Université de Montréal

Optimisation stochastique pour l'affectation du personnel polyvalent dans un centre d'appels téléphoniques

par
Wyean Chan

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

Mémoire présenté à la Faculté des études supérieures
en vue de l'obtention du grade de Maître ès sciences (M.Sc.)
en informatique et recherche opérationnelle

Décembre, 2006

Université de Montréal
Faculté des études supérieures

Ce mémoire intitulé:

Optimisation stochastique pour l'affectation du personnel polyvalent dans un centre d'appels téléphoniques

présenté par:

Wyean Chan

a été évalué par un jury composé des personnes suivantes:

Michel Gendreau,	président-rapporteur
Pierre L'Écuyer,	directeur de recherche
Jean-Yves Potvin,	membre du jury

Mémoire accepté le: .....

RÉSUMÉ

Les centres d'appels téléphoniques occupent une place indéniable dans notre société autant pour obtenir du support que faire des achats ou des ventes à distance. Suivant l'évolution de la technologie, l'opération de ces centres s'est complexifiée : plusieurs types d'appels entrants et/ou sortants, la polyvalence des agents à servir plusieurs types d'appels et des politiques de routage par compétence. Avec l'essor de l'ère des services, les opportunités de percée dans ce domaine présentent un intérêt croissant pour l'industrie.

Dans ce mémoire, nous avons développé des méthodes heuristiques d'optimisation stochastique pour minimiser le coût d'affectation des agents pour une période, soit une simplification au problème de conception d'horaires. À la différence des problèmes présentés dans la littérature, nous nous intéressons au cas où un gestionnaire ne peut pas modifier les politiques de routage et doit assurer un seuil minimal de qualité de service pour chaque type d'appel ainsi que pour tout le centre.

La méthode d'optimisation par coupes linéaires et simulations repose sur la génération de sous-gradients calculés à l'aide d'un simulateur. Partant d'une solution irréalisable, des coupes sont ajoutées jusqu'à l'obtention d'une solution réalisable. Les instances du problème linéaire sont résolues grâce à des solutionneurs externes.

La recherche randomisée par voisinage est une méthode de descente par des mouvements aléatoires dans le domaine des solutions réalisables. Vu le nombre élevé d'évaluations des voisins, nous avons conçu un modèle d'approximation analytique de perte et délai basé sur les formules de chaînes de Markov en temps continu.

Selon nos comparaisons empiriques avec différents exemples, la performance de nos algorithmes était compétitive. Nous présentons également des analyses de sensibilité par rapport à différents paramètres de chaque méthode, ainsi que quelques variations de nos algorithmes.

Mots clé : méthodes heuristiques, programmation linéaire, recherche randomisée par voisinage.

ABSTRACT

Call centers occupy an undeniable place in our society, whether for customer services, purchases or sales. Following the technology advances, modern call center operations became more complex : multi-skill agents, inbounds and outbounds calls and skill-based routing policies. With the growing activity in the service sector, research advancement opportunities present great interest for the industry.

In this thesis, we present heuristic stochastic optimisation algorithms to minimize the staffing cost for a single period, a simplified version of the scheduling problem in workforce management. In contrast with the current literature, we suppose the call center manager operates with limited power such that the routing policy is fixed and he must assure a service level threshold for each call type and for the whole center.

The optimization via linear programming and simulation is based on subgradient based cuts, calculated by simulation. Starting with an infeasible solution, cuts are added until feasibility is reached. The linear problems are solved by an external library solver such as Cplex.

The randomized search is a neighborhood search within the feasible domain. Because of the high number of solutions that need to be evaluated, we developed a loss-delay analytic approximation based on continuous time Markov chain formulas.

The performance of our algorithms is competitive when compared through different examples. We present the sensitivities of different parameters for each algorithm and also the results of alternative variations.

