omises du tableau.

Pour BP-A et F_{SA}, nous obtenons $\mathbf{S} = (80.0, 89.6, 89.3, 79.6, 86.6, 80.6)$ et $\mathbf{A} = (4.1, 3.5, 0.6, 3.0, 1.6, 1.6)$. Les seuils des SL sont quasiment tous atteints, et les pénalités sont dues principalement aux abandons. La meilleure solution de PS* pour F_{SA} donne $\mathbf{S} = (79.2, 90.9, 88.7, 76.5, 88.1, 84.2)$ et $\mathbf{A} = (4.2, 2.9, 0.7, 3.6, 1.4, 1.4)$. La solution de BP-A est légèrement plus raffinée que celle de PS*. BP-A satisfait mieux les seuils de SL, sans toutefois avoir de grand excès au-dessus des seuils. Malgré des proportions d'abandons comparables, la proportion d'abandons agrégée sur tous les types est légèrement plus faible avec BP-A, soit 2.98% contre 3.10% avec PS*.

6.4.5 Robustesse des politiques de routage

Nous comparons la robustesse des politiques de routage face aux imprévus à travers l'exemple du modèle W, présenté à la section 6.4.2. Pour effectuer ces tests, nous sélec-

Coût	Solution de routage
130	**P :** $\mathcal{G}_1 = (\{1\},\{2\},\{3\}), \mathcal{G}_2 = (\{1,3\},\{2\}), \mathcal{G}_3 = (\{2\},\{1\},\{3\}), \mathcal{G}_4 = (\{2\},\{3\},\{1\}), \mathcal{G}_5 = (\{1,3\},\{2\}), \mathcal{G}_6 = (\{3\},\{2\},\{1\}), \mathcal{L}_1 = (\{2\},\{3,5\},\{1\},\{4\},\{6\}), \mathcal{L}_2 = (\{3,5\},\{4\},\{2\},\{1,6\}), \mathcal{L}_3 = (\{3,6\},\{2\},\{5\},\{1,4\})$.
65	**PD* :** $\mathcal{G}_1 = (\{1,2\},\{3\}), \mathcal{G}_2 = (\{1\},\{3\},\{2\}), \mathcal{G}_3 = (\{2\},\{1,3\}), \mathcal{G}_4 = (\{2\},\{1,3\}), \mathcal{G}_5 = (\{1,2,3\}), \mathcal{G}_6 = (\{3\},\{1\},\{2\}), \mathcal{L}_1 = (\{2\},\{3\},\{1\},\{6\},\{5\},\{4\}), \mathcal{L}_2 = (\{3\},\{5\},\{4,6\},\{1,2\}), \mathcal{L}_3 = (\{2\},\{6\},\{3\},\{5\},\{1,4\}), d_{1,2} = 6.8, d_{1,3} = 2.1, d_{2,2} = 134, d_{2,3} = 1.1, d_{4,1} = 155, d_{4,2} = 0.2, d_{4,3} = 30.9, d_{5,1} = 7.8, d_{5,2} = 23.4, d_{5,3} = 5.3, d_{6,1} = 352.$
58	**PS* :** $\mathcal{G}_1 = (\{1\},\{3\},\{2\}), \mathcal{G}_2 = (\{1\},\{3\},\{2\}), \mathcal{G}_3 = (\{1,2\},\{3\}), \mathcal{G}_4 = (\{2\},\{1,3\}), \mathcal{G}_5 = (\{3\},\{1\},\{2\}), \mathcal{G}_6 = (\{3\},\{1,2\}), \mathcal{L}_1 = (\{2,3,5\},\{1\},\{4\},\{6\}), \mathcal{L}_2 = (\{3\},\{1,4\},\{5\},\{2\},\{6\}), \mathcal{L}_3 = (\{2\},\{6\},\{3,5\},\{1,4\}), m_{1,1} = 0.07, m_{1,2} = 0.29, m_{1,3} = 0.52, m_{2,2} = 4.93, m_{2,3} = 0.37, m_{3,1} = 0.3, m_{4,1} = 1.52, m_{4,3} = 0.99, m_{5,2} = 0.92, m_{6,1} = 1.22.$
61	**PDS :** $\mathcal{G}_1 = (\{3\},\{1,2\}), \mathcal{G}_2 = (\{3\},\{2\},\{1\}), \mathcal{G}_3 = (\{2\},\{1\},\{3\}), \mathcal{G}_4 = (\{1\},\{2,3\}), \mathcal{G}_5 = (\{1,2,3\}), \mathcal{G}_6 = (\{1,2,3\}), \mathcal{L}_1 = (\{2,4\},\{1,3,5\},\{6\}), \mathcal{L}_2 = (\{5\},\{6\},\{2\},\{3\},\{4\},\{1\}), \mathcal{L}_3 = (\{1,6\},\{5\},\{2,3\},\{4\}), d_{1,2} = 1.4, d_{1,3} = 2.9, d_{2,1} = 1.2, d_{2,2} = 36, d_{2,3} = 4.5, d_{3,1} = 8.3, d_{4,1} = 52, d_{4,3} = 23.6, d_{5,1} = 14.5, d_{5,2} = 8.5, d_{6,1} = 25.7, d_{6,2} = 16.1, d_{6,3} = 4.3, m_{1,3} = 1.51, m_{2,2} = 4.88, m_{3,1} = 0.01, m_{4,1} = 1.9, m_{4,2} = 0.12, m_{4,3} = 0.95, m_{5,2} = 1.14, m_{5,3} = 0.11, m_{6,1} = 1.34, m_{6,2} = 2.33, m_{6,3} = 0.2.$
331	**LG$c\mu$:** $a_1 = 1154, b_1 = 0.196, a_2 = 1123, b_2 = 0, a_3 = 2262, b_3 = 330, a_4 = 910, b_4 = 0.444, a_5 = 1124, b_5 = 51.1, a_6 = 1994, b_6 = 54.2, e_1 = 3091, f_1 = 119, e_2 = 2730, f_2 = 127, e_3 = 2835, f_3 = 118.$
54	**BP :** $q_{1,1} = 219, q_{1,2} = 276, q_{1,3} = -748, q_{2,1} = 845, q_{2,2} = -1293, q_{2,3} = 317, q_{3,1} = 2446, q_{3,2} = 3686, q_{3,3} = 202, q_{4,1} = -1131, q_{4,2} = 689, q_{4,3} = -1623, q_{5,1} = -331, q_{5,2} = -3059, q_{5,3} = 509, q_{6,1} = -1433, q_{6,2} = -497, q_{6,3} = 350, a_{1,1} = 1.77, a_{1,2} = 1.42, a_{1,3} = 0.223, a_{2,1} = 31.2, a_{2,2} = 7.17, a_{2,3} = 28.2, a_{3,1} = 122, a_{3,2} = 97.4, a_{3,3} = 59.9, a_{4,1} = 2.24, a_{4,2} = 12.9, a_{4,3} = 2.13, a_{5,1} = 36.7, a_{5,2} = 31.0, a_{5,3} = 82.4, a_{6,1} = 3.05, a_{6,2} = 12.4, a_{6,3} = 142, b_{1,1} = 180, b_{1,2} = 22.2, b_{1,3} = 15.7, b_{2,1} = 139, b_{2,2} = 12.6, b_{2,3} = 60.1, b_{3,1} = 24.3, b_{3,2} = 120, b_{3,3} = 10.6, b_{4,1} = 4.02, b_{4,2} = 144, b_{4,3} = 29.2, b_{5,1} = 36.5, b_{5,2} = 6.01, b_{5,3} = 94.8, b_{6,1} = 4.36, b_{6,2} = 20.5, b_{6,3} = 97.5.$
45	**BP-A :** $q_{1,1} = 124, q_{1,2} = 316, q_{1,3} = 24.0, q_{2,1} = 812, q_{2,2} = -594, q_{2,3} = 1137, q_{3,1} = 1648, q_{3,2} = 2189, q_{3,3} = 516, q_{4,1} = -1147, q_{4,2} = 1083, q_{4,3} = -477, q_{5,1} = 769, q_{5,2} = 342, q_{5,3} = 2252, q_{6,1} = -1550, q_{6,2} = 412, q_{6,3} = 2076, a_{1,1} = 2.11, a_{1,2} = 7.09, a_{1,3} = 3.33, a_{2,1} = 1.29, a_{2,2} = 5.54, a_{2,3} = 1.56, a_{3,1} = 4.37, a_{3,2} = 5.80, a_{3,3} = 5.63, a_{4,1} = 1.33, a_{4,2} = 3.63, a_{4,3} = 3.41, a_{5,1} = 3.87, a_{5,2} = 8.40, a_{5,3} = 6.70, a_{6,1} = 0.934, a_{6,2} = 4.41, a_{6,3} = 3.69, b_{1,1} = 1176, b_{1,2} = 167, b_{1,3} = 152, b_{2,1} = 1313, b_{2,2} = 426, b_{2,3} = 863, b_{3,1} = 849, b_{3,2} = 2362, b_{3,3} = 398, b_{4,1} = 169, b_{4,2} = 1138, b_{4,3} = 315, b_{5,1} = 350, b_{5,2} = 322, b_{5,3} = 850, b_{6,1} = 525, b_{6,2} = 218, b_{6,3} = 1067.$
50	**BP-N :** $q_{1,1} = 2102, q_{1,2} = 477, q_{1,3} = 302, q_{2,1} = 2173, q_{2,2} = 190, q_{2,3} = 578, q_{3,1} = 1655, q_{3,2} = 2012, q_{3,3} = -70.5, q_{4,1} = -1855, q_{4,2} = 2188, q_{4,3} = 380, q_{5,1} = 1502, q_{5,2} = 1039, q_{5,3} = 1019, q_{6,1} = -2612, q_{6,2} = 857, q_{6,3} = 1707, a_{1,1} = -3.29, a_{1,2} = 4.49, a_{1,3} = -55.3, a_{2,1} = 76.2, a_{2,2} = 17.8, a_{2,3} = 32.4, a_{3,1} = 132, a_{3,2} = 224, a_{3,3} = 37.0, a_{4,1} = -86.3, a_{4,2} = -0.954, a_{4,3} = -46.6, a_{5,1} = 107, a_{5,2} = -5.66, a_{5,3} = 82.1, a_{6,1} = -22.6, a_{6,2} = -10.1, a_{6,3} = 139, b_{1,1} = 188, b_{1,2} = 51.2, b_{1,3} = 8.47, b_{2,1} = 66.9, b_{2,2} = -34.7, b_{2,3} = 60.3, b_{3,1} = 70.48, b_{3,2} = 243, b_{3,3} = 22.1, b_{4,1} = 12.5, b_{4,2} = 261, b_{4,3} = 92.3, b_{5,1} = 46.7, b_{5,2} = 32.7, b_{5,3} = 253, b_{6,1} = 17.3, b_{6,2} = 15.8, b_{6,3} = 169.$

Tableau 6.XII – Moyen exemple : la meilleure solution sur 3 exécutions pour les politiques de routage choisies et la fonction objectif F_{SA}, en incluant le coût estimé hors-échantillon.

tionnons la meilleure des $r = 5$ solutions de chaque politique pour les fonctions objectifs F_{SA} et $F_{\mathrm{SA}}^{\lambda}$, et nous changeons les paramètres du centre d'appels de manière déterministe. Les solutions de BP et BP-A se trouvent dans le tableau 6.VII à la page 191, et les solutions des autres politiques sont présentées dans le tableau 6.XIII. Les variables $d_{k,i} = 0$ et $m_{k,i} = 0$, et les listes de priorités de longueur 1 sont cachées de ce dernier tableau. Notons que la solution de PDS pour $F_{\mathrm{SA}}^{\lambda}$ ne contient aucun temps de délai. Nous simulons 2000 jours pour chaque scénario. Dans les figures de cette section, nous comparons les politiques selon leur coût relatif au meilleur coût (parmi les politiques testées). La meilleure politique a un coût relatif de 1. L'axe vertical des figures grandit en suivant une échelle logarithmique.

Obj.	Solution de routage
F_{SA}	**P :** $\mathcal{G}_2 = (\{1\},\{2\}), \mathcal{L}_1 = (\{2\},\{1\}), \mathcal{L}_2 = (\{3\},\{2\})$. **PD :** $\mathcal{G}_2 = \mathcal{L}_1 = (\{1,2\}), \mathcal{L}_2 = (\{3\},\{2\}), d_{2,1} = 0.31, d_{2,2} = 43.1, d_{3,2} = 0.0030$. **PS :** $\mathcal{G}_2 = (\{1\},\{2\}), \mathcal{L}_1 = (\{1,2\}), \mathcal{L}_2 = (\{2,3\}), m_{2,1} = 0.49, m_{2,2} = 1.04$. **PDS :** $\mathcal{G}_2 = \mathcal{L}_1 = (\{1,2\}), \mathcal{L}_2 = (\{2,3\}), d_{2,2} = 0.13, m_{2,1} = 0.29, m_{2,2} = 1.06$. **LG$c\mu$:** $a_1 = 18.8, b_1 = 0.834, a_2 = 536, b_2 = 8.23, a_3 = 1892, b_3 = 270, e_1 = 1639, f_1 = 237, e_2 = 75.9, f_2 = 2.033$. **BP-AS :** $q_{1,1} = 778, q_{2,1} = -227, q_{2,2} = 420, q_{3,2} = 751, a_{1,1} = 4.06, a_{2,1} = 20.5, a_{2,2} = 9.58, a_{3,2} = 2.66, b_{1,1} = 234, b_{2,1} = 1595, b_{2,2} = 318, b_{3,2} = 859, m_{2,2} = 1.19$.
$F_{\mathrm{SA}}^{\lambda}$	**P :** $\mathcal{G}_2 = \mathcal{L}_1 = (\{1\},\{2\}), \mathcal{L}_2 = (\{3\},\{2\})$. **PD :** $\mathcal{G}_2 = (\{2\},\{1\}) = \mathcal{L}_1 = (\{1\},\{2\}), \mathcal{L}_2 = (\{3\},\{2\}), d_{2,2} = 9.11$. **PS :** $\mathcal{G}_2 = \mathcal{L}_1 = (\{1,2\}), \mathcal{L}_2 = (\{2\},\{3\}), m_{2,1} = 0.78, m_{2,2} = 0.97$. **PDS :** $\mathcal{G}_2 = \mathcal{L}_1 = (\{1,2\}), \mathcal{L}_2 = (\{2,3\}), m_{2,1} = 0.80, m_{2,2} = 0.97$. **LG$c\mu$:** $a_1 = 6.29, b_1 = 41.1, a_2 = 721, b_2 = 0.065, a_3 = 1928, b_3 = 378, e_1 = 1607, f_1 = 64.7, e_2 = 94.5, f_2 = 12.0$. **BP-AS :** $q_{1,1} = 213, q_{2,1} = -102, q_{2,2} = -245, q_{3,2} = 111, a_{1,1} = 162, a_{2,1} = 52.3, a_{2,2} = 49.1, a_{3,2} = 27.2, b_{1,1} = 1535, b_{2,1} = 610, b_{2,2} = 505, b_{3,2} = 672, m_{1,1} = 0.01, m_{2,1} = 0.02, m_{2,2} = 0.93, m_{3,2} = 0.01$.

Tableau 6.XIII – Solutions utilisées pour les tests de robustesse de l'exemple du modèle W qui n'ont pas déjà été présentées.

Nous commençons par expérimenter la robustesse des politiques avec différents processus d'arrivée. Les politiques de routage ont été optimisées en supposant un processus d'arrivée Poisson-gamma indépendant pour chaque type d'appel, où chaque jour est un processus de Poisson stationnaire avec un taux journalier aléatoire d'une loi gamma. Nous

désirons à présent vérifier les performances des solutions de routage lorsque nous simulons un jour en particulier avec des taux d'arrivée fixes. Les taux sont choisis à une distance maximale d'un écart-type des moyennes des variables gamma, de sorte que ces scénarios sont susceptibles de se produire. Soulignons que les politiques de routage ne sont pas ré-optimisées.

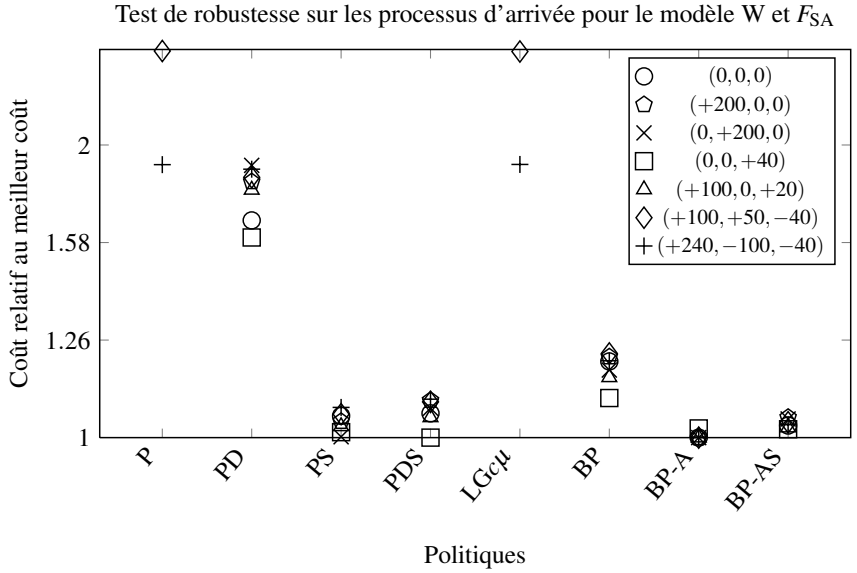


Figure 6.2 – Test de robustesse du modèle W et l'objectif F_{SA} : simuler la meilleure solution de chaque politique en changeant les taux d'arrivée par les quantités (z_1, z_2, z_3). L'axe vertical est échelonné de 1 à 2.5. Plusieurs solutions de P et LG$c\mu$ n'apparaissent pas, car elles sont au-dessus de 2.5.

Les figures 6.2 et 6.3 comparent les résultats où le taux d'arrivée journalier du type d'appel k est fixé à une quantité z_k du taux moyen. Rappelons que les taux d'arrivée moyens sont (3000, 1000, 200). Par exemple, le scénario $(0,0,0)$ représente le cas des processus de Poisson stationnaires avec les taux $(\lambda_1, \lambda_2, \lambda_3) = (3000, 1000, 200)$. La politique BP-A est la plus robuste pour les objectifs F_{SA} et $F^{\lambda}_{\mathrm{SA}}$. En résumé, les politiques BP-AS, PDS et PS présentent aussi des résultats compétitifs, suivies par BP. Par contre, les politiques PD et LG$c\mu$ ont mal performé dans ces tests. L'utilisation de seuils d'agents fixes dans PS et PDS n'est pas nécessairement efficace et peut parfois nuire au routage. Dans cet exemple, le groupe 2 donne la priorité au type d'appel 3, et les politiques PS et PDS affectent une

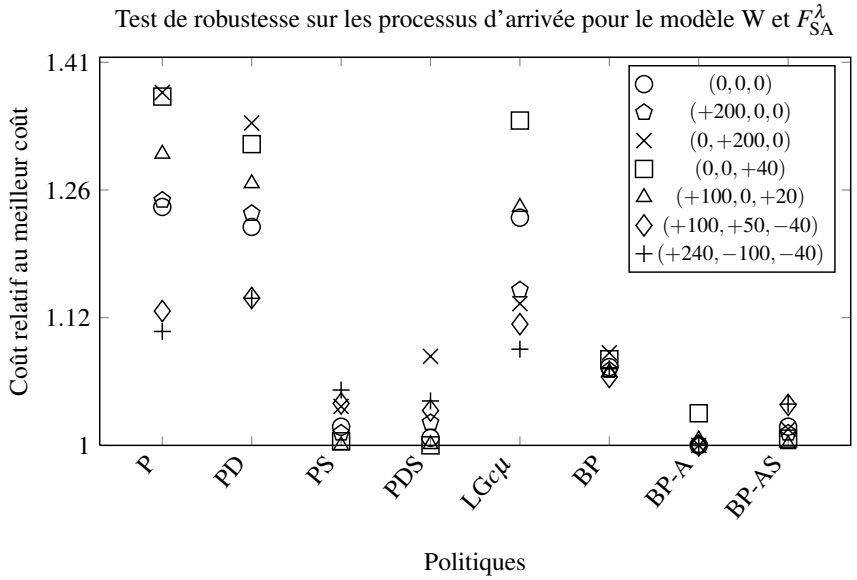


Figure 6.3 – Test de robustesse du modèle W et l'objectif $F_{\mathrm{SA}}^{\lambda}$: simuler la meilleure solution de chaque politique en changeant les taux d'arrivée par les quantités (z_1, z_2, z_3).

valeur positive au seuil $m_{2,2}$. Les politiques PS et PDS performent bien quand le type d'appel 3 devient surchargé, mais pas lorsque le type 3 est sous-chargé et le type 1 est surchargé. Dans ce dernier cas, les agents du groupe 1 doivent servir plus d'appels du type 1, car ils sont les seuls à posséder la compétence requise, et par conséquent, les agents du groupe 2 doivent servir plus d'appels du type 2. Il n'est donc pas une bonne idée d'avoir des seuils fixes dans cette situation. La politique BP-A serait un meilleur choix de routage.

Dans le deuxième test, nous modifions le vecteur d'agents, mais en gardant les processus d'arrivée Poisson-gamma originaux. Nous n'incrémentons pas le nombre total d'agents, parce que le test devient moins intéressant et l'absentéisme est un problème beaucoup plus commun dans la réalité. Le vecteur d'agents original contient $y_1 = 48$ et $y_2 = 12$, et nous changeons ces nombres par les quantités Δy_1 et Δy_2, respectivement. Les figures 6.4 et 6.5 présentent les résultats pour différents scénarios. Notons que le scénario $\Delta y_1 = \Delta y_2 = 0$ correspond au problème original. Réduire le nombre d'agents augmente la charge de travail des agents restants, et les politiques avec seuils (BP-A, BP-AS, PS et PDS) donnent de meilleurs résultats. La politique BP-A est la plus robuste pour F_{SA} et $F_{\mathrm{SA}}^{\lambda}$, et elle est suivie

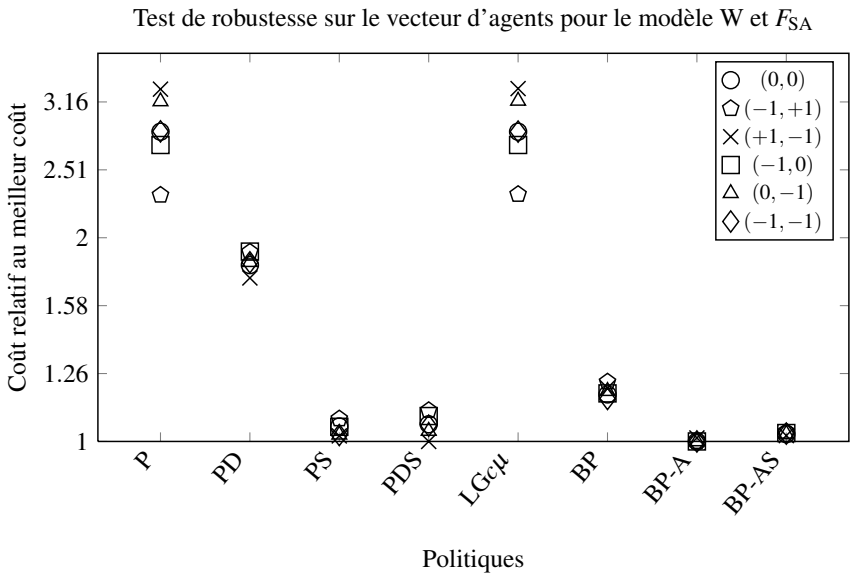


Figure 6.4 – Test de robustesse du modèle W et l'objectif F_{SA} : simuler la meilleure solution de chaque politique en changeant le vecteur d'agents par les quantités $(\Delta y_1, \Delta y_2)$.

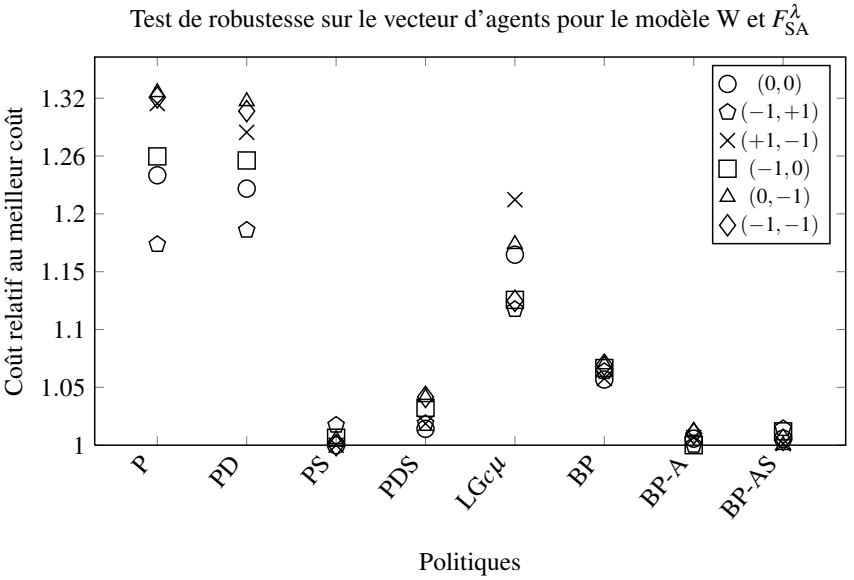


Figure 6.5 – Test de robustesse du modèle W et l'objectif $F_{\mathrm{SA}}^{\lambda}$: simuler la meilleure solution de chaque politique en changeant le vecteur d'agents par les quantités $(\Delta y_1, \Delta y_2)$.

de près par BP-AS et PS.

D'autre part, nous avons expérimenté des exemples de modèles M (2 types et 3 groupes) et N (2 types et 2 groupes) avec différentes configurations des vecteurs d'agents. Chaque modèle possède un groupe de généralistes et un ou deux groupes de spécialistes. Nous avons optimisé les problèmes en variant les quantités de généralistes et de spécialistes, et les conclusions obtenues sont similaires.

6.4.6 Convergence de la méthode AGM

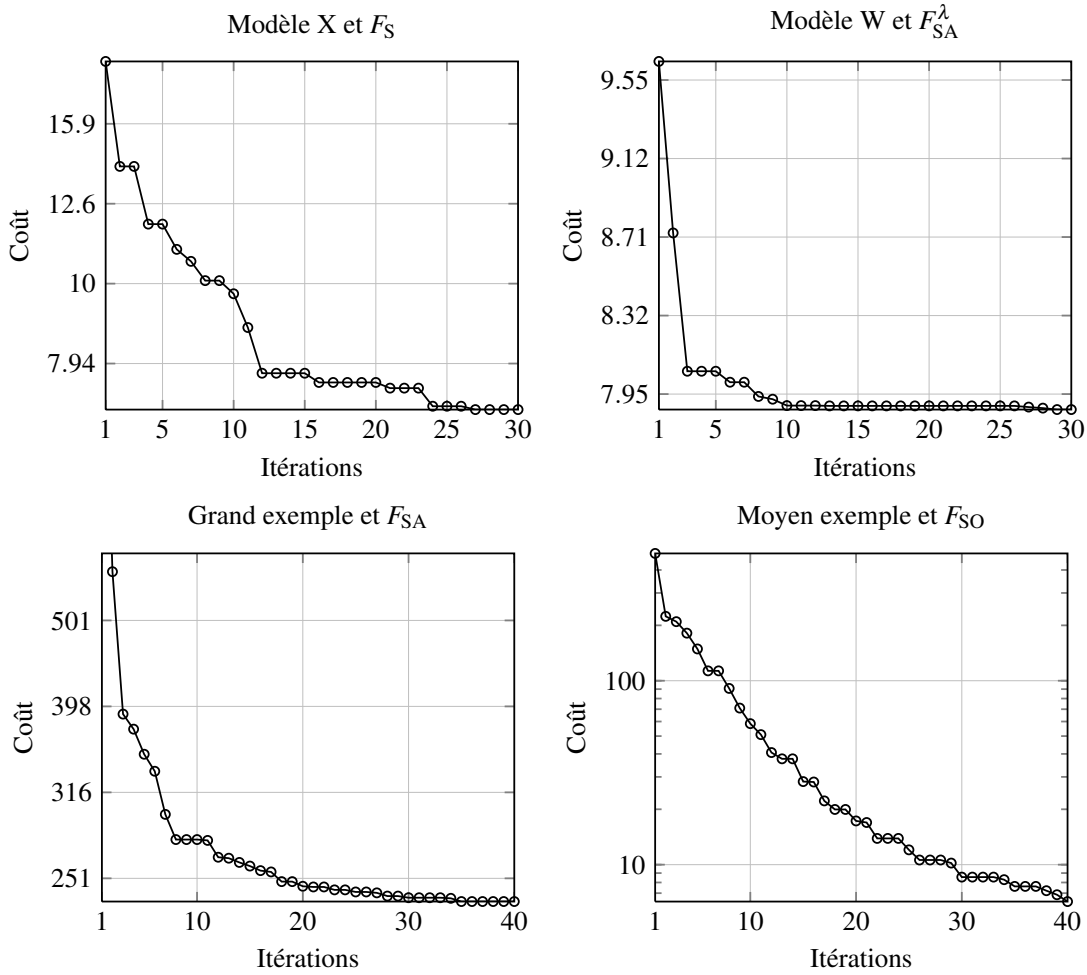


Figure 6.6 – Évolution des meilleures exécutions de l'AGM avec de la politique BP pour différents exemples et fonctions objectifs. L'axe vertical suit une échelle logarithmique.

Dans cette section, nous donnons une idée (empirique) de la convergence de la méthode d'optimisation AGM. La figure 6.6 présente l'évolution des meilleures exécutions de l'AGM avec la politique BP pour différents exemples et fonctions objectifs. En général, l'AGM commence avec une solution très dispendieuse, puis réduit le coût rapidement lors des premières itérations. Cependant, l'AGM réduit faiblement le coût dans la deuxième moitié de l'algorithme. Étant donné que l'axe vertical suit une échelle logarithmique, la courbe semble diminuer substantiellement dans le cas du moyen exemple, mais la réduction de coût devient très faible en réalité. La solution initiale est particulièrement très coûteuse quand il y a beaucoup de variables à optimiser. La solution à l'itération 1 du grand exemple coûte 1423 et n'apparaît pas dans la figure.

6.5 Conclusion

Dans ce chapitre, nous avons étudié exclusivement l'optimisation du routage de centres d'appels. Nous supposons que les paramètres des processus stochastiques du centre d'appels et l'affectation des agents sont fixes. Le problème de routage consiste à décider comment assigner les agents aux appels. Le problème de routage est difficile lorsqu'il y a plusieurs types d'appels et plusieurs groupes d'agents, car il y a beaucoup de choix de routage possibles.

À l'opposé de la littérature, dont les études se basent généralement sur des régimes asymptotiques à trafic intense (volumes très élevés pour tous les types d'appels), nous avons considéré des centres d'appels plus réalistes, avec des modèles stochastiques plus complexes et différents volumes d'appels, élevés ou faibles. Notre approche d'optimisation boîte noire combinée avec la simulation permet d'optimiser le routage pour une grande diversité de centres d'appels.

Nous avons proposé une politique de routage basé sur des poids (BP). Les poids sont calculés en fonction du temps d'attente des appels, du temps d'inoccupation des agents ou du nombre d'agents libres. La politique BP peut approximer les politiques à priorités égales (E), avec des listes de priorités (P), et listes priorités avec délais (PD). Nous avons aussi proposé une adaptation linéaire de la règle $c\mu$ généralisée (LG$c\mu$), cependant ses

paramètres doivent être optimisés. Nous avons comparé la politique BP avec diverses politiques de routage, inspirées de politiques réelles utilisées dans la pratique et de politiques proposées dans la littérature.

Au lieu d'optimiser avec des contraintes dures, notre approche est basée sur la minimisation des coûts de pénalités. Comme il n'existe aucune méthode établie dans l'industrie pour évaluer les solutions, nous choisissons une approche très générale, qui est de supposer une fonction objectif de type boîte noire pouvant être évaluée par la simulation (ou autres évaluateurs). Nous avons présenté différentes fonctions de pénalités à la section 6.1. Nous avons considéré des pénalités sur les niveaux de service, les proportions d'abandons, les temps d'attente moyens, les taux d'occupation, ainsi que des critères d'équités entre les types d'appels ou les groupes d'agents. À notre connaissance, notre étude est la première étude sur le routage des centres d'appels à utiliser une approche de modélisation aussi générale.

Afin d'avoir une comparaison "juste", nous désirions comparer les politiques avec leurs paramètres optimaux. Cependant, optimiser le routage est un problème difficile en considérant les fonctions objectifs de type boîte noire, les multiples types de paramètres de routage et le processus stochastique complexe d'un centre d'appels. Nous avons essayé différentes métaheuristiques qui sont décrites à la section 6.3. Nous avons retenu un algorithme génétique modifié (AGM), qui est présenté à la section 6.3.2, pour l'optimisation de toutes les politiques de routage. L'AGM a l'avantage de pouvoir optimiser les variables continues et discrètes, ainsi que les problèmes combinatoires. L'AGM optimise indirectement les variables en raffinant itérativement les paramètres des lois de probabilité, qui couvriraient (hypothétiquement) une solution optimale.

Les exemples numériques de la section 6.4 ont montré que notre politique BP était toujours parmi les meilleures politiques dans tous les exemples. Nous avons essayé des variantes de BP. La variante BP-S est une version simplifiée qui possède moins de paramètres. Théoriquement, la variante BP-S ne devrait pas être meilleure que BP. Cependant, comme l'optimisation des paramètres est un problème difficile, nous obtenions parfois une meilleure solution avec la version BP-S, car elle était plus simple à optimiser. Nous avons aussi testé BP-S2, qui est une version encore plus simplifiée que BP-S contenant encore

moins de paramètres. La variante BP-S2 n'a pas donné de bonnes performances.

Nous avons aussi testé des variantes de BP qui calculent les poids en fonction du nombre d'agents libres (BP-A et BP-AS). Ces variantes permettent d'imposer des délais aux appels en fonction du nombre d'agents libres, comme les politiques avec seuils d'agents libres PS et PDS. Les politiques BP-A et BP-AS ont excellé quand le centre d'appels avait des types d'appels avec de faibles volumes (et loin d'un régime à trafic intense). La politique BP était généralement meilleure que LG$c\mu$, surtout lorsque la fonction objectif pénalisait les niveaux de service. Cependant, la politique LG$c\mu$ performait souvent aussi bien que BP si le centre d'appels était dans un régime à trafic intense, ou si la fonction objectif pénalisait seulement les temps d'attente moyens ou les abandons.

Les résultats numériques ont montré que BP est une politique de routage utile et flexible pour différents modèles de centres d'appels. Cependant, l'optimisation des paramètres de routage représente un inconvénient sur l'aspect pratique. Avec notre approche de fonction boîte noire, l'optimisation est difficile et requiert généralement la simulation. Dans cette étude, nous nous sommes concentrés à démontrer l'utilité et les performances de la politique BP (une fois la politique optimisée). Une direction de recherche future est le développement d'un algorithme rapide et efficace pour optimiser les paramètres de routage. Nous pourrions utiliser les versions simplifiées de BP afin de trouver une solution initiale à l'optimisation de BP.

Chapitre 7

Planification basée sur l'agrégation et la théorie des files d'attente

La planification des agents pour les centres d'appels multi-compétences est beaucoup plus difficile que pour un centre avec un seul type d'appel, car la simulation est la seule véritable méthode pour obtenir des évaluations précises. Les algorithmes d'optimisation basés sur la simulation, comme Ceẓik et L'Ecuyer [37] et Avramidis et al. [17], demandent cependant beaucoup de temps d'exécution. Il pourrait survenir une situation où un gestionnaire aurait besoin d'une solution rapidement, et l'optimisation par simulation demanderait trop de temps. Il serait alors utile d'avoir un outil d'optimisation rapide, même si le résultat est grossier. La solution pourrait aussi être utilisée comme le point de départ d'une recherche locale.

Nous retrouvons quelques algorithmes de planification dans la littérature pour les centres d'appels multi-compétences qui n'utilisent pas la simulation (voir la section 3.3, page 33). La majorité de ces études se basent sur un régime à trafic intense et les modèles fluides pour optimiser des problèmes avec préférablement des centaines ou plus d'agents dans chaque groupe. Ces études supposent en général que la variance stochastique du centre d'appels est négligeable, et le problème d'optimisation devient déterministe. Bassamboo

et al. [20], Gurvich et Whitt [70] et Harrison et Zeevi [75] se basent sur les modèles fluides. D'autres algorithmes, comme Avramidis et al. [18] et Pot et al. [109], se basent sur des algorithmes d'approximation grossiers, mais rapides.

Dans ce chapitre, nous voulons optimiser le même problème d'affectation des agents dans un centre d'appels multi-compétences qu'Avramidis et al. [18] et Ceẓik et L'Ecuyer [37]. Nous présentons une nouvelle méthode d'affectation heuristique et rapide, basée sur l'agrégation et les formules d'Erlang, voir la section 3.2.1 à la page 30. Notre méthode se base sur le fait qu'avec la mesure de probabilité de délai, nous pouvons déterminer si un appel subit un délai ou non immédiatement à son instance d'arrivée. Nous n'avons pas besoin de mesurer les temps d'attente des appels. La probabilité de délai paraît un peu plus simple à approximer que le SL, et il existe souvent une corrélation négative entre le délai et le SL.

Dans notre approche, nous substituons les contraintes sur les SL par des contraintes sur les probabilités de délai. Cette substitution est heuristique puisque nous ne connaissons pas exactement la bonne équivalence entre le SL et la probabilité de délai, qui peut dépendre de beaucoup de paramètres comme la politique de routage. Cette substitution du SL par la probabilité de délai est un peu similaire à la méthode de Pot et al. [109] qui substitue grossièrement la fonction du SL, $g(\mathbf{y})$, par l'expression $1-b(\mathbf{y})$, où $b(\mathbf{y})$ est la probabilité de blocage et $\mathbf{y}$ est le vecteur d'agents. Leur méthode d'optimisation est cependant très différente à la nôtre.

L'idée de base de notre approche est diamétralement différente aux méthodes fluides, car notre idée repose directement sur les variations stochastiques du système. Notre méthode peut considérer des centres d'appels avec de grands ou petits volumes d'appels, ainsi que de grands ou petits groupes d'agents. Les modèles fluides [101] sont moins fiables pour ce type de problème, car la variance stochastique peut influencer significativement la solution optimale. Mais notre méthode ne marchera pas mieux que les méthodes basées sur les modèles fluides dans un régime à trafic intense, où la variance stochastique est négligeable. Ainsi, dans ce chapitre, nous nous concentrerons sur des problèmes de centres d'appels qui *ne sont pas* dans un régime à trafic intense.

Nous présentons le problème d'affectation des agents à optimiser à la section 7.1. Nous

décrivons notre méthode d'affectation heuristique basée sur l'agrégation, la théorie des files d'attente et la probabilité de délai à la section 7.2. Nous terminons ce chapitre avec quelques exemples numériques à la section 7.3.

7.1 Description du problème

Nous cherchons à optimiser le même problème d'affectation des agents que Ceẓik et L'Ecuyer [37] et Avramidis et al. [18] pour une seule période. Soit un centre d'appels composé de K types d'appels numérotés de 1 à K, et I groupes d'agents numérotés de 1 à I. L'ensemble d'habiletés du groupe i est $\mathcal{S}_i \subseteq \{1,\ldots,K\}$. L'ensemble des groupes pouvant servir le type d'appel k est $\mathcal{T}_k = \{i : k \in \mathcal{S}_i\}$. Afin d'utiliser les modèles de files d'attente, nous supposons que les arrivées suivent des processus de Poisson, et les durées de service et de patience sont des variables exponentielles. Pour le type d'appel k, le taux d'arrivée est λ_k, le temps moyen de service est μ_k^{-1} (indépendant du groupe), et le temps moyen de patience est ν_k^{-1}.

Notre méthode présente quelques similitudes avec les modèles fluides, voir la section 3.3.3 à la page 51, dont l'inconvénient d'avoir un routage contrôlé indirectement par le modèle. Autrement dit, la politique de routage est dynamique, car le système s'adapte automatiquement à la quantité de fluide. La politique de routage fluide n'est pas nécessairement réaliste non plus, et elle est souvent complexe à implanter. En général, les méthodes basées sur les modèles fluides proposent des politiques plus simples à implémenter qui approximent l'état de la solution fluide. Dans notre cas, nous devons également déterminer une politique de routage à notre solution d'affectation des agents.

Nous considérons des contraintes sur les niveaux de service f_{S}, définis par l'équation (2.1) à la page 20, par type d'appel et agrégé sur tous les types. Parce que notre méthode effectue des agrégations et désagrégations, il est préférable que le centre soit homogène (peu de différences entre les types, et les groupes) afin d'induire le moins d'erreurs. Nous supposons en particulier que tous les seuils $l = l_k$ sur les SL sont identiques. Soient c_i le coût d'un agent du groupe i et le vecteur d'agents $\mathbf{y} = (y_1, y_2, \ldots, y_I)^{\mathrm{T}}$. Puisque l'AWT est invariable dans nos problèmes, nous ré-définissons la formule du SL par $g(\mathbf{y})$ qui varie en

fonction du vecteur d'agents. Le problème à optimiser avec contraintes sur les SL est :

$$\begin{aligned} \min \quad & \sum_{i=1}^{I} c_i y_i \\ \text{sujet à :} \quad & g(\mathbf{y}) \geq l, \\ & g_k(\mathbf{y}) \geq l, \quad k = 1, \ldots, K, \\ & \mathbf{y} \geq 0 \text{ et entiers,} \end{aligned} \tag{P0}$$

où g et g_k sont les fonctions de SL agrégé et du type d'appel k. Remarquons que nous pouvons retirer la contrainte agrégée $g(\mathbf{y}) \geq l$, car elle devient redondante avec l'hypothèse $l_k = l$ pour tout k.

7.2 Optimisation stochastique basée sur les probabilités de délai

En général, la probabilité de délai est (un peu) plus simple à calculer que le SL. Nous proposons ici une méthode d'affectation heuristique basée sur les probabilités de délai. Nous traduisons heuristiquement les contraintes sur les SL dans (P0) par des contraintes sur les probabilités de délai. Nous supposons que les contraintes sont d'avoir une probabilité de délai inférieure ou égale à d. Nous appelons la fonction $g^{\mathrm{D}}(\mathbf{y})$ comme étant la fonction de la proportion de délai agrégée sur tous les appels, f_{D}, définie par l'équation (2.5) à la page 22. Nous définissons similairement la fonction $g_k^{\mathrm{D}}(\mathbf{y})$ comme la proportion de délai du type d'appel k. Notre problème d'affectation avec des contraintes sur les probabilités de délai est :

$$\begin{aligned} \min \quad & \sum_{i=1}^{I} c_i y_i \\ \text{sujet à :} \quad & g^{\mathrm{D}}(\mathbf{y}) \leq d, \\ & g_k^{\mathrm{D}}(\mathbf{y}) \leq d, \quad k = 1, \ldots, K, \\ & \mathbf{y} \geq 0 \text{ et entiers,} \end{aligned} \tag{P1}$$

Encore une fois, la contrainte agrégée est redondante ici.

Comme nous modélisons le centre d'appels par des processus markoviens, nous pouvons modéliser ce centre par une chaîne de Markov en temps continu (CMTC). La propriété

PASTA (“*Poisson Arrivals See Time Averages*”), voir Wolff [137], dit : “Soit X un état de la file d'attente avec une probabilité stationnaire π_X correspondant à la proportion du temps que le système passe dans X. À l'état stationnaire, la probabilité qu'un nouvel appel voit, à son arrivée, le système à l'état X est égale à π_X.” Ainsi, la probabilité de délai d'un appel est égale à la somme des probabilités des états où aucun agent n'est libre pour servir le nouvel appel. Sachant les probabilités stationnaires, nous pouvons optimiser le vecteur d'agents $\mathbf{y}$ de manière à servir tous les appels dans le système incluant le nouvel appel pour une fraction $(1-d)$ du temps, où d est la contrainte sur la probabilité le délai.

Puisque l'état du système est stochastique, nous utilisons la programmation stochastique [24] pour optimiser $\mathbf{y}$, où chaque état représente un scénario. Si la dimension du système est petit, nous pourrions optimiser sur tous les scénarios avec leurs probabilités stationnaires correspondantes. Mais étant donné que le nombre d'états est généralement trop grand, nous utilisons la simulation Monte Carlo pour générer M scénarios, et chaque scénario a une masse de probabilité égale à $1/M$ (loi uniforme discrète). Dans ce cas, la contrainte sur la probabilité de délai exige que tous les appels dans le centre doivent être affectés à des agents dans au moins $\lceil (1-d)M \rceil$ scénarios. C'est-à-dire que les files d'attente doivent être vides dans $\lceil (1-d)M \rceil$ scénarios ou plus. Nous exigeons la condition que toutes les files soient vides, car nous supposons que la politique de routage est équitable ; ce qui est plus réaliste aussi. Autrement, sans cette condition, la solution serait portée à utiliser un routage irréaliste. D'un scénario à l'autre, tous les agents serviraient certains types d'appels (sans délai) et aucun agent servirait les autres types d'appels (délai et abandons). Cette solution requerrait moins d'agents, mais elle produira beaucoup trop d'abandons.

Une difficulté importante se présente : nous ne connaissons pas les valeurs des probabilités stationnaires, car elles dépendent du vecteur d'agents $\mathbf{y}$ que nous désirons optimiser ! Nous allons générer heuristiquement les M scénarios à partir d'un centre d'appels simplifié où nous agrégeons les K types d'appels en un seul type, et les I groupes en un seul groupe. Cette simplification se base sur les observations de Wallace et Whitt [132]. En supposant que tous les taux de service sont identiques $\mu_k = \mu, \forall k$, ils observent qu'un centre d'appels, où tous les agents ont 2 ou 3 habiletés soigneusement choisies, peut performer presque aussi bien que si tous les agents avaient toutes les habiletés. Dans ce chapitre, nous

disons qu'un tel centre d'appels est *efficace*. Notre méthode présume l'hypothèse importante que le centre d'appels à optimiser est suffisamment efficace tel que nous pouvons approximer ses mesures de performance à partir d'un centre d'appels où tous les agents sont des généralistes possédant toutes les habiletés.

7.2.1 Processus de naissance et de mort

Nous résumons ici les résultats bien connus d'un système de file d'attente markovien avec un type d'appel et un groupe d'agents [77]. Les appels arrivent selon un processus de Poisson avec taux λ, les temps de service et de patience sont des variables exponentielles de moyennes μ^{-1} et ν^{-1} respectivement, la capacité de la file d'attente est h, et le nombre d'agents est y. Nous avons le processus de naissance et de mort classique d'une file $M/M/y+M$ avec le diagramme de transitions présenté à la figure 7.1 (inspirée de Hillier et Lieberman [77]), où le taux de mort de l'état s est :

$$\mu_s = \begin{cases} s\mu, & s = 1,\ldots,y, \\ y\mu + (s-y)\nu, & s = y+1,\ldots,y+h. \end{cases}$$

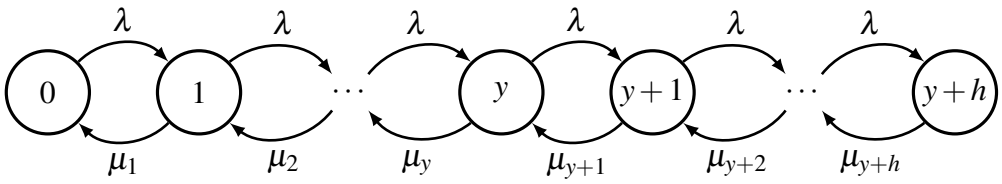


Figure 7.1 – Diagramme de transitions d'un processus de naissance et de mort.

Un état de la CMTC représente le nombre total d'appels (en service ou en attente) dans le centre d'appels. Plus précisément, l'état $x \in \{0,1,\ldots,y\}$ indique que tous les x appels sont en conversation avec un agent, et que la file d'attente est vide. L'état $x \in \{y+1,\ldots,y+h\}$ indique que tous les y agents sont occupés à servir des appels, et qu'il y a $x-y$ appels dans la file d'attente. Pour obtenir les probabilités stationnaires π_s des états $s = 0$ à $y+h$,

nous résolvons l'ensemble d'équations linéaires suivant :

$$\sum_{s=0}^{y+h} \pi_s = 1,$$
$$\pi_s = \frac{\lambda}{\mu_s}\pi_{s-1}, \quad \text{pour } s = 1,\ldots,y+h.$$

Un nouvel appel subit un délai si tous les y agents sont occupés à son arrivée, sauf au dernier état $y+h$ où l'appel sera bloqué. Cependant, nous incluons π_{y+h} dans la probabilité de délai, car la formule du délai f_{D} inclut les appels bloqués, voir l'équation (2.5) à la page 22. La probabilité de délai d'un appel est alors $\sum_{s=y}^{y+h} \pi_s$. Nous désirons trouver le nombre minimum d'agents y^* tel que les probabilités stationnaires satisfont $\sum_{s=y^*}^{y^*+h} \pi_s \leq d$. Notre algorithme utilise les formules d'Erlang C ou A, voir la section 3.2.1 à la page 30, selon la présence d'abandons, pour trouver y^*.

L'agrégation de tous les types d'appels et de tous les groupes d'agents est très heuristique. La politique de routage est aussi simplifiée à la politique du *premier arrivé, premier servi* et du *premier agent libre avant*. Notre approximation serait un peu plus précise si nous modélisions les K types d'appels dans la CMTC, mais le nombre d'états augmenterait exponentiellement. Une possibilité serait alors d'utiliser la simulation pour estimer les probabilités stationnaires π_s, mais nous n'avons pas essayé cela.

7.2.2 Générer les scénarios du problème stochastique

Comme nous l'avons mentionné auparavant, nous approximons le problème stochastique (P1) par M scénarios générés aléatoirement. Voici la procédure pour générer les M scénarios :

Étape 1 Agréger les paramètres :

$$\bar{\lambda} = \sum_{k=1}^{K} \lambda_k, \quad \bar{\mu} = \left(\sum_{k=1}^{K} \frac{\lambda_k/\mu_k}{\bar{\lambda}} \right)^{-1}, \quad \bar{\nu} = \sum_{k=1}^{K} \frac{\nu_k \lambda_k}{\bar{\lambda}}.$$

Le temps service moyen est agrégé par rapport aux durées, alors que la patience est agrégée par rapport aux taux. Ces formules d'agrégation sont plus conser-

vatrices, car la durée de service agrégée tend à être plus élevée, alors que le temps moyen de patience tend à être plus bas.

Étape 2 Considérons un centre d'appels avec un seul type d'appel et un groupe d'agents avec les paramètres $\bar{\lambda}, \bar{\mu}, \bar{\nu}$ et une capacité de file h. Trouver le nombre minimal d'agents y^* qui donne une probabilité de délai égale ou inférieure à d, à l'aide des formules d'Erlang C ou A, voir la section 3.2.1 à la page 30.

Étape 3 Calculer les probabilités stationnaires π_s des états $s = 0, 1, \ldots, y^* + h$ de la CMTC associée au centre d'appels agrégé.

Étape 4 Pour générer un scénario : sélectionner aléatoirement un état $s \in \{0, 1, \ldots, y^* + h\}$ avec probabilité π_s, et ajouter $+1$ pour inclure le nouvel appel. L'état s correspond au nombre total d'appels déjà dans le système, avant l'arrivée du nouvel appel.

Étape 5 Déterminer le type de chacun des $s+1$ appels du scénario. **Pour les s appels déjà présents :** choisir le type k avec une probabilité proportionnelle à la charge du trafic $\dfrac{\lambda_k/\mu_k}{\sum_{j=1}^{K} \lambda_j/\mu_j}$. **Pour le nouvel appel :** choisir le type k avec une probabilité proportionnelle au taux d'arrivé $\dfrac{\lambda_k}{\sum_{j=1}^{K} \lambda_j}$. Puisque nous désirons généralement peu d'abandons, nous supposons que le taux de patience a peu d'influence sur l'agrégation. Nous pourrions possiblement obtenir une meilleure distribution des appels, si nous simulons un centre avec plusieurs types d'appels et un seul groupe d'agents.

Les étapes 4 et 5 doivent être exécutées M fois. Nous définissons $\gamma_{k,m}$ comme le nombre d'appels du type k présents (incluant le nouvel appel s'il est du type k) dans le centre d'appels au scénario m.