Keywords: heuristic methods, linear optimization, randomized neighborhood search.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES

ACD	*Automatic Call Distributor*, gère le routage des appels aux agents
ASA	*Average Speed of Answer*, temps moyen d'attente des appels
AWT	*Acceptable Waiting Time*, durée d'attente des clients jugée acceptable lors du calcul du TSF
CMTC	Chaîne de Markov en Temps Continu, utilisée dans l'analyse des files d'attente
CP	*Cutting-Plane*, algorithme d'optimisation par une méthode de coupes et la simulation
CRTC	Conseil de la Radiodiffusion et des Télécommunications Canadiennes
CSR	*Customer Service Representative*, appelé aussi agent
FCFS	*First Come, First Served*, politique de routage premier arrivé, premier servi
FIFO	*First-in, First-out*, un synonyme de FCFS
IVR	*Interactive Voice Response*, système de traitement automatisé des appels
LD	*Loss-Delay*, approximation perte et délai
PABX	*Private Automatic Branch eXchange*, commutateur connectant les appels téléphoniques au centre d'appels.
RS	*Randomized Search*, algorithme d'optimisation par la recherche randomisée
SIM	Simulation
SL	*Service Level*, le niveau de service
ssi	Si et seulement si
TSF	*Telephone Service Factor*, niveau de service mesuré par le ratio des appels ayant attendus à l'intérieur d'une durée de AWT.
VRU	*Voice Response Unit*, appelé aussi IVR
WFM	*Workforce Management*, gestion de la main-d'oeuvre

LISTE DES NOTATIONS ET DES SYMBOLES

$\mathbf{c}$	Vecteur de taille m représentant le coût d'un agent par groupe
$\mathbf{e}_j$	Vecteur où l'élément j est égal à 1 et 0 pour le reste
$\mathscr{K}_j$	Ensemble des types d'appels pouvant être servis par les agents du groupe j
$g(\mathbf{x})$	Niveau de service global pour une affectation $\mathbf{x}$ d'agents et pour un τ donné
$g_i(\mathbf{x})$	Niveau de service pour les appels du type i pour une affectation $\mathbf{x}$ d'agents et pour un τ donné
l	Seuil minimal requis pour le niveau de service global
l_i	Seuil minimal requis pour le niveau de service pour les appels du type i
$\mathscr{N}$	Ensemble de types d'appels
n	Nombre de types d'appels, $n = \lvert\mathscr{N}\rvert$
$\mathscr{M}$	Ensemble de groupes d'agents
m	Nombre de groupes d'agents, $m = \lvert\mathscr{M}\rvert$
$\mathbf{x}$	Vecteur de taille m représentant le nombre d'agents dans chaque groupe
λ_i	Taux d'arrivée des appels de type i
μ_i	Taux de service des appels de type i, si indépendant des agents
$\mu_{i,j}$	Taux de service des appels de type i par les agents du groupe j
ρ_i	charge du type d'appel i en unité erlang
η_i	Taux de patience des appels du type i
τ_i	Temps d'attente maximal accepté pour les appels du type i, appelé aussi AWT

REMERCIEMENTS

J'aimerais premièrement remercier Pierre L'Écuyer, mon directeur de maîtrise, pour m'avoir soutenu autant académiquement que financièrement tout au long de ma maîtrise. Je le remercie pour son encadrement et aux opportunités offertes qui m'ont permis de mieux connaître le milieu de la recherche académique.

Je voudrais remercier par la suite Athanassios Avramidis pour ses conseils et ses critiques durant tout le développement des algorithmes ainsi que sur mes présentations et rédactions.

Je remercie l'entreprise partenaire Bell Canada, en particulier Naoufel Thabet, et le Conseil de Recherche en Sciences Naturelles et en Génie (CRSNG) pour avoir contribué au financement d'un projet de recherche sur les centres d'appels ainsi qu'à l'octroi d'une bourse d'étude supérieure à incidence industrielle (ESII).

Je remercie Mehmet Tolga Ceẓik, avec qui j'ai travaillé sur le logiciel d'optimisation par programmation linéaire, Eric Buist pour son support sur sa bibliothèque de simulations et Richard Simard pour son support général en tant que responsable du laboratoire.