Il est important à noter que les étapes heuristiques d'agrégation, puis de désagrégation sont basées sur l'hypothèse d'un centre d'appels efficace et d'une politique de routage équitable. Nous ne pouvons pas utiliser cette méthode pour optimiser l'affectation de tous les centres d'appels. Par exemple, supposons un centre d'appels avec $K = I$ types et groupes où le type k ne peut être servi que par le groupe k. Tous les agents sont alors des spécialistes dotés d'une seule habileté. Ce centre d'appels est simplement le regroupement de K centres

indépendants avec un type et un groupe. Clairement, il ne serait pas approprié d'agréger les processus stochastiques, ni même d'attribuer le terme *efficace* à ce centre d'appels.

7.2.3 Planification avec contrainte de délai

Nous reformulons le problème (P1) avec une contrainte par chance sur la *probabilité de délai quand un nouvel appel arrive*. Nous exigeons la règle d'équité telle que la probabilité de délai à tout instant d'arrivée ne dépend pas du type de l'appel, mais du nombre d'appels déjà présents dans le centre d'appels. Ceci découle de notre approximation par un centre d'appels agrégé avec un type d'appel et un groupe d'agents. Un nouvel appel subit un délai seulement si le nombre d'appels déjà présents est égal ou plus grand que le nombre total d'agents. Ainsi, nous avons une seule contrainte par chance, au lieu de K contraintes par chance. Couvrir la probabilité de délai de tous les types d'appels en même temps est plus contraignant. Dans le cas contraire (une contrainte par type), la solution aura tendance à être déséquilibrée en affectant tous les agents à certains types d'appels à un moment, puis à d'autres types à un autre moment. Ceci donnera une distribution irréaliste du travail des agents.

Nous avons le problème de planification stochastique suivant :

$$\begin{aligned} \min \quad & \sum_{i=1}^{I} c_i y_i \\ \text{sujet à :} \quad & \mathbb{P}\left[\begin{array}{ll} \sum_{i\in\mathcal{T}_k} w_{k,i} \geq \gamma_k, & \forall k, \\ \sum_{k\in\mathcal{S}_i} w_{k,i} \leq y_i, & \forall i, \\ w_{k,i} \geq 0 \text{ et entier}, & \forall k,i \end{array}\right] \geq (1-d), \\ & y_i \geq 0 \text{ et entier}, \quad \forall i, \end{aligned} \tag{P2}$$

où d est le seuil maximal sur la probabilité de délai, et γ_k est une variable aléatoire entière représentant le nombre d'appels du type k présents dans le centre d'appels *quand un nouvel appel arrive*. Notons que γ_k inclut le nouvel appel si celui-ci est du type k. La variable $w_{k,i}$ représente le nombre d'agents du groupe i affectés à servir un appel du type k. La variable $w_{k,i}$ est aussi aléatoire, car elle dépend des nombres d'appels $\gamma_1,\ldots,\gamma_K$. Similairement aux

modèles fluides, ce problème suppose que le routage est dynamique et s'adapte en fonction du nombre d'appels présents. Ce routage n'est pas nécessairement réaliste.

Nous optimisons un problème approximatif échantillonné (ou "*sample average problem*") en générant M scénarios. Un scénario représente un instant d'arrivée d'un nouvel appel, et $\gamma_{k,m}$ est le nombre d'appels du type k (incluant le nouvel appel s'il est du type k) présents dans le centre d'appels lors du scénario m. La procédure pour générer les $\gamma_{k,m}$ est donnée à la section 7.2.2. Le problème (P2) devient le problème linéaire suivant :

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{I} c_i y_i - p \sum_{k=1}^{K} \sum_{i=1}^{I} \sum_{m=1}^{M} w_{k,i,m} \\
\text{sujet à :} \quad & \sum_{k \in \mathcal{S}_i} w_{k,i,m} \leq y_i, && \forall i,m, \\
& \sum_{i \in \mathcal{T}_k} w_{k,i,m} \geq b_m \gamma_{k,m}, && \forall k,m, \\
& \sum_{m=1}^{M} b_m \geq (1-d)M, \\
& b_m \in \{0,1\}, && \forall m, \\
& w_{k,i,m} \geq 0, && \forall k,i,m, \\
& y_i \geq 0 \text{ et entier}, && \forall i,
\end{aligned}
\tag{P3}
$$

où $p > 0$ est un petit coût afin que l'optimisation choisisse le vecteur d'agents $\mathbf{y}$ pouvant travailler le plus (c'est-à-dire de maximiser les $w_{k,i,m}$), s'il y a plusieurs vecteurs $\mathbf{y}$ optimaux. La contrainte par chance dans (P2) est remplacée par M variables binaires : $b_1, \ldots, b_M$. La variable binaire $b_m = 1$ si le scénario m est sans délai (tous les appels affectés à des agents), et $b_m = 0$ sinon. Remarquons que les coefficients des variables $w_{k,i,m}$ sont 1 partout, et que les y_i et $\gamma_{k,m}$ sont tous entiers. Donc, il devrait exister une solution optimale telle que toutes les variables $w_{k,i,m}$ sont entières, ceci **même sans** expliciter les contraintes d'intégralité.

Étant donné qu'il faut générer beaucoup de scénarios, le temps d'optimisation du problème à nombres entiers est généralement trop élevé pour des centres d'appels avec plus de 3 types et 3 groupes. Nous relaxons les contraintes d'intégralité des variables y_i et b_m. Nous proposons deux méthodes de relaxation : méthode d'arrondissement itérative (AI) et méthode de relaxation lagrangienne simplifiée (RLS).

7.2.4 Méthode d'arrondissement itérative (AI)

Nous commençons par relaxer toutes les variables entières du problème (P3). Puis, nous exécutons un algorithme itératif qui arrondit les variables binaires b_m. À chaque itération, nous mettons à jour trois partitions de scénarios : $\mathcal{M}$ contient les scénarios pas encore fixés, $\mathcal{M}_1 = \{m : b_m = 1\}$ et $\mathcal{M}_0 = \{m : b_m = 0\}$. Nous avons $\mathcal{M} \cup \mathcal{M}_1 \cup \mathcal{M}_0 = \{1, 2, \ldots, M\}$, et ces trois ensembles sont mutuellement disjoints. L'algorithme se termine lorsque la taille de l'ensemble $\mathcal{M}_1$ est égale ou plus grande que $(1-d)M$. Les scénarios restants $m \in \mathcal{M}$ sont alors ajoutés à $\mathcal{M}_0$, et les b_m correspondants sont fixés à 0. Le problème relaxé est :

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{I} c_i y_i - p \sum_{k=1}^{K} \sum_{i=1}^{I} \sum_{m=1}^{M} w_{k,i,m} - q \sum_{m \in \mathcal{M}} b_m \\
\text{sujet à :} \quad & \sum_{k \in \mathcal{S}_i} w_{k,i,m} \leq y_i, && \forall i, m, \\
& \sum_{i \in \mathcal{T}_k} w_{k,i,m} \geq \gamma_{k,m}, && \forall k, m \in \mathcal{M}_1, \\
& \sum_{i \in \mathcal{T}_k} w_{k,i,m} \geq b_s \gamma_{k,m}, && \forall k, m \in \mathcal{M}, \\
& \sum_{s \in \mathcal{M}} b_m \geq (1-d)M - |\mathcal{M}_1|, \\
& 0 \leq b_m \leq 1, && \forall m \in \mathcal{M}, \\
& y_i, w_{k,i,m} \geq 0, && \forall k, i, m,
\end{aligned}
\tag{P4a}
$$

où $|\mathcal{M}_1|$ est la taille de l'ensemble $\mathcal{M}_1$, et $p, q > 0$ sont de petits coûts positifs. Nous ajoutons le coût q pour maximiser les variables b_m non fixées afin de déterminer les prochains b_m à arrondir. Les contraintes appartenant aux scénarios $m \in \mathcal{M}_0$ sont retirées du problème (P4a). Voici la procédure de la méthode AI :

Étape 1 Débuter avec $\mathcal{M} = \{1, 2, \ldots, M\}$ et $\mathcal{M}_0 = \mathcal{M}_1 = \emptyset$ (toutes les variables b_m sont non fixées).

Étape 2 Résoudre le problème linéaire relaxé (P4a).

Étape 3 Si $|\mathcal{M}_1| \geq (1-d)M$, alors **terminer**.

Étape 4 Choisir le meilleur scénario $m^* = \arg\max_j \{b_j : j \in \mathcal{M}\}$ et fixer $b_{m^*} = 1$. Retirer m^* de $\mathcal{M}$ et l'ajouter à $\mathcal{M}_1$.

Étape 5 Si $|\mathcal{M}_1| \geq (1-d)M$, transférer tous les $m \in \mathcal{M}$ vers $\mathcal{M}_0$ et fixer les $b_m = 0$.

Étape 6 Aller à l'étape 2.

Une approche conservatrice est de fixer une seule variable $b_m = 1$ à la fois à l'étape 4. Il faudra exécuter $\lceil (1-d)M \rceil$ itérations, ce qui peut demander beaucoup de temps d'exécution.

Remarquons qu'à la fin de l'algorithme (quand tous les b_m sont fixés à 0 ou 1), tous les coefficients des contraintes sont à 1, et toutes les constantes sont entières dans (P4a). Alors, la méthode du simplexe, qui visite les points extrêmes de l'enveloppe convexe du domaine des solutions, retournera une solution optimale entière. Donc, nous n'avons pas besoin d'arrondir les variables **y**.

Pour accélérer l'algorithme AI, nous proposons deux heuristiques qui réduisent la taille de $\mathcal{M}$ au moment de l'initialisation :

1. Soit y^* le nombre minimal d'agents requis pour le centre d'appels agrégé, utilisé pour générer les M scénarios. Nous trouvons y^* à l'aide des formules d'Erlang. Ajouter la contrainte : $\sum_{i=1}^{I} y_i \geq (y^* - \delta)$ au problème (P4a), où le paramètre δ sert à contrôler l'agressivité ou le conservatisme de la contrainte. Cette contrainte est plus conservatrice quand δ est grand. Le raisonnement est que le centre d'appels ne peut pas nécessiter moins d'agents qu'un centre où tous les agents sont des généralistes (en supposant des taux de service indépendants des agents, comme dans Wallace et Whitt [132]). Nous utilisons $\delta = 1$ dans nos exemples numériques.

2. Il y a une forte chance que les scénarios ayant un fort volume d'appels subiront un délai, et les scénarios avec un faible volume seront couverts (sans délai). Une heuristique rapide est de trier les scénarios par volume total d'appels. Puis, fixer $b_m = 1$ pour une fraction z_1 des scénarios avec les plus bas volumes, et fixer $b_m = 0$ pour une fraction z_0 des scénarios avec les plus forts volumes. Une approche conservatrice serait de fixer seulement une fraction $z_0 > 0$ des b_m à 0, et de ne pas fixer de b_m à 1 (donc $z_1 = 0$). Le raisonnement est que fixer un mauvais b_m à 1 peut être désastreux, car l'algorithme sera forcé à satisfaire ce mauvais scénario, et nous ne modifions pas les b_m une fois qu'ils sont fixés. Un scénario m est *mauvais* s'il n'est pas optimal d'avoir $b_m = 1$.

Nous avons aussi des heuristiques pour améliorer la vitesse durant l'algorithme AI :

1. À la fin de l'étape 2 (résoudre le problème relaxé), fixer automatiquement tous les b_m à 1 tels que $b_m \geq (1-\varepsilon), m \in \mathcal{M}$ et $\varepsilon > 0$ est un petit nombre. Autrement dit, arrondir à 1 toutes les variables b_m non fixées qui sont très proches de 1.
2. Au lieu choisir systématiquement le plus grand b_m non fixé à l'étape 4, tester les V plus grands candidats b_m (non fixés), puis choisir le candidat qui donne le plus bas coût. Il faut éviter de tester les scénarios identiques. Cette heuristique peut aider à éviter un mauvais choix d'arrondissement, mais le temps d'exécution est multiplié par un facteur V puisqu'il faut résoudre un problème linéaire pour chaque candidat. Comme l'algorithme AI est rapide, nous utilisons $V = 5$ dans nos exemples numériques.

7.2.5 Méthode de relaxation lagrangienne simplifiée (RLS)

En regardant le problème (P3), la contrainte importante est de trouver les scénarios à fixer $b_m = 1$. Une fois que les variables binaires b_m sont optimisées et fixées dans (P3), il devient facile d'optimiser le vecteur d'agents $\mathbf{y}$. La méthode du simplexe retournera une solution entière $\mathbf{y}$, car tous les coefficients sont égaux à 1 et les constantes $\gamma_{k,m}$ sont entières.

Nous appliquons la relaxation lagrangienne sur la contrainte $\sum_{m=1}^{M} b_m \geq (1-d)M$ du problème (P3). Puisque nous n'avons qu'un seul multiplicateur de Lagrange, nous allons optimiser une version simplifiée sans inclure la constante $(1-d)M$ dans la fonction objectif. Nous appelons ce problème (P4b(q)), car nous voulons optimiser le multiplicateur de Lagrange q.

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{I} c_i y_i - p \sum_{k=1}^{K} \sum_{i=1}^{I} \sum_{m=1}^{M} w_{k,i,m} - q \sum_{m=1}^{M} b_m, \\
\text{sujet à :} \quad & \sum_{k \in \mathcal{S}_i} w_{k,i,m} \leq y_i, && \forall i,m, \\
& \sum_{i \in \mathcal{T}_k} w_{k,i,m} \geq b_m \gamma_{k,m}, && \forall k,m, \\
& y_i, w_{k,i,m} \geq 0, && \forall k,i,m, \\
& 0 \leq b_m \leq 1, && \forall m.
\end{aligned}
\tag{P4b(q)}
$$

Nous commençons par optimiser les variables binaires b_m avec une méthode similaire à l'optimisation du dual lagrangien. Il y a deux importants paramètres de coût dans la fonction objectif : le coût des agents $c_i \geq 0$ et le "profit" $q > 0$ de couvrir un scénario. Nous gardons le coût $p \geq 0$ négligeable par rapport à c_i et q. La fonction objectif désire minimiser le coût des agents, mais elle veut aussi maximiser le profit de couvrir les scénarios. Le but est de trouver un équilibre entre les coûts des agents et le profit de couvrir les scénarios. Pour faire cela, nous optimisons le multiplicateur de Lagrange q à l'aide d'une recherche binaire. Nous cherchons le plus petit $q^* > 0$ tel que la solution optimale à (P4b(q^*)) aura un nombre de $(b_m = 1)$ égal ou supérieur à $(1-d)M$. Puis, nous optimisons **y** une fois que les variables b_m sont optimisées et fixées.

Voici l'algorithme RLS :

Étape 1 Initialisation : trouver $q^+ > q^- > 0$, tels que (P4b(q^+)) donne une solution couvrant au moins $(1-d)M$ scénarios, c'est-à-dire $\sum_{m=1}^{M} \mathbb{I}[b_m = 1] \geq (1-d)M$, et que (P4b($q^-$)) donne une solution couvrant moins que $(1-d)M$ scénarios. La fonction indicatrice $\mathbb{I}[b_m = 1]$ retourne 1 si $b_m = 1$, et 0 sinon.

Étape 2 Recherche binaire : assigner $\hat{q} = (q^+ + q^-)/2$, et résoudre (P4b($\hat{q}$)).

Étape 3 Mise à jour : si (P4b($\hat{q}$)) couvre au moins $(1-d)M$ scénarios, alors attribuer $q^+ = \hat{q}$, sinon $q^- = \hat{q}$.

Étape 4 Si $q^+ - q^- < \varepsilon$ (pour un petit $\varepsilon > 0$), alors affecter $q^* = q^+$ et optimiser (P4b(q^*)). Arrondir et fixer les b_m selon la solution optimale à (P4b(q^*)), puis optimiser (P3) (le coût p n'est plus important puisque tous les b_m sont fixés). **Terminer**.

Étape 5 Aller à l'étape 2.

En général, nous suggérons d'utiliser la méthode RLS lorsqu'il y a beaucoup de scénarios. Les désavantages de la méthode AI sont sa lenteur et la possibilité d'arrondir une mauvaise variable b_m à 1. Grâce à la recherche binaire, le temps d'exécution de la méthode RLS n'augmente pas aussi rapidement qu'avec la méthode AI. Par contre, nous avons observé dans des tests numériques que la méthode AI avec les améliorations heuristiques est tout aussi rapide que la méthode RLS. Nous utilisons la méthode AI avec $M = 2000$ scénarios dans nos expériences numériques.

7.2.6 Approximer la politique de routage

Comme les méthodes basées sur les modèles fluides, un inconvénient majeur est que nous *supposons* qu'une politique de routage idéale existe. Cette politique est implicite dans le modèle et elle nous est malheureusement inconnue. Cette politique peut être aussi non réaliste dans la pratique. Par exemple, les services pourraient ne pas respecter la règle de non-préemption, car les allocations des tâches $w_{k,i,m}$ peuvent être très différentes d'un scénario à l'autre, même si nous n'ajoutons qu'un seul appel de plus. Comme les autres méthodes basées sur les modèles fluides [70, 75], nous essayons d'approximer le routage optimal à l'aide d'une politique plus simple. En pratique, un gestionnaire préfère souvent que les taux d'occupation des agents soient balancés entre les groupes. Soient $\bar{\mathbf{y}}$ la solution optimale à (P3), $\mathcal{M}_1$ l'ensemble des scénarios sans délai et $\mathcal{M}_0 = \{1,\dots,M\} \setminus \mathcal{M}_1$ l'ensemble des scénarios avec délai. Nous voulons optimiser l'allocation des tâches $w_{k,i,m}$:

$$
\begin{aligned}
\min \quad & r(o_{\max} - o_{\min}) + t \sum_{k=1}^{K} \sum_{m \in \mathcal{M}_0} v_{k,m} \\
\text{sujet à :} \quad & \sum_{k \in \mathcal{S}_i} w_{k,i,m} \leq \bar{y}_i, && \forall i, m, \\
& \sum_{i \in \mathcal{T}_k} w_{k,i,m} = \gamma_{k,m}, && \forall k, m \in \mathcal{M}_1, \\
& v_{k,m} \geq \gamma_{k,m} - \sum_{i \in \mathcal{T}_k} w_{k,i,m}, && \forall k, m \in \mathcal{M}_0, \\
& o_i = \frac{1}{M\bar{y}_i} \sum_{k \in \mathcal{S}_i} \sum_{m=1}^{M} w_{k,i,m}, && \forall i, \\
& o_{\max} \geq o_i \geq o_{\min}, && \forall i, \\
& w_{k,i,m}, v_{k,m}, o_{\max}, o_{\min} \geq 0, && \forall k, i, m,
\end{aligned}
\tag{P5}
$$

où r est le coût de pénalité sur l'écart d'occupation, t est un petit coût de pénalité sur les appels qui subissent un délai, o_i est le taux d'occupation moyen du groupe i, et $v_{k,m}$ est le nombre d'appels du type k qui subissent un délai dans le scénario $m \in \mathcal{M}_0$. Les variables $o_{\max}$ et $o_{\min}$ représentent les taux d'occupation maximal et minimal parmi les I groupes d'agents. Les variables $o_{\max}, o_{\min}$ et o_i sont restreintes à des valeurs de l'intervalle $[0,1]$. L'objectif est de minimiser l'écart entre $o_{\max}$ et $o_{\min}$, mais nous pénalisons aussi les appels

avec délai. La raison du coût t est pour éviter d'avoir des agents qui restent libres lorsqu'il y a des appels en attente qu'ils peuvent répondre (ce qui n'est pas très réaliste en général).

Nous pourrions aussi reformuler le problème (P5) afin de balancer les taux d'occupation de *chaque scénario* au lieu du taux d'occupation moyen. Il y aurait une variable d'occupation par scénario, soient $o_{\max,m}, o_{\min,m}$ et $o_{i,m}$, et nous aurons le problème suivant :

$$\begin{aligned}
\min \quad & r \sum_{m=1}^{M} (o_{\max,m} - o_{\min,m}) + t \sum_{k=1}^{K} \sum_{m \in \mathcal{M}_0} v_{k,m} \\
\text{sujet à :} \quad & \sum_{k \in \mathcal{S}_i} w_{k,i,m} \leq \bar{y}_i, && \forall i, m, \\
& \sum_{i \in \mathcal{T}_k} w_{k,i,m} = \gamma_{k,m}, && \forall k, m \in \mathcal{M}_1, \\
& v_{k,m} \geq \gamma_{k,m} - \sum_{i \in \mathcal{T}_k} w_{k,i,m}, && \forall k, m \in \mathcal{M}_0, \\
& o_{i,m} = \sum_{k \in \mathcal{S}_i} w_{k,i,m} / \bar{y}_i, && \forall i, m, \\
& o_{\max,m} \geq o_{i,m} \geq o_{\min,m}, && \forall i, m, \\
& w_{k,i,m}, v_{k,m}, o_{\max,m}, o_{\min,m} \geq 0, && \forall k, i, m,
\end{aligned} \tag{P5a}$$

Notons que nous n'optimisons pas le problème (P5a) dans les exemples numériques.

Supposons que la solution du problème (P5) soit $\bar{w}_{k,i,m}$. Nous pouvons essayer de construire une politique de routage qui s'approcherait à ces performances moyennes :

- **Allocation moyenne des tâches** du groupe i sur le type k : $\bar{w}_{k,i} = \sum_{m=1}^{M} \bar{w}_{k,i,m}/M$.
- **Ratio moyen d'occupation** du groupe i sur les appels du type k : $\dfrac{\bar{w}_{k,i}}{\sum_{k=1}^{K} \bar{w}_{k,i}}$.
- **Ratio moyen d'appels servis** du type k par le groupe i : $\dfrac{\bar{w}_{k,i}}{\sum_{i=1}^{I} \bar{w}_{k,i}}$.

Les résultats des simulations montrent que ces estimations sont des mesures grossières par rapport aux performances mesurées par le simulateur. Ces performances dépendent grandement de la politique de routage utilisée. Nous obtenons de bonnes performances en utilisant une politique de routage indépendante des allocations $\bar{w}_{k,i,m}$, soit le routage basé sur des poids BP, présenté à la section 6.2.4, page 163. Nous comparons quelques politiques de routage dans la section numérique.

7.3 Exemples numériques

L'objectif de ces expériences est de tester l'efficacité de notre méthode de planification basée sur l'agrégation, l'approximation de files d'attente et la probabilité de délai pour optimiser l'affectation des agents avec des contraintes sur les SL (et non sur les probabilités de délai) du problème (P0), présenté à la section 7.1. Comme notre méthode optimise les agents sans considérer le routage, il est important de comparer les performances simulées avec une politique de routage concrète et les performances attendues par notre méthode.

Tous nos exemples numériques sont des centres d'appels définis par des processus markoviens et ne possèdent qu'une seule période. Pour le type d'appel k, le processus d'arrivée est Poisson avec un taux λ_k, et les durées de service sont des variables exponentielles de moyenne μ_k^{-1} (et ne dépendent pas du groupe). Les durées de patience sont des variables exponentielles de moyenne de $\nu_k^{-1} = 1/10$ heure pour tous les types d'appels. Tous les taux sont exprimés par unité d'heure. La structure de coût est identique à celle de Ceşik et L'Ecuyer [37], soit $c_i = 1 + 0.1(|\mathcal{S}_i| - 1)$ où $|\mathcal{S}_i|$ est le nombre d'habiletés du groupe i. Étant donné que notre méthode se base sur l'idée d'un centre d'appels *efficace*, les ensembles d'habiletés sont généralement bien balancés. L'AWT est de $\tau_k = \tau = 20$ secondes et les seuil de SL sont $l_k = l = 80\%$ pour tous les types d'appels et agrégé globalement. Nous utilisons la méthode d'optimisation basée sur les probabilités de délai pour résoudre le problème d'affectation (P0). Nous convertissons la cible de 80% du SL en probabilité de délai à l'aide des formules d'Erlang (voir la section 3.2.1 à la page 30) et du centre d'appels agrégé.

Nous optimisons le problème stochastique relaxé avec $M = 2000$ scénarios et la méthode AI présentée à la section 7.2.4, avec les heuristiques d'accélération. Les problèmes linéaires sont résolus à l'aide de la librairie CPLEX version 12. Nous simulons les solutions, et nous comparons les mesures de performance par rapport aux cibles de probabilités de délai et de SL du problème. Nous désirons estimer les performances du centre d'appels sur un horizon de temps infini. Les solutions finales sont simulées avec la méthode des moyennes par lots (ou "*batch means*"). Pour chaque solution, le simulateur exécute une seule longue simulation de 7000 heures d'opération du centre d'appels. Les 2000 pre-

mières heures servent à *réchauffer* la simulation (pour réduire les effets transitoires du centre d'appels vide initial), et aucune donnée statistique n'est recueillie durant cette période. Les 5000 heures suivantes sont divisées en 5 lots de 1000 heures, et nous considérons chaque lot comme une réplication de simulation (suffisamment) indépendante. Les mesures de performance sont estimées à partir de ces 5 réplications.

Dans les tableaux de résultats, les proportions de délai et les SL sont exprimés en pourcentage (%). Notons que la mesure de la *proportion de délai* f_{D} est équivalente à la mesure de la *probabilité de délai* dans notre contexte. La proportion de délai agrégée sur tous les appels est donnée sous la colonne "f_{D}", et le SL agrégé est donné sous la colonne "SL". Nous présentons aussi la différence entre les SL maximal et minimal des types d'appels sous la colonne "ΔSL".

Pour déterminer la qualité de notre méthode, nous comparons : (1) la proportion de délai agrégée simulée avec la proportion de délai agrégée ciblée, (2) le SL agrégé simulé avec le seuil $l = 80\%$, et (3) la distribution des SL par type d'appel autour du seuil $l_k = 80\%$ (mesurée par ΔSL). Rappelons-nous que l'affectation des agents est optimisée en supposant un routage optimal. Pour vérifier les mesures de performance de nos solutions, nous devons obligatoirement spécifier une politique de routage concrète au simulateur. Comme les mesures de performance peuvent varier selon le routage, nous avons testé deux politiques de routage : la politique de routage à priorités égales E (voir la section 2.2.1 à la page 18), et une politique similaire à la politique E, mais le routage est conditionnel au respect du **ratio moyen d'appels servis** défini à la section 7.2.6. Un groupe ne peut pas servir le type d'appel k quand le ratio d'appels du type k servis par ce groupe est supérieur à ce ratio moyen d'appels servis.

7.3.1 Exemple 1 : petit centre d'appels

Nous avons un centre d'appels avec $K = 3$ types et $I = 3$ groupes d'agents. Tous les agents ont deux habiletés : $\mathcal{S}_1 = \{1,2\}, \mathcal{S}_2 = \{2,3\}$ et $\mathcal{S}_3 = \{1,3\}$. Nous testons 5 différents jeux de taux d'arrivée et de service. Le tableau 7.I présente les résultats. Les solutions $\mathbf{y}^*$ obtenues pour les tests 1 à 5 sont données dans le tableau 7.II.

Les colonnes ΔSL montrent que le routage qui respecte les ratios donne des SL beau-

Test	Taux d'arrivée $(\lambda_1, \lambda_2, \lambda_3)$	Taux de service (μ_1, μ_2, μ_3)	f_D ciblée	Routage E			Routage ratio		
				f_D	SL	ΔSL	f_D	SL	ΔSL
1	(500, 500, 500)	(10, 10, 10)	45.0	44.4	79.8	2.6	44.6	78.6	1.2
2	(1000, 500, 200)	(10, 10, 10)	46.0	45.3	79.6	8.3	45.2	79.1	2.6
3	(1000, 500, 200)	(12, 8, 10)	46.0	45.8	79.1	10.9	44.8	79.2	1.6
4	(100, 500, 100)	(12, 8, 10)	36.0	33.2	81.2	12.9	36.0	79.2	5.2
5	(100, 500, 100)	(10, 10, 10)	36.0	33.6	81.4	13.6	35.6	79.2	1.9

Tableau 7.I – Exemple 1 : centre d'appels avec 3 types et 3 groupes. Résultats simulés avec deux politiques de routage différentes.

coup mieux balancés qu'avec la politique de routage E. Cependant, le SL agrégé est légèrement plus bas, et la proportion de délai est un peu plus grande. Il est intéressant de voir que toutes les solutions sont proches de la cible de 80% pour le SL agrégé, ainsi que la proportion de délai agrégée ciblée.

Notre méthode AI avec les heuristiques d'accélération a eu besoin que deux itérations par problème, et le temps d'exécution est autour de 40 secondes. Nous comparons nos solutions avec celles trouvées par l'algorithme de coupes et la simulation de Ceşik et L'Ecuyer [37] (CP). Comme leur méthode n'optimise pas le routage, il faut choisir une politique de routage au préalable. Nous choisissons la politique E. Nous utilisons les mêmes paramètres de simulations (7000 heures d'opération simulées) pour l'algorithme CP. CP exécute peu d'itérations, et le temps d'optimisation est inférieur à 2 minutes.

Test	Notre méthode				CP			
	Solution	SL	Min SL_k	coût	Solution	SL	Min SL_k	coût
1	(60, 59, 33)	79.8	78.6	167.2	(0, 76, 78)	83.1	80.7	169.4
2	(25, 54, 93)	79.6	77.5	189.2	(4, 61, 109)	82.1	80.8	191.4
3	(87, 64, 17)	79.1	74.8	184.8	(91, 0, 79)	84.9	80.3	187.0
4	(10, 65, 10)	81.2	73.1	93.5	(0, 69, 18)	83.4	80.5	95.7
5	(12, 53, 9)	81.4	71.6	81.4	(4, 55, 16)	82.8	80.0	82.5

Tableau 7.II – Exemple 1 : comparaisons des solutions de notre méthode avec la méthode par coupes et la simulation CP. La politique de routage à priorités égales E est utilisée.

Le tableau 7.II compare les résultats entre notre méthode approximative et CP. Ce tableau donne le vecteur d'affectation trouvé, le SL agrégé, le plus petit SL par type ("Min

SL_k") et le coût des agents pour chaque test et chaque méthode. Les vecteurs d'affectation des tests 1 à 3 sont très différents, alors que les solutions pour les test 4 et 5 se ressemblent. Les coûts de nos solutions sont un peu plus bas, mais les SL aussi. Comme notre méthode est une *approximation*, les contraintes sur les SL ne sont pas nécessairement satisfaites à cause des erreurs d'approximation. De plus, notre méthode n'a pas optimisé le problème en supposant la politique de routage E. Il faudrait appliquer une recherche locale à la fin de notre optimisation. CP utilise la simulation et ses solutions satisfont toutes les contraintes. Les résultats montrent que notre méthode peut être utile, malgré les erreurs d'approximation.

7.3.2 Exemple 2 : centre d'appels moyen

Nous considérons un centre d'appels avec $K = 5$ types et $I = 8$ groupes d'agents. Les groupes 1 à 5 ont chacun deux habiletés : $\mathcal{K}_1 = \{1,2\}, \mathcal{K}_2 = \{2,3\}, \mathcal{K}_3 = \{3,4\}, \mathcal{K}_4 = \{4,5\}$ et $\mathcal{K}_5 = \{1,5\}$. Les trois autres groupes ont chacun 3 habiletés : $\mathcal{K}_6 = \{1,3,4\}, \mathcal{K}_7 = \{2,3,5\}$ et $\mathcal{K}_8 = \{1,4,5\}$. Nous testons quatre différents paramètres de taux d'arrivée et de taux de service. Le tableau 7.III présente les résultats. La proportion de délai ciblée est de 52% pour les 4 tests. Les solutions $\mathbf{y}^*$ pour les tests 1 à 4 sont : (56, 43, 47, 51, 53, 0, 0, 0), (71, 49, 32, 15, 83, 0, 0, 0), (49, 40, 52, 59, 55, 0, 0, 0), (61, 33, 58, 13, 85, 0, 0, 0), et les coûts respectifs sont : 275, 275, 280.5 et 275. Remarquons que toutes les solutions n'ont aucun agent avec 3 habiletés. L'optimisation trouve qu'il est moins dispendieux et suffisant d'affecter uniquement des agents avec deux habiletés. Comme dans l'exemple 1, nous simulons les solutions avec deux politiques de routage différentes. Il est intéressant d'observer que les mesures agrégées de la proportion de délai et du SL sont proches de leurs cibles de 52% et 80%, respectivement. Par contre, les SL peuvent varier grandement d'un type d'appel à l'autre. L'écart de SL s'élève jusqu'à 23% dans le test 4 ! Nous essaierons de mieux balancer les SL à l'aide d'une politique de routage plus sophistiquée dans la prochaine section.

Test	Taux d'arrivée $(\lambda_1,\ldots,\lambda_5)$	Taux de service $(\mu_1,\ldots,\mu_5)$	Routage E			Routage ratio		
			f_D	SL	ΔSL	f_D	SL	ΔSL
1	(500, 500, 500, 500, 500)	(10, 10, 10, 10, 10)	50.5	79.6	6.0	50.6	78.5	1.0
2	(1000, 700, 500, 200, 100)	(10, 10, 10, 10, 10)	50.6	78.7	17.8	51.3	77.0	8.2
3	(500, 500, 500, 500, 500)	(12, 11, 10, 9, 8)	50.6	79.4	5.2	50.8	76.0	6.2
4	(1000, 700, 500, 200, 100)	(10, 12, 8, 11, 9)	51.4	77.5	17.1	50.5	77.2	23.0

Tableau 7.III – Exemple 2 : centre d'appels avec 5 types et 8 groupes. Résultats simulés avec deux politiques de routage différentes. La proportion de délai ciblée est de 52% pour les 4 tests.

7.3.3 Politique de routage basé sur des poids (BP)

Étant donné que nous ne connaissons pas exactement la politique de routage du modèle (nous supposons seulement qu'elle existe), nous choisissons ici la politique de routage basé sur des poids BP, présentée à la section 6.2.4, page 163. Cette politique est flexible et les paramètres de routage sont optimisés de manière heuristique par un algorithme génétique. L'optimisation du routage est exécutée une fois que l'affectation optimale $\mathbf{y}^*$ est trouvée (optimisée en supposant un routage idéal). L'optimisation des paramètres de routage requiert la simulation et ceci peut demander plus de temps que l'optimisation des agents !

Pour l'optimisation du routage, nous simulons avec la méthode des moyennes par lots comme pour les exemples 1 et 2. Par contre, la longueur de simulation est plus courte durant l'optimisation. Le simulateur simule 700 heures d'opérations, et les données statistiques sont calculées à partir des 500 dernières heures, divisées en 5 lots de 100 heures chacun. Les résultats finaux sont obtenus en simulant un total de 7000 heures d'opération (et seulement les 5000 dernières heures comptent), comme pour les exemples 1 et 2.

Les tableaux 7.IV et 7.V présentent les performances des solutions des exemples 1 et 2 avec la politique BP. Les paramètres du routage sont optimisés pour minimiser la fonction objectif suivante :

$$F_1 = \sum_{k=1}^{K} \max(l_k - g_k(\mathbf{y}^*), 0)^2,$$

où l_k est le seuil minimal du SL du type d'appel k. Nous avons $l_k = 80\%$ dans nos exemples. La fonction F_1 ne considère pas le SL agrégé, car ce dernier n'est pas nécessaire dans nos

exemples. Il n'y a pas de pénalité ni de profit lorsque le SL est au-dessus du seuil l_k. L'exposant 2 sert à mieux balancer les écarts de pénalité entre les types d'appels. Nous choisissons de ne pas pondérer les pénalités par les taux d'arrivée.

Les colonnes "min SL" et "max SL" montrent les plus petits et les plus grands SL parmi tous les types d'appels. Nous observons que les SL obtenus avec la politique BP sont généralement mieux balancés que ceux obtenus avec les routages E et par ratio dans les exemples précédents. Il est quelques écarts qui sont grands, parce que F_1 ignore les SL qui sont au-dessus du seuil de 80%. Les SL agrégés sont aussi un peu plus élevés avec la politique BP. Les proportions de délai demeurent également proches des cibles.

Test	f_D ciblée	f_D	SL	ΔSL	min SL	max SL
1	45.0	44.5	79.9	0.7	79.5	80.2
2	46.0	45.2	79.7	0.3	79.6	79.9
3	46.0	46.1	81.4	5.5	80.7	86.2
4	36.0	34.2	83.3	2.1	81.5	83.6
5	36.0	34.7	83.4	3.2	81.1	84.3

Tableau 7.IV – Exemple 1 : routage basé sur des poids et la fonction objectif F_1.

Test	f_D ciblée	f_D	SL	ΔSL	min SL	max SL
1	52.0	50.7	79.6	0.3	79.5	79.8
2	52.0	50.7	80.5	6.1	78.4	84.5
3	52.0	50.6	79.4	0.4	79.2	79.6
4	52.0	51.2	78.1	2.7	77.4	80.1

Tableau 7.V – Exemple 2 : routage basé sur des poids et la fonction objectif F_1.

Nous essayons une autre fonction objectif qui est de minimiser l'écart maximal entre les SL et leurs seuils l_k :

$$F_2 = \max_{k=1,\ldots,K}(l_k - g_k(\mathbf{y}^*)).$$

Contrairement à F_1, la fonction F_2 tient compte des SL qui sont au-dessus des seuils l_k. Donc, la fonction F_2 s'applique en tout temps. Par contre, un inconvénient est que F_2 ne

considère que le type d'appel ayant l'écart maximal et ne peut pas différencier les solutions selon les SL des autres types d'appels. Un centre d'appels pourrait avoir un type k auquel il est très difficile de satisfaire le seuil minimal l_k. La fonction F_2 ne serait pas une bonne fonction objectif dans ce cas particulier.

Les tableaux 7.VI et 7.VII présentent les résultats des exemples 1 et 2 avec le routage BP et la fonction objectif F_2. Comme prévu, les SL sont légèrement mieux balancés avec la fonction F_2 que F_1 lorsque les SL sont proches ou supérieurs aux seuils l_k. Autrement, il y a peu de différences entre F_1 et F_2.

Test	f_D ciblée	f_D	SL	ΔSL	min SL	max SL
1	45.0	44.4	79.9	0.2	79.8	80.0
2	46.0	45.5	79.7	0.4	79.5	79.9
3	46.0	45.3	80.3	0.2	80.1	80.3
4	36.0	33.9	82.3	0.5	82.1	82.6
5	36.0	34.2	82.0	0.1	82.0	82.1

Tableau 7.VI – Exemple 1 : routage basé sur des poids et la fonction objectif F_2.

Test	f_D ciblée	f_D	SL	ΔSL	min SL	max SL
1	52.0	50.5	79.7	0.6	79.4	80.0
2	52.0	50.7	79.2	4.1	78.5	82.6
3	52.0	50.8	79.4	1.0	78.9	79.9
4	52.0	51.1	78.8	5.4	77.9	83.3

Tableau 7.VII – Exemple 2 : routage basé sur des poids et la fonction objectif F_2.

7.4 Conclusion

Nous avons étudié une méthode de planification des agents pour une période, basée sur l'agrégation, la théorie des files d'attente et la probabilité de délai. L'agrégation se base sur les observations de Wallace et Whitt [132]. En supposant des taux de service identiques pour tous les types et groupes, ils observent qu'un centre d'appels, où tous les agents ont 2

ou 3 habiletés choisies judicieusement, peut être aussi efficace qu'un centre dont les agents possèdent toutes les habiletés. Nous disons qu'un tel centre d'appels est "efficace".

Nous supposons que les processus stochastiques du centre d'appels sont markoviens. Les types d'appels et les groupes d'agents sont premièrement agrégés pour obtenir un système de file d'attente classique avec un type d'appel et un groupe d'agents. Nous traduisons les contraintes sur les SL en contraintes sur les probabilités de délai. À cause de l'agrégation, les contraintes doivent être identiques pour tous les types d'appels. Nous simulons ce centre d'appels simplifié pour générer divers scénarios de volumes d'appels, et nous désagrégeons les appels en fonctions des taux d'arrivée originaux. Cette procédure d'agrégation puis de désagrégation est une source potentielle d'erreurs dans notre méthode, en particulier si le centre d'appels n'est pas efficace (si tous les agents ont une seule habileté par exemple).

Similairement aux méthodes d'optimisation basées sur les modèles fluides, la politique de routage est optimisée implicitement par le modèle. Cependant, la solution de routage n'est pas nécessairement réaliste ou implémentable en pratique. Il faut alors déterminer une politique de routage qui approxime le routage optimal. À la différence des méthodes basées sur les modèles fluides (où les variances stochastiques sont négligeables), notre approche vise plutôt des centres d'appels dont les volumes d'appels présentent des variances stochastiques significatives.

Les exemples numériques ont montré que notre méthode d'optimisation peut être utile. Les résultats étaient plus précis pour les centres d'appels bien balancés et pour les mesures de performance agrégées. Il est cependant plus difficile de déterminer la bonne politique de routage pour contrôler les mesures de performance par type d'appel. Nous avons obtenu de bons résultats, en optimisant le routage séparément, avec la politique de routage basé sur des poids, étudiée dans le chapitre 6.

CHAPITRE 8

MODÈLE DE CMTC BASÉ SUR LE CLIENT À LA TÊTE DE LA FILE D'ATTENTE

Les modèles d'approximation analytiques ont leurs attraits pour évaluer précisément (en général) les systèmes de files d'attente, malgré qu'ils ne soient applicables en pratique que pour des systèmes à petites dimensions. Ils permettent en particulier d'optimiser les centres d'appels par la programmation dynamique, dont l'optimisation du routage. La décision optimale de routage dépendra généralement de l'état courant du centre d'appels.

Les modèles d'approximation traditionnels se basent souvent sur une chaîne de Markov en temps continu (CMTC), où l'état de la chaîne représente le nombre d'appels ou le nombre d'agents (occupés et libres) dans le système. L'information disponible dans la CMTC n'est pas tout à fait suffisante pour approximer un centre d'appels avec plusieurs types et groupes d'agents. Les mesures de performance et la politique de routage d'un centre d'appels sont souvent basées sur les temps d'attente. Par exemple, le routage peut dépendre du temps de délai depuis l'arrivée d'un client. Le niveau de service (SL), qui est une mesure très répandue dans l'industrie, n'est pas évident à calculer à partir uniquement des longueurs des files d'attente (à l'exception du cas simple avec un type d'appel et un groupe d'agents). Le SL est défini selon la fonction de répartition du temps d'attente d'un

client, alors que la longueur de la file permet principalement d'estimer le temps d'attente moyen.

L'ajout des temps d'attente aux états de la CMTC permettrait d'approximer un plus grand nombre de modèles de centres d'appels, de politiques de routage et de mesures de performance. Motivés par cette idée, Koole et al. [90] proposent de remplacer le compteur du nombre d'appels dans la file d'attente de la CMTC traditionnelle par un compteur du temps d'attente du client à la tête de la file. Ceci permet en outre d'approximer des politiques de routage avec des temps de délai ; ce qui n'est pas évident à faire avec le modèle de CMTC traditionnel. Ils implémentent et testent leur modèle d'approximation pour un système $M/M/n$ (un type d'appel et un groupe de n agents) et un centre d'appels de modèle N (2 types et 2 groupes, voir la figure 2.1 à la page 16). Ils utilisent cette nouvelle CMTC pour optimiser le routage d'un centre d'appels de modèle N à l'aide de la programmation dynamique dans Koole et al. [89]. La politique de routage optimale dépend du temps d'attente de l'appel à la tête de chaque file et des nombres d'agents libres ou occupés.

Toutefois, leur modélisation de la CMTC impose quelques restrictions. Ils supposent aucun abandon. Les clients du même type sont servis selon la règle du *premier arrivé, premier servi*. Puisqu'ils utilisent la même variable pour décrire le nombre d'agents libres et le compteur de temps d'attente, chaque file d'attente doit alors être associée à un groupe d'agents. Leur modèle suppose qu'un groupe ne peut pas avoir d'agents libres tant que la file d'attente associée à ce groupe n'est pas vide.

Dans ce court chapitre, nous proposons et implémentons une variante simple du modèle d'approximation de Koole et al. [90] pour une file d'attente $M/M/n$. Nous utilisons, autant que possible, les mêmes notations que celles retrouvées dans leur article. Nous résumons le modèle d'approximation de ces auteurs à la section 8.1. Nous présentons nos modifications au modèle à la section 8.2. Puis, nous comparons notre variante avec le modèle de Koole et al. [90] pour le système $M/M/n$ avec des exemples numériques à la section 8.3. Les résultats numériques montrent que notre variante donne une approximation plus précise de la fonction de répartition du temps d'attente.

Remarquons qu'il n'y a aucun avantage réel à approximer un système $M/M/n$ avec une CMTC basée sur le temps d'attente du client à la tête de la file d'attente. Néanmoins,

cet exercice sert à présenter le modèle d'approximation de base et présente une valeur académique. Implémenter un modèle avec plusieurs types d'appels, comme le modèle N, ajouterait une valeur pratique, mais nous n'avons pas eu le temps de faire cela. L'implémentation de notre variante pour un centre d'appels plus grand constitue un futur projet de recherche.

8.1 Modèle d'approximation basé sur le client à la tête de la file

Nous résumons le modèle d'approximation de Koole et al. [90] pour un modèle de file d'attente markovien $M/M/n$, où il y a un type de client, un groupe de n serveurs homogènes et une file d'attente de capacité infinie. Nous employons les termes *client* et *serveur*, au lieu d'*appel* et *agent*, car le modèle est valide pour les systèmes de file d'attente en général. Il n'y a aucun abandon. Le modèle suppose qu'il n'y a aucun serveur libre tant qu'il y a des clients en attente, et les clients sont servis par ordre d'arrivée ou *premier arrivé, premier servi*.

La chaîne de Markov est définie par l'ensemble d'états $\{-n, -n+1, \ldots, 0, 1, 2, \ldots\}$. Un état $i \leq 0$ indique qu'il y a $-i$ serveurs libres et une file d'attente vide. Un état $i > 0$ représente un compteur du temps d'attente du client à la tête de la file et implique que tous les n serveurs sont occupés. (Notons que dans la CMTC traditionnelle, les états $i \geq 0$ représentent le nombre de clients en attente.) Par exemple, l'état $-n$ indique que tous les n serveurs sont libres et que la file d'attente est vide (donc, il n'y a aucun client dans le système). L'état 0 correspond à la situation où tous les n serveurs sont occupés et la file est vide.

Afin de garder un modèle markovien, le temps d'attente du client à la tête de la file est estimé à l'aide d'un compteur, qui incrémente selon une loi exponentielle de moyenne $1/\gamma$. Les auteurs choisissent de ne pas modéliser les transitions d'arrivée dans la CMTC dès qu'il y a un client en attente. Donc, la transition de l'état $i > 0$ à l'état $i+1$ survient à un taux γ. L'état $i > 0$ signifie que le client à la tête de la file a attendu jusqu'à présent une durée égale à la somme de i variables exponentielles indépendantes de taux γ. Ces i variables de temps sont indépendantes, car nous supposons une CMTC (sans mémoire). Remarquons

que la somme de i variables exponentielles indépendantes de taux γ est équivalente à une variable de loi d'Erlang avec les paramètres de forme i et d'échelle $1/\gamma$. Tous les clients en attente ont un compteur de temps d'attente, mais l'état de la CMTC contient seulement le compteur du client à la tête de la file. Nous supposons que les compteurs de tous les clients en attente incrémentent au même instant que celui du client à la tête. Lorsque le client à la tête de la file se faire répondre et quitte la file, la valeur du compteur du prochain client à la tête de la file est déterminée par une loi géométrique. Nous présentons cette loi géométrique un peu plus loin.

L'approximation de la fonction de répartition du temps d'attente devient plus précise et raffinée en choisissant un grand γ, mais le temps de calcul augmente également. Le taux d'arrivée est λ et le taux de service est μ. Notons que le taux d'incrément (ou de naissance) de l'état $i > 0$ est γ, mais le taux d'incrément à l'état 0 est λ, comme la file d'attente est vide.

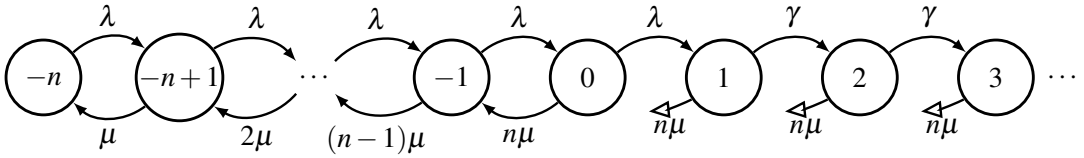


Figure 8.1 – Diagramme de transitions de la chaîne de Markov pour une file $M/M/n$.

La figure 8.1, inspirée de Koole et al. [90], présente le diagramme de transitions d'un système $M/M/n$. Remarquons que les transitions de service (ou de mort) sortant des états $i > 0$ sont différenciées par des flèches à tête blanche. La somme des taux de transition de service d'un état $i > 0$ est égale à $n\mu$, mais comme cet état correspond à un temps d'attente, le prochain état peut être n'importe quel état entre 0 et i, inclusivement. Pour chaque état $i > 0$, il y a donc i transitions de mort (qui décrémentent l'état). La différence des compteurs d'attente entre deux clients, qui entrent successivement dans la file, suit une loi géométrique de paramètre $\lambda/(\lambda+\gamma)$. Le modèle suppose que deux clients peuvent arriver à des moments très proches, tels que l'état du système ne change pas quand le client arrivé le plus tôt se fait répondre et sort de la file. Soit $p_{i,j}$ la probabilité qu'une transition

de service sortant de l'état i entre directement à l'état j, nous avons :

$$p_{i,j} = \begin{cases} 1 - \sum_{h=0}^{i-1} \left(\frac{\lambda}{\lambda+\gamma}\right)\left(\frac{\gamma}{\lambda+\gamma}\right)^h & \text{pour } i > 0,\ j = 0, \\ \left(\frac{\lambda}{\lambda+\gamma}\right)\left(\frac{\gamma}{\lambda+\gamma}\right)^{i-j} & \text{pour } i > 0,\ j \in \{1,\ldots,i\}, \\ 1 & \text{pour } -n < i \leq 0,\ j = i-1, \\ 0 & \text{autrement.} \end{cases} \tag{8.1}$$

Remarquons qu'il semble étrange d'utiliser le terme $\lambda + \gamma$, alors qu'il n'apparaît pas dans les taux de transition de la CMTC. Dans notre variante, qui sera présentée à la prochaine section, nous proposerons des définitions différentes pour les taux de transition et les probabilités $p_{i,j}$.

Pour calculer le vecteur de probabilités stationnaires $\boldsymbol{\pi}$, la chaîne de Markov est tronquée à l'ensemble d'états $\{-n, -n+1, \ldots, D\}$. Puis nous résolvons le système d'équations linéaires $\boldsymbol{\pi}\mathbf{G} = \mathbf{0}$ et $\sum_i \pi_i = 1,\ \pi_i \geq 0$, où $\mathbf{G}$ est la matrice génératrice de taille $(n+D+1) \times (n+D+1)$ composée des taux de transition $g_{i,j}$ suivants :

$$g_{i,j} = \begin{cases} \lambda & \text{pour } -n \leq i \leq 0,\ j = i+1, \\ \gamma & \text{pour } 0 < i < D,\ j = i+1, \\ (n+i)\mu & \text{pour } -n < i \leq 0,\ j = i-1, \\ p_{i,j} n\mu & \text{pour } 0 < i \leq D,\ 0 \leq j \leq i. \end{cases} \tag{8.2}$$

Dans la matrice $\mathbf{G}$, les éléments sur la diagonale sont remplacés par $g_{i,i} = -\sum_{j=-n, j\neq i}^{D} g_{i,j}$, ce qui correspond à la somme des éléments de la rangée i, excluant l'élément $g_{i,i}$, multipliée par -1. Les paramètres pour contrôler la qualité du modèle d'approximation sont $\gamma > 0$ et $D \geq 0$. Choisir des grands γ et D donnera une meilleure approximation, mais augmentera aussi le temps de calcul. Il faut sélectionner judicieusement γ et D, car l'estimation peut être erronée si la capacité D est trop petite par rapport au raffinement procuré par γ. Remarquons que D représente la *capacité du compteur de temps d'attente*, mais la supposition d'une capacité illimité de la file d'attente demeure inchangée.

À présent, pour estimer le niveau de service, il faut déterminer la fonction de répartition du temps d'attente d'un client. Pour faire cette analyse, nous considérons les transitions qui engendrent le début de service d'un client. Remarquons que tout client doit être servi (et activer une transition de service), comme le modèle suppose aucun abandon. Lorsqu'il y a des serveurs libres (les états $i < 0$), tout nouveau client est répondu immédiatement. Ceci correspond aux transitions de taux λ et le temps d'attente est zéro. Si tous les serveurs sont occupés et la file est non vide (les états $i > 0$), alors les services ne sont initiés qu'aux transitions de service (à la fin de service d'un client). Notons qu'aucun service n'est initié à l'état 0, car la file d'attente est vide et tous les serveurs sont occupés.