Finalement, je tiens à remercier le support de ma famille, plus particulièrement mes parents qui travaillent sans relâche.

CHAPITRE 1

INTRODUCTION

Les centres d'appels téléphoniques, une industrie croissante, occupent une place importante dans notre société autant pour fournir du support aux clients que pour être un médium supplémentaire pour augmenter les ventes d'une compagnie. Ce service est rendu indispensable notamment pour les grands organismes tels que les institutions gouvernementales, les institutions financières, les compagnies de transport et les services d'urgence 911. La communication à distance permet aux entreprises de réduire leurs coûts grâce aux économies d'échelle en regroupant les employés dans un établissement et permet également d'offrir des services en dehors des heures d'ouverture régulières des établissements, parfois 24 heures par jour.

Aux États-Unis, selon les statistiques du bureau de recensement américain, l'industrie des centres d'appels, classée dans la catégorie de service de support d'entreprise (l'activité principale étant les services d'appels), était composée de 6 271 centres d'appels avec une valeur de vente de 12.0 $ milliards, employant 292 315 personnes pour un coût salarial annuel de 4.7 $ milliards en 1997 [12]. D'après le recensement de 2002, le nombre de centres d'appels a été réduit à 5 696, mais la valeur de l'industrie a grimpé à 13.4 $ milliards avec un effectif de 413 912 employés et un coût salarial annuel de 7.2 $ milliards [13]. Il est à noter que plus de 50% de ces établissements sont des centres de télémarketing. Sans considérer l'activité économique de l'entreprise, Brigandi et al. [7] estiment qu'en 1994 il y avait aux États-Unis 350 000 entreprises qui employaient 6.5 millions de personnes dont les fonctions étaient reliées aux centres d'appels, comparé à 1 650 entreprises et un demi million en 1980. Depuis les dernières années, l'externalisation des appels (*outsourcing*) a étendu l'industrie à l'échelle mondiale. Plusieurs grandes companies ont déménagé leurs centres d'appels ou ont contracté les services d'un sous-traitant à l'étranger où le coût salarial est moins élevé. À chaque année aux États-Unis, un grand nombre d'emplois de support de service sont transférés vers l'Inde. Ainsi, il arrive fréquemment que les clients soient servis au téléphone par des personnes situées

de l'autre côté de la planète.

1.1 Description d'un centre d'appels

Mehrotra [33] définit un centre d'appels comme étant un groupe dont l'activité économique principale est de communiquer avec les clients ou des clients potentiels. Un centre d'appels est constitué grosso modo d'un ensemble de lignes réseaux, de routeurs, d'employés et d'ordinateurs. Ce centre peut être situé dans un établissement, réparti dans plusieurs établissements ou même être dans une salle virtuelle où les employés sont connectés à partir de leur résidence. Avec l'avancement de la technologie, les centres d'appels évoluent en centres de contacts lorsque d'autres média de communications s'ajoutent, par exemple les courriels et les télécopies. Gans et al. [21] et Koole [29] donnent une bonne description du fonctionnement d'un centre d'appels.

Un *appel* (ou *contact*) représente une communication entre un client et une entreprise. Ces appels sont généralement catégorisés par type, représentant le service demandé et sa provenance d'origine. Les services offerts sont souvent décidés par les cadres supérieurs à l'extérieur du centre d'appels. Chaque type de service requiert une habileté particulière de la part de l'employé. Une habileté peut être la langue, la connaissance technique d'un produit spécifique ou toute autre aptitude acquise lors d'une formation ou non. Évidemment, un client peut être servi par plusieurs employés avant d'être satisfait. Un appel est également distingué selon son émetteur : *entrant* ou *sortant*. Un appel *entrant* est initié par le client souvent pour demander un service ou une information. Un appel *sortant* est initié par un employé généralement pour la vente d'un produit. Une entreprise effectue des appels sortants pour augmenter ses profits et ceci permet aussi de contrer la baisse de la productivité durant les périodes plus calmes. Un centre d'appels ayant ces deux types est appelé un centre *mixte*.