Supposons qu'un service soit initié, la probabilité que la chaîne de Markov était à l'état i juste avant le début du service est donnée par :

$$\alpha_i = \frac{\pi_i \Lambda_i}{\sum_{j=-n}^{D} \pi_j \Lambda_j}, \tag{8.3}$$

où π_i est la probabilité stationnaire de l'état i et Λ_i est la somme des taux de transition sortant de i qui correspondent à un début de service. Nous avons :

$$\Lambda_i = \begin{cases} \lambda & \text{pour } -n \leq i < 0, \\ 0 & \text{pour } i = 0, \\ n\mu & \text{pour } 0 < i \leq D. \end{cases} \tag{8.4}$$

Lorsque le client à la tête de la file se fait répondre à l'état $i < 0$, son temps d'attente est 0. Quand le client à la tête de la file se fait répondre à l'état $i > 0$, son temps d'attente est une somme de i variables exponentielles indépendantes de moyenne $1/\gamma$ chacune. Son temps d'attente suit alors une loi d'Erlang avec les paramètres de forme i et d'échelle $1/\gamma$, dont la fonction de répartition est $F(t;i,\gamma) = 1 - e^{-\gamma t} \sum_{j=0}^{i-1} \frac{(\gamma t)^j}{j!}$. Finalement, la fonction du niveau de service d'un client, c'est-à-dire la probabilité que le temps d'attente W soit inférieur à t, est approximativement :

$$\mathbb{P}[W \leq t] \approx \sum_{i=-n}^{-1} \alpha_i + \sum_{i=1}^{D} F(t;i,\gamma)\alpha_i. \tag{8.5}$$

Les auteurs résolvent analytiquement le modèle d'approximation du système particulier $M/M/1$ (avec un seul serveur). La fonction de répartition du temps d'attente est :

$$\mathbb{P}[W \leq t] = 1 - \frac{\lambda(\lambda+\gamma)}{\lambda^2+\mu\gamma} e^{\frac{\gamma}{\mu+\gamma}(\lambda-\mu)t}.$$

Cette équation converge vers la formule du niveau de service d'Erlang C pour une file d'attente $M/M/1$ quand $\gamma \to \infty$ et $D \to \infty$ (une CMTC non tronquée).

8.2 Modifications proposées au modèle d'approximation

Dans cette section, nous proposons quelques modifications simples au modèle de Koole et al. [90]. Remarquons que les transitions de taux γ pour compter le temps d'attente ressemblent aux transitions fictives dans la technique d'uniformisation d'une CMTC, présentée à la section 4.1.3, page 72. Plus précisément, une transition fictive correspondrait à une transition de temps d'attente où il n'y a eu aucune arrivée d'appel.

La première modification consiste à ajouter des transitions d'arrivée de taux λ aux états $i \geq 0$. Dans le modèle original, la transition de taux γ correspond à un accroissement du temps d'attente, qui est indépendant des arrivées des clients. Dans notre variante du modèle, nous divisons la transition de taux γ en deux types de transition mutuellement exclusifs, afin d'expliciter si l'arrivée d'un client a eu lieu ou non. Nous définissons $\gamma = \lambda + \gamma_0$ avec $\gamma_0 \geq 0$. Ces deux types de transition sont :

1. Une transition de taux λ qui correspond à l'événement d'une **augmentation du temps d'attente** du client à la tête de la file et **à l'arrivée d'un client**.
2. Une transition de taux γ_0 qui correspond à l'événement d'une **augmentation du temps d'attente** du client à la tête de la file, mais **sans arrivée d'un client**.

La figure 8.2 présente le nouveau diagramme de transitions, où les transitions de taux γ de la figure 8.1 sont remplacées par les transitions de taux λ et γ_0. En somme, le passage d'un état $i > 0$ à l'état $(i+1)$ indique une augmentation du temps d'attente du client à la tête de la file avec la possibilité d'arrivée d'**au plus un** nouveau client. Notons que les transitions de service sont identiques à ceux de la figure 8.1.

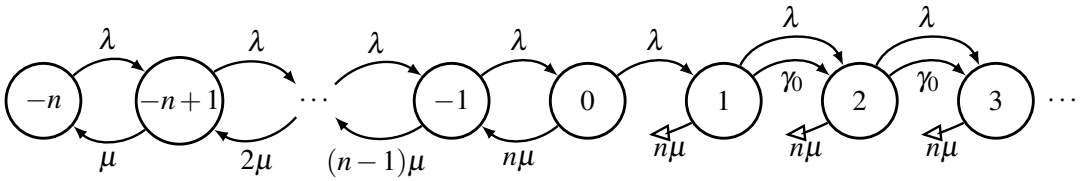


Figure 8.2 – Nouveau diagramme de transitions de la chaîne de Markov que nous proposons pour un système $M/M/n$.

Cette séparation de la transition de taux γ implique que la chaîne de Markov ne peut pas rester dans le même état quand un client se fait répondre. En particulier si la chaîne est à l'état $i > 1$ juste avant le début de service du client, alors l'état suivant j doit être $0 \leq j < i$; **ce qui diffère aussi du modèle original**. Les équations (8.1) et (8.2) deviennent :

$$p_{i,j} = \begin{cases} 1 - \sum_{h=0}^{i-2} \left(\frac{\lambda}{\lambda+\gamma_0}\right)\left(\frac{\gamma_0}{\lambda+\gamma_0}\right)^h & \text{pour } i > 0,\ j = 0, \\ \left(\frac{\lambda}{\lambda+\gamma_0}\right)\left(\frac{\gamma_0}{\lambda+\gamma_0}\right)^{i-j-1} & \text{pour } i > 1,\ j \in \{1,\ldots,i-1\}, \\ 1 & \text{pour } -n < i \leq 0,\ j = i-1, \\ 0 & \text{autrement.} \end{cases} \tag{8.6}$$

$$g_{i,j} = \begin{cases} \lambda & \text{pour } -n \leq i \leq 0,\ j = i+1, \\ \lambda+\gamma_0 & \text{pour } 0 < i < D,\ j = i+1, \\ (n+i)\mu & \text{pour } -n < i \leq 0,\ j = i-1, \\ p_{i,j} n\mu & \text{pour } 0 < i \leq D,\ 0 \leq j < i. \end{cases} \tag{8.7}$$

Le dénominateur $\lambda + \gamma_0$ et la loi géométrique dans (8.6) sont beaucoup plus cohérents avec la CMTC de la figure 8.2 que les équations originales (8.1) et la CMTC de la figure 8.1. Le nombre de transitions entre le compteur de temps d'attente du client à la tête de la file et celui du second client dans la file suit une loi géométrique de paramètre $\lambda/(\lambda+\gamma_0)$.

Pour garder la même somme totale des taux de transition que le modèle original, nous pouvons choisir $\gamma_0 \geq 0$ tel que $\gamma = \lambda + \gamma_0$. Les équations (8.3) – (8.5) demeurent identiques dans notre variante. En général, il faut choisir D suffisamment grand (en particulier si γ

est grand), sinon la troncation de la capacité du compteur de temps peut causer des erreurs d'approximation substantielles.

8.3 Exemples numériques

Nous comparons la précision de notre variante avec le modèle original pour l'approximation de la formule d'Erlang C. Les exemples sont des systèmes $M/M/n$, tirés des figures 2(a), 2(b), 3(a) et 3(b) de l'article de Koole et al. [90]. Nous référons à leur article tout au long de cette section. Notre variante utilise les mêmes paramètres D et $\gamma = \lambda + \gamma_0$ que le modèle original. Notre implémentation du modèle original semble valide, car les résultats numériques sont virtuellement identiques à ceux présentés dans leur article. L'unité de temps utilisé dans les exemples est arbitraire puisqu'elle est la même pour toutes les mesures. Dans les figures, la mesure de probabilité $\mathbb{P}_{\mathrm{O}}$ est calculée par le modèle original, et $\mathbb{P}_{\mathrm{M}}$ est calculée par notre modèle modifié.

Exemple : figure 2(a)

Nous comparons la précision des modèles pour l'approximation de la fonction de répartition du temps d'attente W d'un client, $\mathbb{P}[W \le t]$, avec la formule d'Erlang C. Les paramètres sont $\lambda = 3,\ \mu = 1,\ \gamma = \frac{2\mu}{3\gamma}(Dn)^{3/4}$ et nous varions $D = 10$, 50, 200. Ces trois valeurs de D donnent les valeurs $\gamma \approx 3.53$, 11.82 et 33.43, respectivement. Notons qu'il y a une erreur dans l'article pour la formule de γ : le paramètre n n'est pas un indice. La figure 8.3 montre que notre modèle donne une approximation plus précise de $\mathbb{P}[W \le t]$ comme fonction de t.

Exemple : figure 2(b)

Cet exemple est similaire à l'exemple précédent où nous estimons la fonction de répartition du temps d'attente d'un client. Les paramètres de l'exemple sont $\lambda = 37,\ \mu = 1,\ \gamma = \frac{3\mu n}{4\lambda}D^{0.9}$ et nous varions $D = 50$, 200, 1000. Ces trois valeurs de D donnent les valeurs $\gamma \approx 27.42$, 95.47 et 406.37, respectivement. La figure 8.4 montre également que

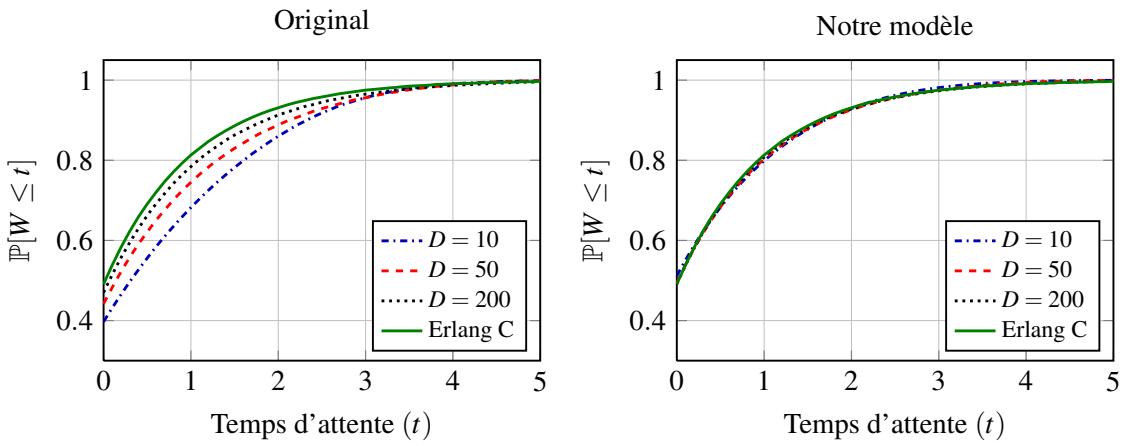


Figure 8.3 – Comparaison de la fonction de répartition du temps d'attente avec la formule d'Erlang C, basée sur l'exemple de la figure 2(a) de l'article original.

notre modèle donne une meilleure approximation de la fonction de répartition du temps d'attente.

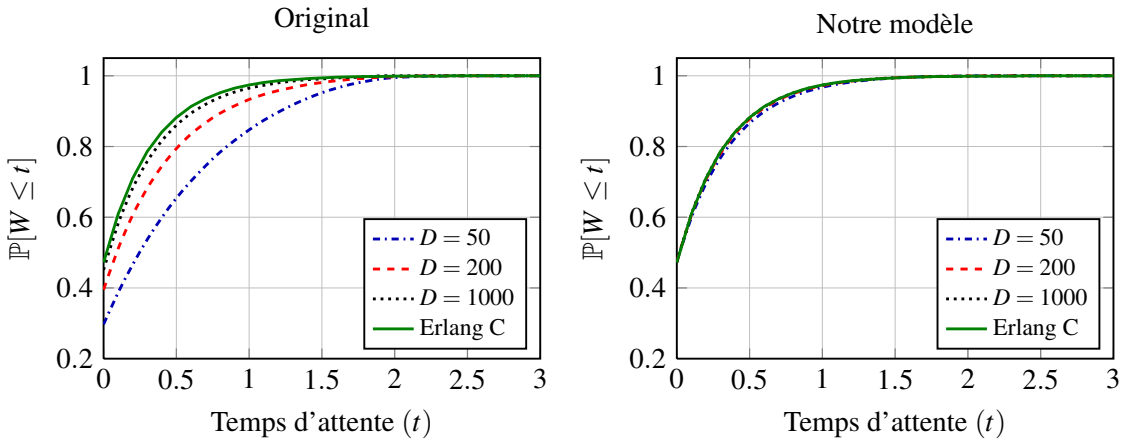


Figure 8.4 – Comparaison de la fonction de répartition du temps d'attente avec la formule d'Erlang C, basée sur l'exemple de la figure 2(b) de l'article original.

Exemple : figure 3(a)

Dans cet exemple, nous mesurons l'erreur absolue de la fonction de répartition du temps d'attente des modèles d'approximation par rapport à la formule d'Erlang C. Les

paramètres sont $\mu = 1$, $n = 4$ et $\gamma = \frac{2\mu}{3\lambda}(Dn)^{3/4}$. Nous fixons $D = 200$ et nous expérimentons les modèles sous différents niveaux d'intensité de trafic $\rho = \lambda/(n\mu) = 1/4,\ 1/2,\ 3/4$ et $7/8$, en variant λ. Ces quatre valeurs de ρ donnent les valeurs $\lambda = 1,\ 2,\ 3,\ 3.5$, et les valeurs $\gamma \approx 100.3,\ 50.1,\ 33.4,\ 28.7$, respectivement.

La figure 8.5 montre clairement que notre modèle est plus précis, à l'exception du cas $\rho = 7/8$ où l'intensité du trafic est la plus grande. L'erreur est causée par la troncation trop sévère de la chaîne de Markov lorsque l'intensité du trafic est forte. À cause du manque d'états (où le client attend longtemps), les probabilités stationnaires des petits états (moins d'attente) sont augmentées artificiellement ; ce qui explique la surestimation de la fonction du SL.

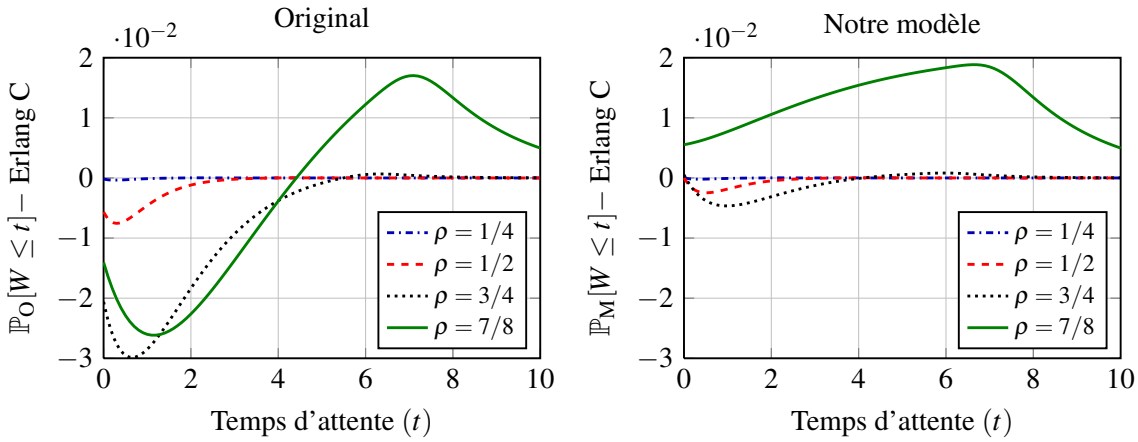


Figure 8.5 – Erreurs absolues de la fonction de répartition du temps d'attente W des modèles d'approximation par rapport à la formule d'Erlang C pour différents niveaux d'intensité de trafic ρ, pour l'exemple de la figure 3(a) de l'article original.

Dans la figure 8.6, nous augmentons la capacité $D = 800$ (4 fois plus grande), et nous ajustons la formule $\gamma = \frac{2\mu}{3\lambda}(Dn/4)^{3/4}$ afin de garder les mêmes taux γ que le cas $D = 200$. La courbe $\rho = 7/8$ s'est grandement améliorée dans notre modèle, et notre modèle demeure le plus précis. Remarquons que les 3 autres courbes ($\rho = 1/4,\ 1/2$ et $3/4$) ne présentent pas de changement significatif par rapport à la figure 8.5. Avec $D = 800$, les deux modèles d'approximation ne surestiment plus la fonction du SL.

Il faudrait raffiner le compteur de temps d'attente, en augmentant sa fréquence, pour réduire les erreurs absolues des courbes $\rho = 1/4,\ 1/2$ et $3/4$. Dans la figure 8.7, nous

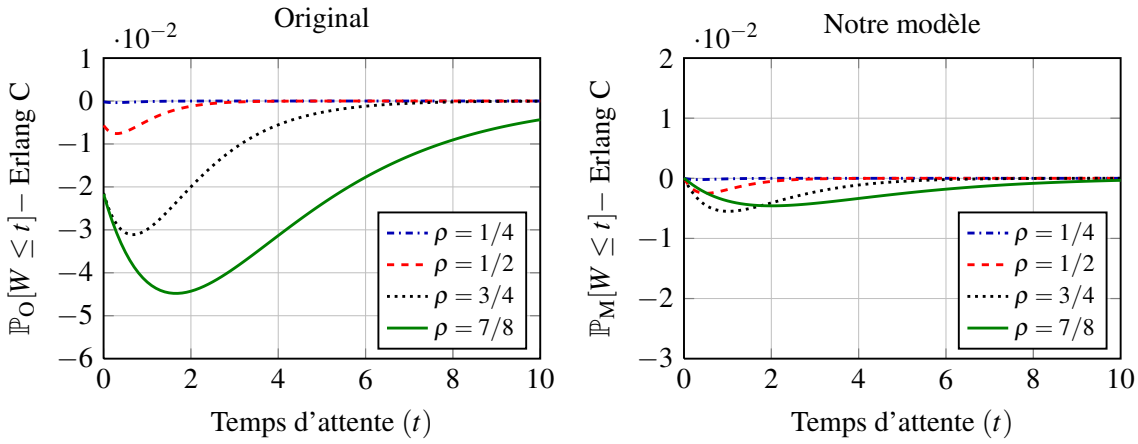


Figure 8.6 – Erreurs absolues de la fonction de répartition du temps d'attente W des modèles d'approximation par rapport à la formule d'Erlang C pour différents niveaux d'intensité de trafic ρ, pour l'exemple de la figure 3(a) avec $D = 800$. Notez que l'échelle de l'axe vertical du modèle original a changé.

gardons $D = 800$, mais nous doublons les valeurs de $\gamma \approx 200.6$, 100.2, 66.8 et 57.3, respectivement pour $\rho = 1/4,\ 1/2,\ 3/4$ et $7/8$. Les résultats montrent que les erreurs de toutes les courbes ont diminué, et que notre modèle demeure meilleur que l'original.

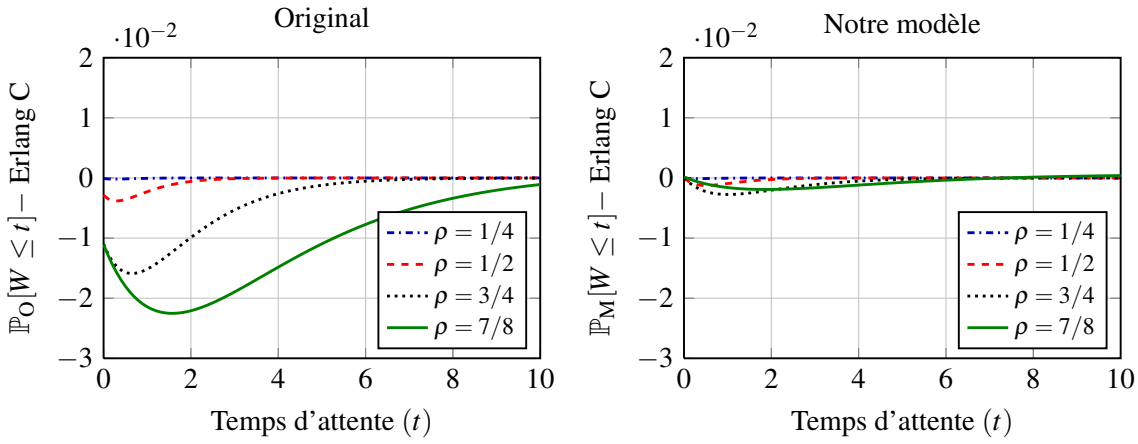


Figure 8.7 – Erreurs absolues de la fonction de répartition du temps d'attente W des modèles d'approximation par rapport à la formule d'Erlang C pour différents niveaux d'intensité de trafic ρ, pour l'exemple de la figure 3(a) avec $D = 800$ et γ deux fois plus grand.

Exemple : figure 3(b)

Cet exemple est identique à l'exemple de la figure 3(a), sauf que $D = 1000$ et γ augmente par un facteur de $5^{3/4} \approx 3.344$. Nous mesurons l'erreur absolue de la fonction de répartition du temps d'attente entre les modèles d'approximation et la formule d'Erlang C. Les autres paramètres de l'exemple sont $\mu = 1$, $n = 4$, $\gamma = \frac{2\mu}{3\lambda}(Dn)^{3/4}$ et nous testons pour $\rho = \lambda/(n\mu) = 1/4,\ 1/2,\ 3/4$ et $7/8$. Ces quatre valeurs de ρ donnent les valeurs $\lambda = 1$, 2, 3, 3.5, et les valeurs $\gamma \approx 335.3$, 167.7, 111.8, 95.8, respectivement.

La figure 8.8 montre que notre modèle est meilleur. Par contre, notre modèle surestime la fonction du niveau de service dans le cas de $\rho = 7/8$. Même si D augmente, la précision du modèle d'approximation peut diminuer si γ augmente trop vite. Il faudrait plus d'états dans la chaîne de Markov, car le compteur de temps d'attente augmente plus rapidement quand γ est grand.

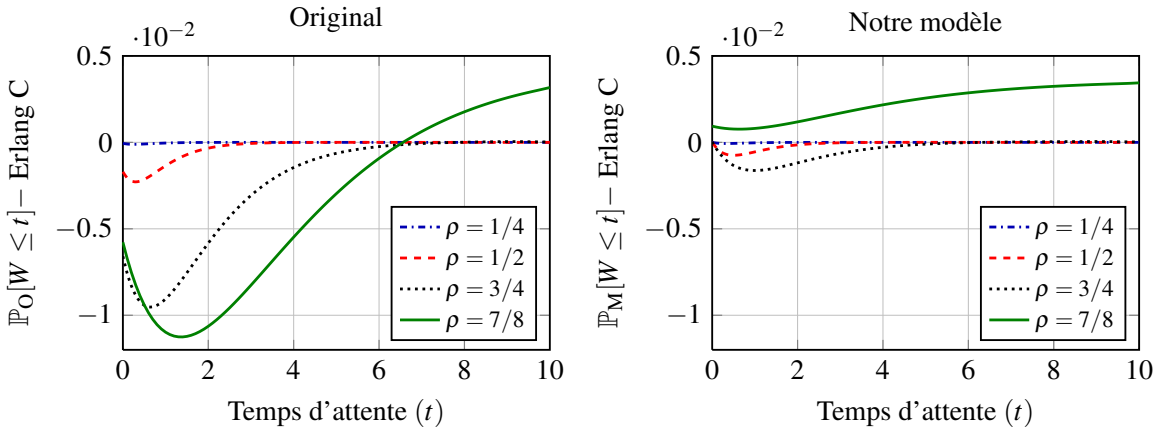


Figure 8.8 – Erreurs absolues de la fonction de répartition du temps d'attente W des modèles d'approximation par rapport à la formule d'Erlang C pour différents niveaux d'intensité de trafic ρ, pour l'exemple de la figure 3(b) dans l'article.

Comme pour l'exemple de la figure 3(a) et la figure 8.6, nous augmentons la capacité $D = 2000$ (2 fois plus grande), mais nous gardons γ inchangé en multipliant par un facteur de $2^{-3/4} \approx 0.5946$. La figure 8.9 montre que notre modèle est nettement meilleur, et les deux modèles d'approximation ne surestiment plus la fonction du SL.

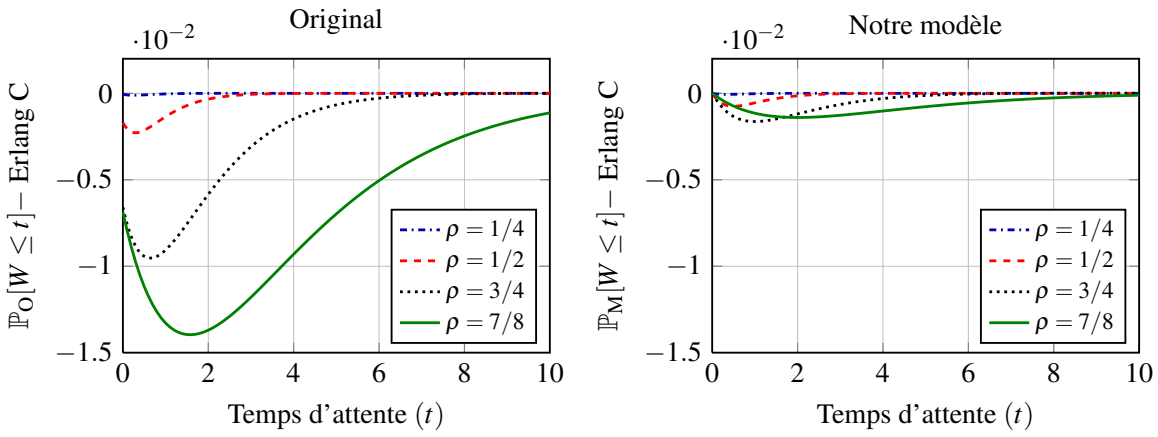


Figure 8.9 – Erreurs absolues de la fonction de répartition du temps d'attente W des modèles d'approximation par rapport à la formule d'Erlang C pour différents niveaux d'intensité de trafic ρ, pour l'exemple de la figure 3(b) avec $D = 2000$.

8.4 Conclusion

Dans ce chapitre, nous avons étudié le modèle de Koole et al. [90] pour l'approximation d'une CMTC, sans abandon, en se basant sur le temps d'attente du client à la tête de la file. Ce modèle remplace l'information sur la longueur de la file d'attente du modèle traditionnel de CMTC par un compteur du temps d'attente du client à la tête de la file. Ce compteur n'incrémente pas à des intervalles de temps fixes, mais en suivant un processus de Poisson. Ce modèle demeure donc markovien. L'intérêt de ce modèle est qu'il permet d'approximer une CMTC avec une politique de routage basée sur les temps d'attente, les nombres d'agents libres ou occupés, et les priorités. Nous avons présenté leur modèle pour approximer un système de base, la file d'attente $M/M/n$.

Nous avons proposé des modifications simples à leur modèle dans la section 8.2. Notre variante diffère essentiellement du modèle de Koole et al. [90] par le processus de comptage du temps d'attente qui inclut explicitement l'événement d'arrivée d'un client. Nous distinguons deux types d'événement qui incrémentent le compteur de temps d'attente. Soit le compteur de temps d'attente incrémente avec l'arrivée d'un nouveau client, ou le compteur incrémente sans arrivée d'un nouveau client. Le calcul des probabilités que nous proposons est aussi plus cohérent avec le modèle de CMTC à approximer. La précision du modèle de

CMTC dépend du taux d'incrément du compteur de temps d'attente (un processus de Poisson). Plus ce taux est élevé, plus le temps d'attente estimé sera raffiné et précis. Mais la CMTC nécessitera plus d'états, ce qui ralentit les calculs.

Dans les résultats numériques de la section 8.3, nous avons comparé la précision de notre variante et le modèle de Koole et al. [90] pour estimer la fonction de répartition du temps d'attente (la fonction du SL) de divers files $M/M/n$ avec de petites à hautes intensités de trafic. Les résultats ont montré que notre variante est significativement plus précise que le modèle original dans tous les cas.

Dans ce chapitre, nous avons présenté l'idée de base de l'approximation d'une CMTC basée sur le client à la tête de la file d'attente pour un système $M/M/n$. Un futur projet de recherche serait d'implémenter notre variante pour un centre d'appels avec plusieurs types d'appels et groupes d'agents.

Chapitre 9

Planification des agents avec recours

Avis important : ce chapitre présente des idées sur la planification stochastique avec recours d'un centre d'appels multi-compétences à poursuivre dans un futur projet de recherche. Nous n'avons pas eu le temps d'implanter et de tester les méthodes proposées.

Dans ce chapitre, nous présentons un algorithme d'optimisation stochastique avec possibilité de recours pour le problème d'affectation des agents. Dans la réalité, les taux d'arrivée exacts des appels sont inconnus, et ils sont estimés par des modèles de prévision. La possibilité de recours permet à un gestionnaire d'ajouter des agents (si les taux d'arrivée observés sont plus forts que prévu) ou de retirer des agents afin de réduire les dépenses (si les taux observés sont plus faibles que prévu).

Nous retrouvons plusieurs études sur le problème de planification stochastique des agents dans la littérature [20, 59, 75, 100, 113, 117, 136], voir les résumés à la section 3.3.3, page 51. La majorité de ces études étudie un centre d'appels avec un seul type d'appel et un groupe d'agents. La majorité considère seulement la variabilité stochastique des arrivées d'appels, comme nous aussi dans ce chapitre. Les quelques études sur la planification stochastique pour un centre avec plusieurs types d'appels et groupes d'agents se

basent sur des régimes à trafic intense et les modèles fluides. Certaines études optimisent également le routage des appels, puis certaines incluent des actions de recours dans leur problématique.

Nous proposons ici une méthode pour optimiser le problème d'affectation d'un centre d'appels multi-compétences avec des taux d'arrivée stochastiques et des options de recours sur les agents, à l'aide de la simulation et de la programmation linéaire. Pour simplifier le problème, nous nous concentrerons sur un centre d'appels ayant une seule période représentant une journée entière. Il est cependant possible d'étendre notre algorithme pour optimiser le problème de planification des quarts de travail vu au chapitre 5. Notre algorithme est une extension de la méthode de coupes linéaires et de simulation de Ceẓik et L'Ecuyer [37]. Nous modélisons explicitement les arrivées des appels par des processus de Poisson avec des taux stochastiques dans la problématique, et nous ajoutons des actions de recours sur les agents. Notre approche permet aussi de considérer des contraintes probabilistes telles que la contrainte de satisfaire un seuil minimal de SL avec une certaine probabilité. Notons qu'il n'y a pas de résultat numérique dans ce chapitre par manque de temps, et ces idées constituent un futur projet de recherche.

9.1 Problème de planification traditionnel sans recours

Nous commençons par décrire le problème de planification traditionnel de Ceẓik et L'Ecuyer [37], présenté à la section 3.3.2, page 42. Considérons un centre d'appels avec K types d'appels et I groupes d'agents. Soient $c_i > 0$ le coût d'un agent du groupe i, y_i le nombre d'agents du groupe i, $g_k(\mathbf{y})$ et $g(\mathbf{y})$ les SL du type d'appel k et agrégé sur tous les appels avec les seuils minimaux l_k et l à satisfaire. Ainsi, un gestionnaire d'un centre d'appels voudrait trouver $\mathbf{y} = (y_1, \ldots, y_I)^{\mathrm{T}}$ qui optimise le problème suivant :

$$\begin{aligned} \min \quad & \textstyle\sum_{i=1}^{I} c_i y_i \\ \text{sujet à :} \quad & g(\mathbf{y}) \geq l, \\ & g_k(\mathbf{y}) \geq l_k, \qquad k = 1, \ldots, K, \\ & \mathbf{y} \geq 0 \text{ et entier.} \end{aligned} \tag{P0}$$

Des algorithmes d'optimisation [17, 18, 37, 109] ont été proposés pour résoudre le problème (P0), mais ils n'optimisent en considérant que les taux d'arrivée moyens et sans option de recours.

Cependant, le problème P0 n'est pas très réaliste en pratique. Premièrement, l'affectation des agents doit se faire à l'avance, car un gestionnaire ne peut pas trouver $\mathbf{y}$ agents instantanément ou juste avant l'ouverture de la journée. Le problème d'affectation se fait parfois quelques semaines ou des mois à l'avance. (La planification des quarts se fait souvent quelques jours à l'avance.) Les agents peuvent aussi nécessiter une période de formation. La majorité des méthodes d'affectation et de planification des centres d'appels supposent que les taux d'arrivée des appels sont connus et sans erreur. Ceci qui est loin d'être le cas dans la réalité. La prévision des volumes d'appels est un problème difficile en soi, et elle est souvent assujettie à des erreurs, en particulier pour les prévisions à long terme. En général, les prévisions sont plus précises lorsque l'horizon de temps est court (prévisions pour l'après-midi ou le lendemain par exemple).

Dans la réalité, un gestionnaire utilise des moyens de recours afin d'adapter l'effectif d'agents au volume d'appels observés. Si le volume d'appels est plus élevé que prévu, alors le gestionnaire appelle des agents en renfort en annulant des tâches ou des périodes de formation, ou en payant des heures supplémentaires. À l'opposé, si le volume est plus faible que prévu, alors le gestionnaire peut réduire le nombre d'agents en offrant des congés sans solde ou en affectant des agents à des tâches de bureau ou à des séances de formation.

9.2 Problème de planification avec recours à l'avance

Nous considérons une version simple du problème avec recours pour un centre composé de K types d'appels et I groupes d'agents. Supposons qu'au jour 0, le gestionnaire doit déterminer l'effectif d'agents $\mathbf{y}$ nécessaire pour la journée J, qui arrive dans J jours. Le gestionnaire doit alors optimiser $\mathbf{y}$ en fonction des prévisions des volumes d'appels à long terme disponibles au jour 0. Soit $\Lambda = (\lambda_1, \ldots, \lambda_k)$ le vecteur aléatoire représentant les taux d'arrivée des K types d'appels de la journée J. Supposons que les prévisions soient données sous la forme d'une loi de probabilité discrète multivariée définie sur une portée

finie de M réalisations possibles, disons $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_M$, où $\boldsymbol{\lambda}_m = (\lambda_{1,m}, \lambda_{2,m}, \ldots, \lambda_{K,m})$, avec les masses de probabilité correspondantes $p_1, p_2, \ldots, p_M \geq 0$ et $\sum_{m=1}^{M} p_m = 1$. Si M est grand, nous approximons la distribution de Λ par une loi empirique en générant aléatoirement $M_0 < M$ échantillons. Dans le cas d'une portée infinie ou d'une loi de probabilité continue, nous pouvons approximer la distribution de Λ de manière similaire par une loi de probabilité empirique basée sur M échantillons.

Dans le jargon de la programmation stochastique [24], chacune des M réalisations représente un scénario, où le scénario $\boldsymbol{\lambda}_m$ a une probabilité p_m de se réaliser. Remarquons que notre problématique ne se restreint pas obligatoirement qu'aux taux d'arrivée aléatoires. Les paramètres de distribution des temps de service ou de patience pourraient aussi être différents pour chaque scénario. Pour simplifier la présentation, nous supposons pour le reste du chapitre que seulement les taux d'arrivée sont stochastiques.

Supposons qu'à la veille, c'est-à-dire au jour $J-1$, le gestionnaire révise ses prévisions avec les dernières tendances de la semaine et il obtient des prévisions beaucoup plus précises. Dans cette version simplifiée du problème, nous supposons que les révisions donnent les taux d'arrivée **exacts** (une hypothèse importante). Le gestionnaire peut alors corriger la planification en fonction des révisions exactes, et aucune action de recours ne sera nécessaire au jour J. Advenant le scénario m, les actions de recours possibles sont d'ajouter $r^+_{i,m}$ agents ou de retirer $r^-_{i,m}$ agents du groupe i. Le nombre d'agents affectés au groupe i au jour J sera $y_i + r^+_{i,m} - r^-_{i,m}$.

Nous pénalisons les actions de recours : (1) retirer un agent du groupe i diminue le coût de $q_i^- < c_i$ et (2) ajouter un agent au groupe i augmente le coût de $q_i^+ > c_i$. Étant donné que $(q_i^+ - q_i^-) > 0$, alors la solution doit forcément avoir $r^+_{i,m} = 0$ ou $r^-_{i,m} = 0$ (ou les deux), car il n'y a aucun intérêt d'ajouter des agents, puis de les retirer. Notons qu'au lieu des inégalités strictes, si nous permettions que $c_i = q_i^-$ ou $c_i = q_i^+$, alors il n'y aurait aucun avantage à planifier $\mathbf{y}$ à l'avance puisque le coût serait le même lors de la révision ! Un gestionnaire pourrait planifier beaucoup trop d'agents dans $\mathbf{y}$ au jour 0, puis retirer l'excès lors de la révision. Ou bien, il pourrait planifier aucun agent $\mathbf{y} = (0, \ldots, 0)^{\mathrm{T}}$ au jour 0, et ajouter le bon nombre d'agents lors de la révision.

Soient $g(\boldsymbol{\lambda}, \mathbf{y})$ le SL agrégé et $g_k(\boldsymbol{\lambda}, \mathbf{y})$ le SL pour le type d'appel k avec en paramètres

les taux d'arrivée $\boldsymbol{\lambda}$ et le vecteur d'agents $\mathbf{y}$. Si le gestionnaire désire satisfaire toutes les contraintes de SL, et ce, peu importe le scénario qui se produira, alors le problème avec recours est le suivant :

$$\begin{aligned} \min \quad & \sum_{i=1}^{I} c_i y_i + \sum_{m=1}^{M} \sum_{i=1}^{I} \left(q_i^+ r_{i,m}^+ - q_i^- r_{i,m}^- \right) p_m \\ \text{sujet à :} \quad & g(\boldsymbol{\lambda}_m, \mathbf{y} + \mathbf{r}_m^+ - \mathbf{r}_m^-) \geq l, && \forall m, \\ & g_k(\boldsymbol{\lambda}_m, \mathbf{y} + \mathbf{r}_m^+ - \mathbf{r}_m^-) \geq l_k, && \forall m, \forall k, \\ & r_{i,m}^+ \geq 0 \text{ et entier}, && \forall i, \forall m, \\ & 0 \leq r_{i,m}^- \leq y_i \text{ et entier}, && \forall i, \forall m, \\ & y_i \geq 0 \text{ et entier}, && \forall i, \end{aligned} \tag{P1}$$

où $\mathbf{r}_m^+ = (r_{1,m}^+, r_{2,m}^+, \ldots, r_{I,m}^+)^{\mathrm{T}}$ et $\mathbf{r}_m^- = (r_{1,m}^-, r_{2,m}^-, \ldots, r_{I,m}^-)^{\mathrm{T}}$. Notons que la vraie borne supérieure de $r_{i,m}^-$ est $r_{i,m}^- \leq (y_i + r_{i,m}^+)$, mais puisque la structure de coût empêche d'avoir simultanément $r_{1,m}^- > 0$ et $r_{1,m}^+ > 0$, nous choisissons directement la borne $r_{i,m}^- \leq y_i$. En pratique, les variables de recours $r_{i,m}^-, r_{i,m}^+$ pourraient être bornées par des contraintes supplémentaires.

En résumé, nous supposons que le gestionnaire optimise le problème (P1) au jour 0 et planifie $\mathbf{y}$ agents pour le jour J. Lorsque les prévisions deviennent plus précises (voire exactes selon notre hypothèse) à l'approche du jour J, le gestionnaire apportera les modifications données par $\mathbf{r}_m^+$ et $\mathbf{r}_m^-$ en fonction du scénario m prédit avec certitude. Remarquons que la solution à (P1) satisfait toutes les contraintes de SL pour tous les M scénarios. Plus tard, nous présenterons une extension avec une contrainte par chance (ou probabiliste) sur la satisfaction des SL du problème (P1).

9.3 Extension à l'algorithme de coupes linéaires avec simulation

Pour résoudre le problème (P1), nous proposons une extension de la méthode de Ceẓik et L'Ecuyer [37], présentée à la section 3.3.2, page 42, qui utilise la programmation linéaire et la simulation. Afin de simplifier la lecture, nous discutons de la fonction g dans cette section, mais la discussion s'applique tout aussi bien aux fonctions g_k. Cette méthode

heuristique remplace la contrainte non linéaire sur g par un ensemble de coupes linéaires ajoutées itérativement. Basée sur l'hypothèse de concavité de la fonction g (ou plutôt la courbe de g en forme d'un "S"), cette méthode suppose qu'elle ne coupe aucune solution réalisable. Elle s'arrête à la première solution réalisable au problème (P1), et cette solution sera optimale. Puisque nous ne savons pas comment calculer exactement g, il serait plus exact de dire que nous optimisons par rapport à une fonction simulée $\hat{g}$. Nous supposons que la fonction $\hat{g}$ converge vers g lorsque le nombre de réplications de simulation est suffisamment grand.

Pour initialiser l'algorithme, nous ajoutons des contraintes à l'aide des variables continues $w_{k,i,m}$ afin de couvrir une fraction $\alpha_{k,m}$ du volume d'appels de chaque type k et scénario m. Les paramètres $\alpha_{k,m} \geq 0$ sont fixés par l'utilisateur, généralement à une valeur autour de 1. L'ensemble d'habiletés du groupe i est $\mathcal{S}_i \subseteq \{1,2,\ldots,K\}$, et $\mathcal{T}_k = \{i : k \in \mathcal{S}_i, \forall i\}$ est l'ensemble des groupes pouvant servir le type d'appel k. Cette initialisation est identique à celle d'Avramidis et al. [17] présentée dans le chapitre 5. Notre algorithme est une méthode itérative, et il résout le problème suivant à l'itération v :

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{I} c_i y_i + \sum_{m=1}^{M} \sum_{i=1}^{I} \left(q_i^+ r_{i,m}^+ - q_i^- r_{i,m}^- \right) p_m \\
\text{sujet à :} \quad & \mathbf{B}_m^{(v)} (\mathbf{y} + \mathbf{r}_m^+ - \mathbf{r}_m^-) \geq \mathbf{b}_m^{(v)}, && \forall m, \\
& \sum_{i \in \mathcal{T}_k} w_{k,i,m} \mu_{k,i} \geq \alpha_{k,m} \lambda_{k,m}, && \forall k, \forall m, \\
& \sum_{k \in \mathcal{S}_i} w_{k,i,m} = y_i + r_{i,m}^+ - r_{i,m}^-, && \forall i, \forall m, \\
& w_{k,i,m} \geq 0, && \forall k, \forall i, \forall m, \\
& r_{i,m}^+ \geq 0 \text{ et entier}, && \forall i, \forall m, \\
& 0 \leq r_{i,m}^- \leq y_i \text{ et entier}, && \forall i, \forall m, \\
& y_i \geq 0 \text{ et entier}, && \forall i.
\end{aligned}
\qquad (\mathrm{P2}^{(v)})
$$

Pour chaque scénario m, nous avons un simulateur avec les paramètres $\boldsymbol{\lambda}_m$ et nous générons des coupes basées sur les sous-gradients indépendamment des autres scénarios. Les différents scénarios sont reliés uniquement par les variables $\mathbf{y}$. L'ensemble des coupes linéaires générées pour le scénario m est représentée par la paire $(\mathbf{B}_m^{(v)}, \mathbf{b}_m^{(v)})$, et toutes ces

contraintes forment une structure angulaire par bloc :

$$\begin{array}{lllll}
\mathbf{B}_1^{(v)}\mathbf{y} & +\mathbf{B}_1^{(v)}(\mathbf{r}_1^+ - \mathbf{r}_1^-) & & & \geq \mathbf{b}_1^{(v)} \\
\mathbf{B}_2^{(v)}\mathbf{y} & & +\mathbf{B}_2^{(v)}(\mathbf{r}_2^+ - \mathbf{r}_2^-) & & \geq \mathbf{b}_2^{(v)} \\
\vdots & & & \ddots & \vdots \\
\mathbf{B}_M^{(v)}\mathbf{y} & & & \cdots \quad +\mathbf{B}_M^{(v)}(\mathbf{r}_M^+ - \mathbf{r}_M^-) & \geq \mathbf{b}_M^{(v)}.
\end{array}$$

Si le nombre de scénarios est élevé, nous pouvons utiliser des méthodes de décomposition pour résoudre le problème (P2$^{(v)}$).

Notre extension optimise les M scénarios simultanément. Nous débutons avec les ensembles de contraintes initiales $(\mathbf{B}_m^{(1)}, \mathbf{b}_m^{(1)}), \forall m$ vides. À l'itération v, nous optimisons le problème linéaire (P2$^{(v)}$) et nous obtenons la solution $\mathbf{y}^{(v)}, \mathbf{r}_1^{+,(v)}, \ldots, \mathbf{r}_M^{+,(v)}, \mathbf{r}_1^{-,(v)}, \ldots, \mathbf{r}_M^{-,(v)}$. Soit $\bar{\mathbf{y}}_m = \mathbf{y}^{(v)} + \mathbf{r}_m^{+,(v)} - \mathbf{r}_m^{-,(v)}$ le nombre effectif d'agents pour le scénario m. Pour chaque scénario m, nous simulons le vecteur $\bar{\mathbf{y}}_m$ afin de vérifier la réalisabilité du scénario. Si le scénario m n'est pas réalisable, alors nous estimons le sous-gradient au point $\bar{\mathbf{y}}_m$, et nous ajoutons une coupe linéaire dans $(\mathbf{B}_m^{(v+1)}, \mathbf{b}_m^{(v+1)})$. Comme chaque scénario est simulé indépendamment des autres, le nombre maximal de simulations à exécuter par itération est $(I+1)M$, comparé à $I+1$ dans l'algorithme original. Cependant, il n'est probablement pas nécessaire de générer une coupe de sous-gradient pour tous les M scénarios à chaque itération. Nous présentons quelques raisonnements afin de réduire le nombre de simulations à la section 9.5. L'algorithme se termine lorsque la solution est réalisable pour tous les M scénarios.

Approche à deux étapes (TS)

Il pourrait être tentant d'optimiser le problème (P1) par une approche à deux étapes (TS). La méthode commence par optimiser chaque scénario m indépendamment à l'aide de l'algorithme de Ceẓik et L'Ecuyer [37] ou autres. Soit $\hat{\mathbf{y}}_m = (\hat{y}_{1,m}, \hat{y}_{2,m}, \ldots, \hat{y}_{I,m})^{\mathrm{T}}$ la solution du scénario m. La deuxième étape consiste simplement à couvrir les solutions

$\hat{\mathbf{y}}_1, \hat{\mathbf{y}}_2, \ldots, \hat{\mathbf{y}}_M$:

$$\begin{aligned} \min \quad & \sum_{i=1}^{I} c_i y_i + \sum_{m=1}^{M} \sum_{i=1}^{I} \left(q_i^+ r_{i,m}^+ - q_i^- r_{i,m}^- \right) p_m \\ \text{sujet à :} \quad & y_i + r_{i,m}^+ - r_{i,m}^- \geq \hat{y}_{i,m}, \quad \forall i, \forall m, \\ & r_{i,m}^+ \geq 0 \text{ et entier}, \quad \forall i, \forall m, \\ & 0 \leq r_{i,m}^- \leq y_i \text{ et entier}, \quad \forall i, \forall m, \\ & y_i \geq 0 \text{ et entier}, \quad \forall i. \end{aligned}$$

Cependant, la méthode TS est sous-optimale. Voici deux exemples.

Exemple 1. Soit un centre d'appels de modèle M avec $K = 2$ types et $I = 3$ groupes, dont deux groupes de spécialistes et un groupe de généralistes (voir la figure 2.1 à la page 16). Les ensembles d'habiletés sont : $\mathcal{S}_1 = \{1\}, \mathcal{S}_2 = \{2\}$ et $\mathcal{S}_3 = \{1,2\}$. Supposons que les coûts des généralistes soient $c_3 = 1.05, q_3^+ = 1.3, q_3^- = 1$, et les coûts des spécialistes soient $c_i = 1, q_i^+ = 1.2, q_i^- = 0.9$ pour $i = 1,2$. Supposons qu'il y a $M = 2$ scénarios possibles $\boldsymbol{\lambda}_1 = (\lambda, 0)$ et $\boldsymbol{\lambda}_2 = (0, \lambda)$ avec probabilité 0.5 chacun, et $\mu_1 = \mu_2 > \lambda$ tels qu'il ne nécessite qu'un seul agent par scénario.

La méthode TS trouve les solutions $\hat{\mathbf{y}}_1 = (1,0,0)$ et $\hat{\mathbf{y}}_2 = (0,1,0)$ dans la première étape. Conséquemment, la solution finale de TS est $\mathbf{y} = (1,1,0), r_{1,2}^- = r_{2,1}^- = 1$ et les autres variables à zéro, pour un coût de 1.1. La solution de TS consiste à planifier 2 agents à l'avance et de retirer un agent en fonction du scénario réalisé. Alors que la solution optimale est d'avoir un seul généraliste : $\mathbf{y} = (0,0,1)$ et les autres variables à zéro, pour un coût de 1.05.

Notons que pour cet exemple, la méthode TS trouverait la solution optimale si nous ajoutions le mécanisme de transfert d'agents (voir les sections 3.3.2.1 et 5.1, aux pages 47 et 97). Par contre, le mécanisme de transfert n'assure pas l'optimalité de TS. Nous démontrons ceci dans le prochain exemple. Par ailleurs, il serait plus approprié de pénaliser les transferts d'agents puisqu'ils ressemblent aux actions de recours, comme le transfert d'un agent correspond au retrait d'un agent d'un groupe et à l'ajout d'un agent dans un autre groupe.

Exemple 2. Une faiblesse de la méthode TS est que la deuxième phase n'utilise aucune information de la simulation. Cette phase ne se base que sur les solutions trouvées durant la première phase (qui utilise par contre la simulation). Nous présentons un exemple où la

méthode TS est sous-optimale même avec les variables de transfert.

Reprenons l'exemple 1, mais avec les coûts suivants : $c_3 = 3.5, q_3^+ = 6, q_3^- = 1$ pour les généralistes, et $c_i = 2, q_i^+ = 4, q_i^- = 1$ pour les groupes de spécialistes $i = 1,2$. Les actions de recours sont fortement pénalisées. Supposons encore $M = 2$ scénarios avec probabilités égales : $\boldsymbol{\lambda}_1 = (\lambda, 0)$ et $\boldsymbol{\lambda}_2 = (0, \lambda)$, mais les généralistes (plus expérimentés et plus coûteux) servent plus rapidement que les spécialistes. Supposons qu'il faut 2 généralistes contre 3 spécialistes pour satisfaire λ dans chaque scénario. La solution optimale est $\mathbf{y} = (0,0,2)$ et 0 pour les autres variables, avec un coût total de 7.

Cependant, la méthode TS trouve les solutions $\hat{\mathbf{y}}_1 = (3,0,0)$ et $\hat{\mathbf{y}}_2 = (0,3,0)$ dans la première étape, puisque 3 agents spécialistes coûtent moins chers que 2 généralistes. Remarquons qu'à l'étape 2, TS ne sait pas que 2 agents généralistes sont suffisants pour satisfaire les deux scénarios. TS trouve la solution $\mathbf{y} = (3,3,0), r_{1,2}^- = r_{2,1}^- = 3$, et les autres variables à 0, pour un coût total de 9. Les variables de transfert ne sont **pas utiles**, car le coût de 3 généralistes est plus cher. La solution de TS est alors sous-optimale.

9.4 Planification avec recours et contrainte par chance

L'objectif du problème (P1) est d'optimiser l'affectation des agents tout en assurant que les minimums de SL seront respectés dans tous les M scénarios. Dans cette section, nous présentons une variante du problème (P1) avec une contrainte par chance où seulement une fraction $0 \leq \Phi \leq 1$ des M scénarios doivent être réalisables. Nous remplaçons les contraintes sur g et g_k dans (P1) par la contrainte de chance :

$$\mathbb{P}_{\boldsymbol{\lambda}} \left[\begin{array}{l} g(\boldsymbol{\lambda}, \mathbf{y} + \mathbf{r}_m^+ - \mathbf{r}_m^-) \geq l, \\ g_k(\boldsymbol{\lambda}, \mathbf{y} + \mathbf{r}_m^+ - \mathbf{r}_m^-) \geq l_k, \forall k \end{array} \right] \geq \Phi,$$

où $\mathbb{P}_{\boldsymbol{\lambda}}$ est la mesure de probabilité sur $\boldsymbol{\lambda}$. Notons que nous retrouvons le problème (P1) si $\Phi = 1$.

Pour adapter notre algorithme de la section 9.3 à la nouvelle contrainte par chance, nous ajoutons M variables binaires ϕ_m telles que $\phi_m = 1$ indique que le scénario m doit être satisfait, autrement $\phi_m = 0$. Le problème (P2$^{(v)}$) devient :

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{I} \hat{\Phi} c_i y_i + \sum_{m=1}^{M} \sum_{i=1}^{I} \left(q_i^+ r_{i,m}^+ - q_i^- r_{i,m}^- \right) p_m \\
\text{sujet à :} \quad & \mathbf{B}_m^{(v)} (\mathbf{y} + \mathbf{r}_m^+ - \mathbf{r}_m^-) \geq \mathbf{b}_m^{(v)} \phi_m, && \forall m, \\
& \sum_{i \in \mathcal{T}_k} w_{k,i,m} \mu_{k,i} \geq \alpha_{k,m} \lambda_{k,m} \phi_m, && \forall k, \forall m, \\
& \sum_{k \in \mathcal{S}_i} w_{k,i,m} = y_i + r_{i,m}^+ - r_{i,m}^-, && \forall i, \forall m, \\
& \sum_{m=1}^{M} p_m \phi_m \geq \Phi, && \\
& \phi_m \in \{0,1\}, && \forall m, \\
& w_{k,i,m} \geq 0, && \forall k, \forall i, \forall m, \\
& r_{i,m}^- \leq y_i, && \forall i, \forall m, \\
& 0 \leq r_{i,m}^+, r_{i,m}^- \leq N \phi_m \text{ et entier}, && \forall i, \forall m, \\
& y_i \geq 0 \text{ et entier}, && \forall i,
\end{aligned}
\tag{P3$^{(v)}$}
$$

où N est un nombre très grand et $\hat{\Phi}$ est un paramètre que nous décrivons un peu plus loin. L'équation $\sum_{m=1}^{M} p_m \phi_m \geq \Phi$ remplace la contrainte par chance. Nous n'ajoutons pas les variables ϕ_m dans la fonction objectif afin de garder le problème linéaire. Nous ajoutons une borne $N\phi_m$ sur les variables de recours afin de forcer ces variables à valoir 0 lorsque $\phi_m = 0$. Il faut ajuster le coût des agents $\mathbf{y}$ en fonction de la fraction de scénarios satisfaits ($\phi_m = 1$), parce que le coût des actions de recours est 0 quand un scénario n'est pas satisfait ($\phi_m = 0$). Pour faire ceci, nous approximons la fraction des scénarios réalisables à l'aide du paramètre $\hat{\Phi}$, où la vraie valeur $\hat{\Phi}$ est $\sum_{m=1}^{M} p_m \phi_m$. Cependant, nous devons approximer $\hat{\Phi}$ puisque nous ne connaissons pas les solutions ϕ_m à l'avance, et nous voulons garder la linéarité de la fonction objectif. Par contre, il y a un cas spécial où nous connaissons exactement $\hat{\Phi}$ avant même de résoudre le problème. Nous présentons ce cas dans la proposition 9.1.

Proposition 9.1. Si les masses de probabilité sont toutes égales, c'est-à-dire $p_m = 1/M$ pour $m = 1, \ldots, M$, alors la solution finale aura $\sum_{m=1}^{M} \phi_m = \lceil M\Phi \rceil$. Ainsi, le choix du paramètre $\hat{\Phi} = \sum_{m=1}^{M} p_m \phi_m = \dfrac{\lceil M\Phi \rceil}{M}$ est exact.

Preuve. La preuve est directe. Par la contrainte $\sum_{m=1}^{M} p_m \phi_m \geq \Phi$ et les probabilités $p_m = 1/M, \forall m$, nous avons $\sum_{m=1}^{M} \phi_m \geq M\Phi$. Puisque ϕ_m sont des variables binaires, nous avons

par conséquent la borne inférieure $\sum_{m=1}^{M} \phi_m \geq \lceil M\Phi \rceil$.

Pour démontrer la borne supérieure, supposons que la solution optimale ait $\sum_{m=1}^{M} \phi_m = (\lceil M\Phi \rceil + N)$, où $N \geq 1$ et entier. En réduisant N variables $\phi_m = 1$ à 0, nous relaxons les contraintes tout en restant réalisable pour (P3$^{(v)}$). Par conséquent, le coût optimal devrait diminuer ou demeurer inchangé. Si le coût optimal reste inchangé, nous avons trouvé une autre solution optimale telle que $\sum_{m=1}^{M} \phi_m = \lceil M\Phi \rceil$, autrement nous avons trouvé une meilleure solution optimale (ce qui est une contradiction avec l'hypothèse d'optimalité).

La proportion (minimale) de scénarios réalisables que nous obtenons avec la solution optimale est donc $\lceil M\Phi \rceil / M$. □

Si les masses de probabilité p_m ne sont pas égales ou si les valeurs sont grandes (donnant une couverture trop grossière de Φ), nous pouvons fractionner les p_m en dupliquant les scénarios afin de mieux uniformiser la distribution. Par contre, le problème linéaire contiendra plus de variables.

Nous présentons une autre formulation du problème (P3$^{(v)}$) sans le paramètre $\hat{\Phi}$. Remarquons que nous ajoutons le paramètre $\hat{\Phi}$ parce que nous annulons les actions de recours du scénario m lorsque la variable binaire $\phi_m = 0$. Si nous ne forçons pas les variables de recours à 0, alors la solution optimale serait de retirer tous les agents de $\mathbf{y}$, soient $r^-_{i,m} = y_i$ et $r^+_{i,m} = 0$ lorsque $\phi_m = 0$. Par contre, ceci n'élimine pas complètement le coût de $\mathbf{y}$ car $c_i > q^-_i$. Nous devons alors réajuster le coût à l'aide de variables supplémentaires $\hat{r}^-_{i,m}$ afin de diminuer le coût de $(c_i - q^-_i)\hat{r}^-_{i,m}$ si $\phi_m = 0$. Remarquons que nous n'aurions pas besoin des variables $\hat{r}^-_{i,m}$ si $c_i = q^-_i$. Mais ceci n'est pas réaliste, parce qu'un gestionnaire n'aura qu'à planifier beaucoup trop d'agents $\mathbf{y}$ à l'avance, et de retirer le surplus d'agents lors du

recours. La nouvelle version du problème avec recours et contrainte par chance est :

$$
\begin{aligned}
\min \quad & \sum_{i=1}^{I} c_i y_i + \sum_{m=1}^{M} \sum_{i=1}^{I} \left(q_i^+ r_{i,m}^+ - q_i^- r_{i,m}^- - (c_i - q_i^-)\hat{r}_{i,m}^- \right) p_m \\
\text{sujet à :} \quad & \mathbf{B}_m^{(v)}(\mathbf{y} + \mathbf{r}_m^+ - \mathbf{r}_m^-) \geq \mathbf{b}_m^{(v)} \phi_m, && \forall m, \\
& \sum_{i \in \mathcal{T}_k} w_{k,i,m} \mu_{k,i} \geq \alpha_{k,m} \lambda_{k,m} \phi_m, && \forall k, \forall m, \\
& \sum_{k \in \mathcal{S}_i} w_{k,i,m} = y_i + r_{i,m}^+ - r_{i,m}^-, && \forall i, \forall m, \\
& \sum_{m=1}^{M} p_m \phi_m \geq \Phi, \\
& \phi_m \in \{0,1\}, && \forall m, \\
& w_{k,i,m} \geq 0, && \forall k, \forall i, \forall m, \\
& \hat{r}_{i,m}^- \leq y_i, && \forall i, \forall m, \\
& 0 \leq \hat{r}_{i,m}^- \leq N(1 - \phi_m), && \forall i, \forall m, \\
& r_{i,m}^+ \geq 0 \text{ et entier}, && \forall i, \forall m, \\
& 0 \leq r_{i,m}^- \leq y_i \text{ et entier}, && \forall i, \forall m, \\
& y_i \geq 0 \text{ et entier}, && \forall i,
\end{aligned}
\tag{P4$^{(v)}$}
$$

où N est un grand nombre. Le dernier terme de la fonction objectif sert à réajuster le coût de la solution. Nous n'avons pas besoin d'expliciter les contraintes d'intégralité des variables $\hat{r}_{i,m}^-$ puisqu'elles sont bornées par des variables entières et 0, et que les $\hat{r}_{i,m}^-$ n'apparaissent que dans la fonction objectif. La variable $\hat{r}_{i,m}^- = y_i$ si $\phi_m = 0$, autrement $\hat{r}_{i,m}^- = 0$. Nous devrons diviser le coût final par $\sum_{m=1}^{M} p_m \phi_m$ pour obtenir le coût moyen conditionnel aux scénarios m réalisables (où $\phi_m = 1$).

Avec le problème (P4$^{(v)}$), il est préférable que les probabilités p_m soient uniformes, et de dupliquer les scénarios m si les p_m sont larges (ou non uniformes) afin de couvrir le plus précisément possible la proportion Φ.

9.5 Implémentation de l'algorithme

L'extension directe de la méthode de coupe de Ceẓik et L'Ecuyer [37] génère une coupe de sous-gradient pour chaque scénario irréalisable à chaque itération. Il faut donc faire au

plus $(I+1)M$ simulations par itération, ce qui consomme beaucoup de temps d'exécution. Cependant, il est probablement possible de faire moins de simulations, s'il existe une certaine corrélation entre les scénarios. Le raisonnement est que si les taux d'arrivée de deux scénarios diffèrent de peu, alors leurs solutions seront probablement très similaires. Dans cette section, nous présentons quelques idées afin de réduire le nombre de simulations. Cette section s'applique pour l'optimisation des problèmes (P2$^{(v)}$), (P3$^{(v)}$) et (P4$^{(v)}$) qui se basent tous sur les coupes par sous-gradients. Nous supposons que seulement les taux d'arrivée et le nombre d'agents peuvent changer dans les différents scénarios.

9.5.1 Cas spécial : un facteur d'achalandage unique

Nous commençons par le cas spécial où il n'y a qu'un facteur d'achalandage unique β affectant tous les types d'appels. Le facteur β est une variable aléatoire prenant les valeurs $\beta_1,\ldots,\beta_M$ avec probabilités $p_1,\ldots,p_M$. Le taux d'arrivée réel du type k dans le scénario m est $\lambda_{k,m}=\beta_m\lambda_k$, où λ_k est le taux d'arrivée de base fixe. Sans perte de généralité, supposons les M scénarios soient ordonnés par ordre croissant tel que $\beta_1\leq\beta_2\leq\cdots\leq\beta_M$ et il existe un indice $\hat{m}$ tel que $\hat{m}=\min\{i:\sum_{m=1}^{i}p_m=\Phi\}$ (l'égalité de la somme est importante). Dans le cas où la somme n'égale pas exactement Φ, nous pouvons dupliquer le scénario $m'=\min\{i:\sum_{m=1}^{i}p_m>\Phi\}$ en deux scénarios identiques m'_1 et m'_2 mais avec des masses de probabilité telles que $\sum_{m=1}^{m'_1}p_m=\Phi$ et $p_{m'_1}+p_{m'_2}=p_{m'}$, et nous obtenons $\hat{m}=m'_1$.

Si la contrainte est d'assurer la réalisabilité des contraintes de SL avec probabilité Φ, alors il serait raisonnable d'optimiser uniquement les scénarios 1 à $\hat{m}$. C'est-à-dire de retirer les scénarios $\hat{m}+1$ à M du problème et de résoudre avec (P2$^{(v)}$), ou bien de fixer $\phi_m=1$ pour $m=1,\ldots,\hat{m}$, et fixer $\phi_m=0$ pour $m>\hat{m}$ dans les problèmes (P3$^{(v)}$) et (P4$^{(v)}$). L'idée se base sur les formules d'Erlang telles que pour obtenir le même SL, le nombre d'agents doit augmenter de manière monotone quand le taux d'arrivée augmente. Nous pouvons aussi penser aux modèles fluides (où les problèmes deviennent déterministes). Étant donné un facteur d'achalandage unique pour tous les types d'appels, une solution fluide réalisable pour le scénario m (en supposant un problème fluide) sera également réalisable pour tous les scénarios $j<m$. Par conséquent, les solutions des scénarios $1,\ldots,\hat{m}$ ne peuvent pas coûter plus chers que ceux des scénarios $\hat{m}+1,\ldots,M$. Il serait alors raisonnable de

satisfaire uniquement les scénarios 1 à $\hat{m}$.

Nous pouvons donc réduire le nombre de simulations en ignorant les scénarios $\hat{m}+1$ à M. Cependant, générer une coupe de sous-gradient pour les $\hat{m}$ scénarios restants peut demander encore beaucoup de simulations. Dans la prochaine section, nous proposons de générer des coupes que pour quelques scénarios à chaque itération.

9.5.2 Simulation partielle des scénarios

L'idée se base sur l'hypothèse que les scénarios présentent des similitudes telles que les solutions ne varient pas énormément d'un scénario à l'autre. Dans ce cas, nous pourrions éviter de générer une coupe de sous-gradient pour chaque scénario irréalisable à chaque itération de l'algorithme.

Soit $\bar{M}^{(v)} \in \{1,2,\ldots,M\}$ le nombre de scénarios irréalisables à l'itération v. Normalement, nous exécutons $(I+1)\bar{M}^{(v)}$ simulations pour estimer les $\bar{M}^{(v)}$ sous-gradients. Nous proposons de générer $\tilde{M} = \lceil \gamma \bar{M}^{(v)} \rceil$ coupes par sous-gradients à l'itération v demandant $(I+1)\tilde{M}$ simulations, où $0 < \gamma \leq 1$ est un paramètre. Les $\tilde{M}$ scénarios à simuler peuvent être choisis aléatoirement parmi les $\bar{M}^{(v)}$ scénarios irréalisables. Le choix peut aussi se faire par une sélection à la ronde (ou "*round-robin*") ou à l'aide d'une pondération sur les niveaux de violation des contraintes.

9.6 Futur projet de recherche

Dans ce chapitre, nous avons présenté des idées pour l'optimisation stochastique de l'affectation des agents avec des actions de *recours à l'avance* dans un centre d'appels multi-compétence. Le terme *recours à l'avance* signifie que les actions de recours doivent être exécutées avant la journée d'opération. À notre connaissance, il n'existe pratiquement aucune étude sur la planification stochastique avec recours pour les centres d'appels composés de plusieurs types d'appels et groupes d'agents, et avec des volumes d'appels et des politiques de routage réalistes.

Notre problématique suppose que la planification des agents peut être réalisée longtemps à l'avance (quelques semaines par exemple) basée sur des prévisions stochastiques

des volumes d'appels. À l'approche de la journée d'opération (disons la veille), nous disposons des prévisions exactes des taux d'arrivée, mais changer l'affectation (ajouter ou retirer des agents) impose des coûts supplémentaires. Ce type de problème stochastique est beaucoup plus réaliste, parce qu'en pratique, les gestionnaires doivent toujours planifier selon des prévisions probabilistes des volumes d'appels. Le taux d'absentéisme est aussi un aspect stochastique important dans la planification, mais nous ne l'avons pas inclus dans notre problématique.

Nous avons présenté une extension de l'algorithme d'affectation des agents de Ceẓik et L'Ecuyer [37] pour inclure les actions de recours. Contrairement à leur problème, nous considérons des taux d'arrivée stochastiques dans la modélisation du problème d'affectation. Nous utilisons une approche par scénario où nous ajoutons des variables de recours (associées aux ajouts et aux retraits des agents). Nous avons aussi considéré une version du problème avec une contrainte par chance pour la satisfaction des niveaux de service.

Nous n'avons cependant pas eu le temps d'implémenter et de tester nos méthodes. Ces idées constituent un futur projet de recherche.

CHAPITRE 10

CONCLUSION

La gestion d'un centre d'appels multi-compétences est un problème difficile à cause de sa nature stochastique et des multiples niveaux de décisions. Dans cette thèse, nous avons étudié différents problèmes de décisions : les problèmes d'affectation des agents, de planification des horaires (quarts de travail), et du routage des appels aux agents. Nous avons aussi étudié quelques modèles d'approximation de centre d'appels. Nous résumons ici les principales contributions de chaque chapitre de cette thèse.

Dans la pratique, les centres d'appels sont souvent modélisés entièrement par des processus markoviens. Nous pouvons donc approximer ces centres d'appels à l'aide de chaînes de Markov en temps continu (CMTC). Dans le chapitre 4, nous avons développé un simulateur de la chaîne de Markov en temps discret (CMTD) imbriquée de la CMTC. En principe, notre simulateur simule la CMTC avec des compteurs au lieu d'objets de programmation, ce qui le rend plus rapide qu'un simulateur traditionnel par événements discrets (ED).

En appliquant la méthode d'uniformisation d'une CMTC, nous rendons les instances de transition de la CMTC indépendantes des transitions effectuées par la CMTD imbriquée. Les instances de transition peuvent être obtenues simplement en générant le nombre de transitions par une loi de Poisson, puis de répartir les instances de ces transitions aléatoirement et uniformément sur l'horizon de temps à simuler. La CMTD est simulée par la suite

indépendamment des temps de transition. Un inconvénient de l'uniformisation est l'introduction de transitions fictives qui ralentissent la simulation. Il est possible que le simulateur CMTD soit plus lent qu'un simulateur ED, si l'uniformisation est mal appliquée.

Nous avons appliqué la simulation CMTD pour le problème d'affectation des agents, optimisé avec l'algorithme de Ceẓik et L'Ecuyer [37]. L'optimisation génère des coupes linéaires basées sur les sous-gradients de la fonction du niveau de service (SL). Comme nous estimons les sous-gradients à l'aide de la simulation, ces sous-gradients contiennent du bruit et ils peuvent fausser l'algorithme d'optimisation. Pour réduire le bruit, nous avons étudié principalement des méthodes pour mieux synchroniser le simulateur CMTD par des variables aléatoires communes (VAC). Nous proposons deux méthodes pour utiliser les VAC. La première méthode consiste à utiliser les VAC uniquement au moment d'estimer un sous-gradient lors de la génération d'une coupe. Elle minimise le nombre de transitions fictives et chaque sous-gradient est estimé avec des VAC différentes. L'autre méthode est d'incrémenter le taux total d'uniformisation itérativement au besoin. À la différence de la première méthode, elle ne permet pas de réduire le taux total de transition. Par contre, elle permet de ré-utiliser les mêmes VAC pour estimer plusieurs sous-gradients. Ces deux méthodes fonctionnent bien.

Dans nos expériences numériques, nous avons optimisé des modèles de centres d'appels markoviens et non markoviens. Pour les problèmes non markoviens, nous avons combiné l'usage des simulateurs CMTD et ED. L'algorithme débute l'optimisation avec le simulateur CMTD, puis il change pour le simulateur ED vers la fin de l'optimisation. Les résultats numériques ont montré une amélioration des performances d'optimisation dans tous les cas, due principalement à l'accélération du temps de simulation.

Dans le chapitre 5, nous avons étudié le problème de planification des quarts de travail pour un centre d'appels multi-compétences. Nous avons proposé une adaptation des algorithmes de coupes linéaires et de simulation de Ceẓik et L'Ecuyer [37] et Atlason et al. [12]. Nous avons inclus le mécanisme de transfert d'agents proposé par Bhulai et al. [23] qui permet de transférer temporairement un agent vers un groupe requérant moins d'habiletés.

Les difficultés majeures de ce problème sont le nombre élevé de variables entières et

l'approximation des sous-gradients par la simulation. Le nombre élevé de variables entières nous contraint généralement à résoudre une version relaxée du problème avec des variables continues. Nous avons étudié des méthodes pour arrondir judicieusement les variables relaxées. Nous proposons d'arrondir les variables en fonction d'un seuil fractionnaire fixe durant l'optimisation, puis d'optimiser ce seuil par une recherche binaire lors de la dernière itération. Nous nous sommes rendus compte qu'il n'était pas vraiment important que la solution arrondie soit réalisable pour le problème relaxé, car d'autres coupes s'ajouteront au cours de l'optimisation. Nous avons trouvé qu'il était plus efficace de garder le problème relaxé, mais d'arrondir la solution juste pour estimer les sous-gradients (car le simulateur doit simuler des nombres entiers d'agents). Une autre difficulté provient du mécanisme de transfert d'agents. Il est difficile d'arrondir les variables de transfert et de garder la solution cohérente, car les transferts peuvent être exécutés en chaîne à travers plusieurs groupes. Nous avons développé un algorithme pour maintenir la cohérence entre les transferts d'agents et le nombre d'agents.

À la fin de l'algorithme, nous exécutons une recherche locale pour corriger et améliorer la solution. Nous avons étendu les méthodes de recherche locale d'Avramidis et al. [18]. La première méthode incrémente le nombre d'agents afin de trouver une solution réalisable sous longue simulation. La deuxième méthode décrémente le nombre d'agents afin de réduire le coût des agents, tout en gardant la solution réalisable. La troisième méthode tente de réduire le coût en transférant des agents vers des groupes moins coûteux ou en affectant des quarts de travail moins coûteux aux agents. Les exemples numériques ont montré que notre algorithme performe mieux que l'approche traditionnelle à deux étapes.

Le chapitre 6 est consacré à l'étude des politiques de routage. L'affectation des agents est une donnée fixe du problème. Nous avons proposé une nouvelle politique de routage basé sur des poids (BP). Nous avons considéré différentes règles pour calculer ces poids : en fonction des temps d'attente des appels, des temps d'inactivité des agents ou du nombre d'agents libres. Cette politique de poids est innovatrice, car elle mélange les informations des appels et des agents. Ceci permet d'avoir une seule règle pour contrôler le routage d'un nouvel appel vers un agent inoccupé, ainsi que d'un agent qui se libère vers un appel en attente. Des poids négatifs permettent de créer des conditions de délai, où un agent demeure

libre même s'il y a des appels en attente auxquels il peut répondre. La politique BP peut simuler des politiques plus simples comme le routage par priorités et temps de délai.

D'autre part, nous avons proposé une adaptation linéaire de la règle $c\mu$ généralisée pour des fonctions de type boîte noire, qui ne sont pas nécessairement convexes. Nous avons aussi étudié des routages basés sur la pratique, comme le routage par priorités, temps de délai et seuils d'agents libres.

Dans nos problèmes d'optimisation, l'objectif était de minimiser une fonction de pénalité basée sur des mesures de performance et d'équité, sans contrainte dure. Au lieu de prendre l'approche traditionnelle des approximations ou simplifications du centre d'appels, nous désirons optimiser le routage de tout centre d'appels pouvant être modélisé par un simulateur et une fonction objectif de type boîte noire. Notre travail représente une première étude sur l'optimisation du routage pour des centres d'appels multi-compétences réalistes.

Nous avons proposé un algorithme génétique modifié (AGM) pour l'optimisation de tous les paramètres des politiques de routage considérées dans ce chapitre. L'AGM est très flexible : il peut optimiser les variables entières et continues, ainsi que les problèmes combinatoires. Dans la section numérique, nous avons comparé les performances optimisées de notre politique BP avec celles des politiques tirées de la pratique et la littérature, à travers plusieurs exemples. Nous avons considéré des exemples plus élaborés avec des processus d'arrivée Poisson-gamma et des durées de service de lois log-normales. Les résultats numériques ont montré que notre politique BP réussit en général aussi bien ou mieux que la meilleure des autres politiques.

Dans le chapitre 7, nous nous sommes intéressés au développement d'un algorithme heuristique pour l'affectation des agents sans l'aide d'un simulateur de centre d'appels. Nous agrégeons le problème en se basant sur les observations de Wallace et Whitt [132]. En supposant des durées de service indépendantes des types d'appels et des groupes d'agents, ils observent qu'un centre d'appels, où chaque agent a 2 ou 3 habiletés judicieusement choisies, peut performer presque aussi bien qu'un centre d'appels où tous les agents possèdent toutes les habiletés. Nous disons qu'un tel centre d'appels est *efficace*. Puis nous simulons les états d'une file d'attente afin de générer les scénarios d'un problème linéaire stochastique. Les contraintes sur les niveaux de service sont converties en contraintes sur

les probabilités de délai. L'approche par agrégation et désagrégation est une source potentielle d'erreurs, et requiert que les contraintes des types d'appels soient identiques. Nous avons utilisé une approche d'optimisation stochastique par scénario pour minimiser le coût des agents sous la contrainte de satisfaire la probabilité de délai. Nous avons proposé deux méthodes de relaxation afin d'optimiser le problème stochastique à nombres entiers plus rapidement.

Les exemples numériques ont montré de bonnes performances d'optimisation pour les centres d'appels "efficaces" et une bonne satisfaction des mesures de performance agrégées. Une difficulté importante était de déterminer la politique de routage optimale, car notre algorithme optimise l'affectation des agents indépendamment du routage. Nous avons obtenu de bons résultats en choisissant et en optimisant la politique de routage BP, que nous avons proposée au chapitre 6.

Futures directions de recherche

Dans le chapitre 8, nous avons proposé une variante au modèle de CMTC de Koole et al. [90] qui est basé sur le temps d'attente du client à la tête de la file. Les modèles d'approximation traditionnels représentent souvent l'état de la CMTC en fonction du nombre d'appels dans le système. Un inconvénient de ces modèles est l'information limitée sur les temps d'attente des clients. Koole et al. [90] remplacent le compteur du nombre de clients en attente par un compteur qui représente le temps d'attente du client à la tête de la file. Un état représente le nombre de serveurs libres ou un compteur de temps d'attente lorsque tous les serveurs sont occupés. Nous avons proposé des modifications simples au modèle original de Koole et al. [90] en changeant les transitions d'incrémentation de la CMTC afin de toujours inclure les taux d'arrivée. Nous avons comparé notre variante avec le modèle original pour des exemples du système $M/M/n$ (un type de client, un groupe de n serveurs et sans abandon). Les résultats numériques ont montré que notre modèle donne constamment de meilleures approximations que le modèle original.

Il n'y a pas d'avantage réel à utiliser ces modèles pour approximer les performances d'une file $M/M/n$, puisqu'il existe déjà la formule d'Erlang C. Une direction de recherche

future est d'appliquer notre variante pour approximer des centres d'appels avec plusieurs types d'appels et groupes d'agents. Il existe très peu de méthodes d'approximation pour les centres d'appels multi-compétences, parce que la CMTC devient rapidement immense et complexe.

Dans le chapitre 9, nous avons considéré le problème d'affectation stochastique des agents avec actions de recours. La problématique consiste à affecter des agents à l'avance à moindre coût, à partir de prévisions probabilistes. Puis, il y a la possibilité de modifier ultérieurement les affectations (ajouter ou retirer des agents), en payant des coûts de pénalités quand les prévisions révisées et plus précises deviennent disponibles. Nous avons proposé des idées pour adapter l'algorithme de coupes de Ceẓik et L'Ecuyer [37] pour optimiser ce type de problème stochastique avec recours. Nous avons aussi considéré une version du problème avec une contrainte par chance, où les seuils de niveaux de service doivent être atteints avec une certaine probabilité.

Dans ce chapitre, nous avons présenté les idées de base d'un algorithme d'affectation stochastique avec recours pour les centres d'appels multi-compétences. Ces idées constituent un futur projet de recherche. Une autre direction de recherche est de considérer les options de recours intra-journaliers. Ce type de problème est plus réaliste, mais aussi plus compliqué.

Finalement, revenons au problème d'optimisation du routage. Contrairement aux problèmes de planification des agents, le routeur doit généralement effectuer des milliers ou plus de décisions de routage dans une journée. Il est donc primordial d'optimiser les paramètres de routage rapidement dès qu'un imprévu survient. Or nous nous sommes concentrés sur l'étude des propriétés et des performances de la politique de routage BP, et nous avons choisi un algorithme d'optimisation général sans nous soucier particulièrement du temps d'exécution. Une direction de recherche importante est le développement d'un algorithme rapide pour l'optimisation des paramètres de routage.

BIBLIOGRAPHIE

[1] M.A. Abramson, C. Audet, G. Couture, J.E. Dennis, Jr., S. Le Digabel et C. Tribes. The NOMAD project. Logiciel disponible sur `http://www.gerad.ca/nomad`, 2013.

[2] O. Z. Akşin, M. Armony et V. Mehrotra. The modern call center : A multi-disciplinary perspective on operations management research. *Production and Operations Management*, 16(6):665–688, 2007.

[3] S. Aldor-Noiman, P. Feigin et A. Mandelbaum. Workload forecasting for a call center : Methodology and a case study. *Annals of Applied Statistics*, 3:1403–1447, 2009.

[4] B. H. Andrews et S. M. Cunningham. L.L. Bean improves call-center forecasting. *Interfaces*, 25(6):1–13, 1995.

[5] B. H. Andrews et H. L. Parsons. L.L. Bean chooses a telephone agent scheduling system. *Interfaces*, 19(6):1–9, 1989.

[6] M. Armony. Dynamic routing in large-scale service systems with heterogeneous servers. *Queueing Systems*, 51(3–4):287–329, 2005.

[7] M. Armony et A. R. Ward. Fair dynamic routing in large-scale heterogeneous-server systems. *Operations Research*, 58(3):624–637, 2010.

[8] M. Armony et A. R. Ward. Blind fair routing in large-scale service systems with heterogeneous customers and servers. *Operations Research*, 61(1):228–243, 2013.

[9] S. Asmussen et P. W. Glynn. *Stochastic Simulation*. Springer-Verlag, New York, 2007.

[10] R. Atar. Scheduling control for queueing systems with many servers : Asymptotic optimality in heavy traffic. *Annals of Applied Probability*, 15(4):2606–2650, 2005.

[11] R. Atar, C. Giat et N. Shimkin. The $c\mu/\theta$ rule for many-server queues with abandonment. *Operations Research*, 58(5):1427–1439, 2010.

[12] J. Atlason, M. A. Epelman et S. G. Henderson. Call center staffing with simulation and cutting plane methods. *Annals of Operations Research*, 127:333–358, 2004.

[13] J. Atlason, M. A. Epelman et S. G. Henderson. Optimizing call center staffing using simulation and analytic center cutting plane methods. *Management Science*, 54(2): 295–309, 2008.

[14] C. Audet et J. E. Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17:188–217, 2006.

[15] Avaya. Avaya business advocate release 3 version 11 user guide, Mai 2002. Disponible sur `http://support.avaya.com/elmodocs2/multivantage/233823_2/210711_1/210711_1.pdf`. Accédé en janvier 2012.

[16] Avaya. Programming call vectors in avaya aura call center, Juin 2010. Disponible sur `http://support.avaya.com/css/P8/documents/100081995`. Accédé en janvier 2012.

[17] A. N. Avramidis, W. Chan, M. Gendreau, P. L'Ecuyer et O. Pisacane. Optimizing daily agent scheduling in a multiskill call centers. *European Journal of Operational Research*, 200(3):822–832, 2010.

[18] A. N. Avramidis, W. Chan et P. L'Ecuyer. Staffing multi-skill call centers via search methods and a performance approximation. *IIE Transactions*, 41:483–497, 2009.

[19] A. N. Avramidis, A. Deslauriers et P. L'Ecuyer. Modeling daily arrivals to a telephone call center. *Management Science*, 50(7):896–908, 2004.

[20] A. Bassamboo, J. M. Harrison et A. Zeevi. Design and control of a large call center : Asymptotic analysis of an LP-based method. *Operations Research*, 54(3):419–435, 2006. ISSN 0030-364X.

[21] A. Bassamboo et A. Zeevi. On a data-driven method for staffing large call centers. *Operations Research*, 57(3):714–726, 2009. ISSN 0030-364X.

[22] K. C. Benson, D. Goldsman et Amy R. Pritchett. Ranking and selection procedures for simulation. Dans *Proceedings of the 2006 Winter Simulation Conference*, pages 179–185. IEEE Press, 2006.

[23] S. Bhulai, G. Koole et A. Pot. Simple methods for shift scheduling in multi-skill call centers. *Manufacturing and Service Operations Management*, 10:411–420, 2008.

[24] J. R. Birge et F. Louveaux. *An introduction to stochastic programming*. Springer, New York, 1997.

[25] Z. I. Botev, D. P. Kroese, R. Y. Rubinstein et P. L'Ecuyer. The cross-entropy method for optimization. Dans *Handbook of Statistics, Volume 31 : Machine Learning*. North Holland, 2013.

[26] L. Brown, N. Gans, A. Mandelbaum, A. Sakov, H. Shen, S. Zeltyn et L. Zhao. Statistical analysis of a telephone call center : A queueing-science perspective. *Journal of the American Statistical Association*, 100:36–50, 2005.

[27] E. S. Buffa, M. J. Cosgrove et B. J. Luce. An integrated work shift scheduling system. *Decision Sciences*, 7(4):620–630, 1976.

[28] E. Buist. *Simulation des centres de contacts*. Thèse de doctorat, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Canada, 2009.

[29] E. Buist. User's guide for ContactCenters Simulation Library. Generic simulator for blend and multi-skill call centers, 2013. Disponible sur `http://www.iro.umontreal.ca/~simardr/contactcenters/doc/pdf/guidemsk.pdf`. Accédé en février 2013.

[30] E. Buist, W. Chan et P. L'Ecuyer. Speeding up call center simulation and optimization by Markov chain uniformization. Dans *Proceedings of the 2008 Winter Simulation Conference*, pages 1652–1660, Pistacaway, New-Jersey, 2008. IEEE Press.

[31] E. Buist et P. L'Ecuyer. A Java library for simulating contact centers. Dans M. E. Kuhl, N. M. Steiger, F. B. Armstrong et J. A. Joines, éditeurs, *Proceedings of the 2005 Winter Simulation Conference*, pages 556–565. IEEE Press, 2005.

[32] Bureau of Labor Statistics. Absences from work of employed full-time wage and salary workers by occupation and industry (Table 47), 2010. Current Population Survey – Employment and Earnings. United States Department of Labor. Disponible sur `http://www.bls.gov/cps/tables.htm`. Accédé en septembre 2011.

[33] Bureau of Labor Statistics. Occupational employment and wages, 2007, Septembre 2010. United States Department of Labor. Disponible sur `http://www.bls.gov/oes/2007/may/oes_nat.htm`. Accédé en septembre 2011.

[34] Bureau of Labor Statistics. Occupational employment and wages, May 2010 – Customer service representatives, Mai 2011. Occupational Employment Statistics. United States Department of Labor. Disponible sur `http://www.bls.gov/oes/current/oes434051.htm`. Accédé en septembre 2011.

[35] Bureau of Labor Statistics. Occupational outlook handbook – Customer service representatives, 2011. Edition 2010-11. United States Department of Labor. Disponible sur `http://www.bls.gov/oco/ocos280.htm`. Accédé en septembre 2011.

[36] Bureau of Labor Statistics. An overview of U.S. occupational employment and wages in 2010, Juin 2011. Occupational Employment Statistics (OES) Highlights. United States Department of Labor. Disponible sur `http://www.bls.gov/oes/highlight_2010.pdf`. Accédé en septembre 2011.

[37] M. T. Ceẓik et P. L'Ecuyer. Staffing multiskill call centers via linear programming and simulation. *Management Science*, 54(2):310–323, 2008.

[38] W. Chan, G. Koole et P. L'Ecuyer. Dynamic call center routing policies using call waiting and agent idle times. *Manufacturing & Service Operations Management*, 2013. En révision.

[39] N. Channouf et P. L'Ecuyer. A normal copula model for the arrival process in a call center. *International Transactions in Operational Research*, 19:771–787, 2012.

[40] N. Channouf, P. L'Ecuyer, A. Ingolfsson et A. N. Avramidis. The application of forecasting techniques to modeling emergency medical system calls in Calgary, Alberta. *Health Care Management Science*, 10(1):25–45, 2007.

[41] Ph. Chevalier, R. A. Shumsky et N. Tabordon. Routing and staffing in large call centers with specialized and fully flexible servers. Rapport technique, Simon Graduate School of Business, University of Rochester, 2004.

[42] Cisco Systems, Inc. Scripting and media routing guide for cisco unified contact center enterprise, release 8.5(1), juillet 2008. Disponible sur `http://www.cisco.com/en/US/docs/voice_ip_comm/cust_contact/contact_center/ipcc_enterprise/ipccenterprise8_5_1/user/guide/ipce85sg.pdf`. Accédé en septembre 2011.

[43] Cisco Systems, Inc. Configuration guide for cisco unified ICM/contact center enterprise and hosted, release 8.5(2), July 2011. Disponible sur `http://www.cisco.com/en/US/docs/voice_ip_comm/cust_contact/contact_center/ipcc_enterprise/ipccenterprise8_5_2/configuration/guide/icm85config.pdf`. Accédé en septembre 2011.

[44] Conseil de la radiodiffusion et des télécommunications canadiennes (CRTC). Normes définitives d'utilisation d'indicateurs de la qualité du service pour la réglementation des compagnies de téléphone et autres questions connexes, 2000. Décision CRTC 2000-24. Voir `http://www.crtc.gc.ca/fra/archive/2000/DT2000-24.htm`. Accédé le 09/09/2011.

[45] R. B. Cooper. *Introduction to Queueing Theory*. North-Holland, New York, NY, seconde édition, 1981.

[46] D. R. Cox et W. L. Smith. *Queues*. Monographs on Statistical Subjects. Chapman & Hall, 1961. ISBN 9780412109300.

[47] G. B. Dantzig. A comment on Edie's "traffic delays at toll booths". *Operations Research*, 2(3):339–341, 1954.

[48] H. A. David. *Order Statistics*. Wiley, seconde édition, 1981.

[49] P.-T. de Boer, D. P. Kroese, S. Mannor et R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

[50] A. Deslauriers. Modélisation et simulation d'un centre d'appels téléphoniques dans un environnement mixte. Mémoire de maîtrise, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Canada, 2003.

[51] A. Deslauriers, J. Pichitlamken, P. L'Ecuyer, A. Ingolfsson et A. N. Avramidis. Markov chain models of a telephone call center with call blending. *Computers and Operations Research*, 34(6):1616–1645, 2007.

[52] R. Fletcher. *Practical Methods of Optimization*. Wiley, seconde édition, 1987.

[53] B. L. Fox et P. W. Glynn. Discrete-time conversion for simulating finite-horizon Markov processes. *SIAM Journal on Applied Mathematics*, 50:1457–1473, 1990.

[54] G. J. Franx, G. Koole et A. Pot. Approximating multi-skill blocking systems by hyper-exponential decomposition. *Performance Evaluation*, 63:799–824, 2006.

[55] A. Fredericks. Congestion in blocking systems-a simple approximation technique. *The Bell System Technical Journal*, 59(6):805–827, 1980.

[56] M. C. Fu. Optimization by simulation : A review. *Annals of Operations Research*, 53:199–247, 1994.

[57] M. C. Fu. Gradient estimation. Dans S. G. Henderson et B. L. Nelson, éditeurs, *Simulation*, Handbooks in Operations Research and Management Science, pages 575–616. Elsevier, Amsterdam, The Netherlands, 2006. Chapitre 19.

[58] N. Gans, G. Koole et A. Mandelbaum. Telephone call centers : Tutorial, review, and research prospects. *Manufacturing and Service Operations Management*, 5:79–141, 2003.

[59] N. Gans, H. Shen, Y.-P. Zhou, N. Korolev, A. McCord et H. Ristock. Parametric stochastic programming models for call-center workforce scheduling. 2012. Manuscrit.

[60] N. Gans et Y.-P. Zhou. A call-routing problem with service-level constraints. *Operations Research*, 51(2):255–271, 2003.

[61] O. Garnett, A. Mandelbaum et M. Reiman. Designing a call center with impatient customers. *Manufacturing and Service Operations Management*, 4(3):208–227, 2002. ISSN 1253-4614.

[62] F. Glover. Tabu search– Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[63] F. Glover. Tabu search– Part II. *ORSA Journal on Computing*, 2(1):4–32, 1990.

[64] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Boston, MA, USA, première édition, 1989. ISBN 0201157675.

[65] D. Goldsman, S.-H. Kim, W. S. Marshall et Barry L. Nelson. Ranking and selection for steady-state simulation : Procedures and perspectives. *INFORMS Journal on Computing*, 14(1):2–19, 2002.

[66] W. K. Grassmann. Finding transient solutions in Markovian event systems through randomization. Dans W. J. Stewart, éditeur, *Numerical Solutions of Markov Chains*, pages 357–371. Marcel Dekker Inc, New York, NY, 1991.

[67] L. V. Green et P. J. Kolesar. The pointwise stationary approximation for queues with nonstationary arrivals. *Management Science*, 37(1):84–97, 1991.

[68] L. V. Green, P. J. Kolesar et J. Soares. Improving the SIPP approach for staffing service systems that have cyclic demands. *Operations Research*, 49(4):549–564, 2001.

[69] D. Gross et D. R. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research*, 32(2): 343–361, 1984.

[70] I. Gurvich et W. Whitt. Queue-and-idleness-ratio controls in many-server service systems. *Mathematics of Operations Research*, 34(2):363–396, 2009. ISSN 0364-765X.

[71] I. Gurvich et W. Whitt. Service-level differentiation in many-server service systems via queue-ratio routing. *Operations Research*, 58(2):316–328, 2010. ISSN 0030-364X.

[72] S. Halfin et W. Whitt. Heavy-traffic limits for queues with many exponential servers. *Operations Research*, 29:567–588, 1981.

[73] R. Hallowell. The relationships of customer satisfaction, customer loyalty, and profitability : An empirical study. *International Journal of Service Industry Management*, 7(4):27–42, 1996.

[74] P. Hansen et N. Mladenovic. Variable neighborhood search : Principles and applications. *European Journal of Operations Research*, 130:449–467, 2001. Invited Review.

[75] J. M. Harrison et A. Zeevi. A method for staffing large call centers based on stochastic fluid models. *Manufacturing and Service Operations Management*, 7(1):20–36, 2005.

[76] J. L. Heskett, T. O. Jones, G. W. Loveman, W. E. Sasser Jr. et L. A. Schlesinger. Putting the service-profit chain to work. *Harvard Business Review*, pages 164–174, March–April 1994. Reprint 94204.

[77] Frederick S. Hillier et Gerald J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, huitième édition, 2005. ISBN 0-07-252744-7.

[78] T. J. Holloran et J. E. Byrn. United Airlines station manpower planning system. *Interfaces*, 16(1):39–50, 1986.

[79] D. Holman. Employee wellbeing in call centres. *Human Resource Management Journal*, 12(4):35–50, 2002.

[80] R. Ibrahim et P. L'Ecuyer. Forecasting call center arrivals : Fixed-effects, mixed-effects, and bivariate models. *Manufacturing and Services Operations Management*, 15(1):72–85, 2013.

[81] A. A. Jagers et E. A. van Doorn. Convexity of functions which are generalizations of the Erlang loss function and the Erlang delay function. *SIAM Review*, 33:281–282, 1991.

[82] G. Jongbloed et G. Koole. Managing uncertainty in call centers using Poisson mixtures. *Applied Stochastic Models in Business and Industry*, 17:307–318, 2001.

[83] O. Jouini, G. Koole et A. Roubos. Performance indicators for call centers with impatient customers. *IIE Transactions*, 45(3):341–354, 2013.

[84] C. Kao, W. T. Song et S.-P. Chen. A modified quasi-Newton method for optimization in simulation. *International Transactions in Operational Research*, 4(3):223–233, 1997.

[85] E. G. Keith. Operator scheduling. *AIIE Transactions*, 11(1):37–41, 1979.

[86] J. E. Kelley Jr. The cutting-plane method for solving convex programs. *Journal of the Society for Industrial and Applied Mathematics*, 8(4):703–712, 1960.

[87] P. J. Kolesar, K. L. Rider, T. B. Crabill et W. E. Walker. A queuing-linear programming approach to scheduling police patrol cars. *Operations Research*, 23(6): 1045–1062, 1975.

[88] G. Koole. Redefining the service level in call centers. Rapport technique, Department of Stochastics, Vrije Universiteit, Amsterdam, 2003.

[89] G. Koole, B. F. Nielsen et T. B. Nielsen. Optimization of overflow policies in call centers, 2009. Soumis.

[90] G. Koole, B. F. Nielsen et T. B. Nielsen. First in line waiting times as a tool for analysing queueing systems. *Operations Research*, 60(5):1258–1266, 2012.

[91] G. Koole et A. Pot. Approximate dynamic programming in multi-skill call centers. Dans M. E. Kuhl, N. M. Steiger, F. B. Armstrong et J. A. Joines, éditeurs, *Proceedings of the 2005 Winter Simulation Conference*, pages 576–583. IEEE Press, 2005.

[92] G. Koole, A. Pot et J. Talim. Routing heuristics for multi-skill call centers. Dans *Proceedings of the 2003 Winter Simulation Conference*, pages 1813–1816. IEEE Press, 2003.

[93] G. Koole et J. Talim. Exponential approximation of multi-skill call centers architecture. Dans *Proceedings of QNETs*, pages 23/1–10, 2000.

[94] P. Larrañaga, R. Etxeberria, J. A. Lozano, B. Sierra, I. Inza et J. M. Peña. A Review of the Cooperation between Evolutionary Computation and Probabilistic Graphical Models. Dans A. Ochoa, M. R. Soto et R. Santana, éditeurs, *Proceedings of the Second Symposium on Artificial Intelligence (CIMAF-99)*, pages 314–324, Habana, Cuba, march 1999.

[95] A. M. Law et W. D. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill, New York, NY, troisième édition, 2000.

[96] S. Le Digabel. Algorithm 909 : NOMAD : Nonlinear optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4):1–15, 2011.

[97] P. L'Ecuyer et E. Buist. Variance reduction in the simulation of call centers. Dans *Proceedings of the 2006 Winter Simulation Conference*, pages 604–613. IEEE Press, 2006.

[98] P. L'Ecuyer et E. Buist. Variance reduction in the simulation of call centers. Dans *Proceedings of the 2006 Winter Simulation Conference*, pages 604–613. IEEE Press, 2006.

[99] P. L'Ecuyer, N. Giroux et P. W. Glynn. Stochastic optimization by simulation : Numerical experiments with the $M/M/1$ queue in steady-state. *Management Science*, 40(10):1245–1261, 1994.

[100] S. Liao, C. Van Delft, G. Koole et O. Jouini. Shift-scheduling of call centers with uncertain arrival parameters. Dans *MOSIM 2010*, pages 1–10, Hammamet, Tunisia, 2010. `http://www.enim.fr/mosim2010/program.php`.

[101] A. Mandelbaum, W. Massey et M. I. Reimann. Strong approximations for Markovian service networks. *Queueing Systems*, 30:149–201, 1998.

[102] A. Mandelbaum et A. L. Stolyar. Scheduling flexible servers with convex delay costs : Heavy-traffic optimality of the generalized c-mu rule. *Operations Research*, 52:836–855, 2004.

[103] V. Mehrotra. Ringing up big business. *ORMS Today*, 24(4):18–24, October 1997.

[104] V. Mehrotra, O. Ozlük et R. Saltzman. Intelligent procedures for intra-day updating of call center agent schedules. *Production and Operations Management*, 19(3):353–367, 2010.

[105] J. M. Milner et T. L. Olsen. Service-level agreements in call centers : Perils and prescriptions. *Management Science*, 54(2):238–252, 2008.

[106] H. Mühlenbein et G. Paaß. From recombination of genes to the estimation of distributions 1. binary parameters. Dans Hans-Michael Voigt, Werner Ebeling, Ingo Rechenberg et Hans-Paul Schwefel, éditeurs, *Parallel Problem Solving from Nature*

— *PPSN IV*, volume 1141 de *Lecture Notes in Computer Science*, pages 178–187. Springer Berlin / Heidelberg, 1996.

[107] Nortel Networks. Nortel call center set up and operation guide, Mars 2008. Disponible sur `http://support.avaya.com/css/P8/documents/100096569`. Accédé en septembre 2011.

[108] O. Perry et W. Whitt. Responding to unexpected overloads in large-scale service systems. *Management Science*, 55(8):1353–1367, 2009.

[109] A. Pot, S. Bhulai et G. Koole. A simple staffing method for multi-skill call centers. *Manufacturing and Service Operations Management*, 10:421–428, 2008.

[110] F. F. Reichheld et W. E. Sasser Jr. Zero defections : Quality comes to services. *Harvard Business Review*, pages 105–111, September–October 1990. Reprint 90508.

[111] J. Riordan. *Stochastic Service Systems*. John Wiley & Sons Inc., New York, NY, 1962. The SIAM series in applied mathematics.

[112] T. R. Robbins et T. Harrison. A simulation-based scheduling model for call centers with uncertain arrival rates. Dans *Proceedings of the 2008 Winter Simulation Conference*, pages 2884–2889. IEEE Press, 2008.

[113] T. R. Robbins et T. P. Harrison. A stochastic programming model for scheduling call centers with global service level agreements. *European Journal of Operational Research*, 207(3):1608–1619, 2010. `http://dx.doi.org/10.1016/j.ejor.2010.06.013`.

[114] Rockwell Automation, Inc. Arena simulation, 2005. Voir `http://www.arenasimulation.com`.

[115] S. M. Ross. *Introduction to Probability Models*. Academic Press, dixième édition, 2010. ISBN 9780123756879.

[116] R. Y. Rubinstein et D. P. Kroese. *The Cross-Entropy Method : A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation and Machine Learning*. Springer-Verlag, 2004.

[117] R. Saltzman et V. Mehrotra. Managing trade-offs in call center agent scheduling : methodology and case study. Dans *Proceedings of the 2007 summer computer simulation conference*, pages 643–651. Society for Computer Simulation International, 2007.

[118] H. Shen et J. Z. Huang. Interday forecasting and intraday updating of call center arrivals. *Manufacturing and Service Operations Management*, 10(3):391–410, 2008.

[119] R. A. Shumsky. Approximation and analysis of a call center with flexible and specialized servers. *OR Spectrum*, 26:307–330, 2004.

[120] M. E. Sisselman et W. Whitt. Value-based routing and preference-based routing in customer contact centers. *Production and Operations Management*, 16(3):277–291, 2007.

[121] R. Soyer et M. M. Tarimcilar. Modeling and analysis of call center arrival data : A Bayesian approach. *Management Science*, 54(2):266–278, 2008.

[122] D. A. Stanford et W. K. Grassmann. The bilingual server system : A queueing model featuring fully and partially qualified servers. *INFOR*, 31(4):261–277, 1993.

[123] K. Storbacka, T. Strandvik et C. Grönroos. Managing customer relationships for profit : The dynamics of relationship quality. *International Journal of Service Industry Management*, 5(5):21–38, 1994.

[124] H. M. Taylor et S. Karlin. *An Introduction to Stochastic Modeling*. Academic Press, San Diego, troisième édition, 1998.

[125] J. W. Taylor. A comparison of univariate time series methods for forecasting intraday arrivals at a call center. *Management Science*, 54(2):253–265, 2008.

[126] J. W. Taylor. Density forecasting of intraday call center arrivals using models based on exponential smoothing. *Management Science*, 58(3):534–549, 2012.

[127] T. Tezcan et J. G. Dai. Dynamic control of *n*-systems with many servers : Asymptotic optimality of a static priority policy in heavy traffic. *Operations Research*, 58 (1):94–110, 2010.

[128] G. M. Thompson. Labor staffing and scheduling models for controlling service levels. *Naval Research Logistics*, 8:719–740, 1997.

[129] W. Tych, D. J. Pedregal, P. C. Young et J. Davies. An unobserved component model for multi-rate forecasting of telephone call demand : The design of a forecasting support system. *International Journal of Forecasting*, 18:673–695, 2002.

[130] J. A. van Mieghem. Dynamic scheduling with convex delay costs : the generalized c*mu* rule. *Annals of Applied Probability*, 5:809–833, 1995.

[131] J. A. van Mieghem. Due-date scheduling : Asymptotic optimality of generalized longest queue and generalized largest delay rules. *Operations Research*, 51(1):113–122, 2003.

[132] R. B. Wallace et W. Whitt. A staffing algorithm for call centers with skill-based routing. *Manufacturing and Service Operations Management*, 7(4):276–294, 2005.

[133] J. Weinberg, L. D. Brown et J. R. Stroud. Bayesian forecasting of an inhomogeneous Poisson process with applications to call center data. *Journal of the American Statistical Association*, 102(480):1185–1198, 2007.

[134] W. Whitt. Understanding the efficiency of multi-server service systems. *Management Science*, 38:708–723, 1992.

[135] W. Whitt. Dynamic staffing in a telephone call center aiming to immediately answer all calls. *Operations Research Letters*, 24:205–212, 1999.

[136] W. Whitt. Staffing a call center with uncertain arrival rate and absenteeism. *Production and Operations Management*, 15:88–102, 2006.

[137] R. W. Wolff. Poisson arrivals see time averages. *Operations Research*, 30:223–231, 1982.

[138] R. W. Wolff. *Stochastic Modeling and the Theory of Queues*. Prentice-Hall, New York, NY, 1989.