Les employés qui interagissent avec les clients sont appelés des *agents* et souvent des *représentants au service à la clientèle* ou *Customer Service Representatives* (CSRs). Le personnel est appelé polyvalent lorsqu'un agent peut servir plusieurs types d'appels. Ces agents sont affectés à des appels selon leurs habiletés. Ils sont parfois affectés à

des tâches ne demandant qu'à servir des types d'appels spécifiques et ne requérant qu'une partie de leurs habiletés. Les agents partageant un ensemble de tâches communes forment un *groupe d'agents*. Sans compter les coûts de formation, le coût d'un agent est souvent déterminé en fonction du nombre de tâches affectés à son groupe. Un groupe d'agents est appelé *spécialiste* s'il est affecté à peu de tâches et *généraliste* dans le cas de plusieurs tâches.

1.1.1 Traitement des appels sortants

Effectuer des appels sortants, en contactant des clients ou des clients potentiels, est une source de profits additionnels pour une entreprise. Ceci permet d'augmenter la productivité du centre durant les heures plus calmes en convertissant les lignes réseaux non utilisées pour les communications sortantes.

Dans le cas où ce sont les agents qui composent, le traitement est très simple. L'agent pige un numéro de téléphone dans une base de données et effectue l'appel.

Dans un autre cas, les appels sont composés par un équipement appelé le *composeur* ou aussi *dispositif de composition prédictive*. Cet appareil effectue plusieurs appels simultanément à partir d'une base de données de clients ou de l'annuaire téléphonique et le nombre d'appels composés varie en fonction du nombre d'agents libres. La gestion du composeur est contrôlée par une *politique de numérotation*. La probabilité de succès d'un appel (qu'un client réponde) est généralement faible et varie selon l'heure de la journée. Dans le cas d'une connexion réussie, le composeur achemine l'appel à un agent capable de le servir. Lorsque le nombre de succès est plus élevé que le nombre d'agents disponibles, les clients en excès sont mis dans une file d'attente ou sont déconnectés. Ces échecs sont appelés des *mismatchs*. Bien entendu, lorsque les appels sont composés manuellement par les agents, il n'y pas de *mismatch*.

1.1.2 Traitement des appels entrants

Les clients appellent aux centres pour diverses raisons. Pour recevoir ces appels, un centre est connecté au réseau de service téléphonique à l'aide d'un commutateur appelé

Private Automatic Branch eXchange (PABX ou aussi PBX). La capacité maximale de clients connectés simultanément est déterminée par le nombre de lignes téléphoniques que le centre a louées de la compagnie de téléphone. Un client recevra un signal occupé si toutes ces lignes sont déjà occupées. L'appel est ainsi dit *bloqué* et le client est perdu. À partir du PABX, un appel est acheminé vers un système de traitement automatisé ou à un routeur afin de trouver un agent.

Plusieurs centres d'appels utilisent un système de traitement vocal automatisé, nommé *Interactive Voice Response* (IVR) et aussi *Voice Response Unit* (VRU), permettant au client d'interagir à l'aide du clavier de son téléphone. Avec le développement de la technologie, certains systèmes peuvent être commandés directement de la voix du client. Dans certaines industries, la majorité des appels arrivant dans un centre est souvent constituée de quelques types principaux. La plupart des clients d'une banque appellent pour obtenir le solde de leurs comptes, pour faire des transferts de fonds, payer leurs factures, etc. Un IVR couvre généralement les services principaux et offre un libre-service au client. La capacité d'un IVR est "infinie", ainsi il n'y aucun temps d'attente. Dans plusieurs centres d'appels bancaires, 80% des appels se terminent dans l'IVR sans jamais avoir recours à un agent. Ainsi, un IVR permet parfois de réduire considérablement le nombre d'agents requis. Pour un client désirant parler avec un agent, l'IVR sert à l'identification du type d'appel, et à recueillir le dossier du client si disponible, avant de le transférer à un agent via le routeur.

Les appels destinés aux agents sont acheminés et distribués par le *routeur*, appelé aussi un *distributeur automatique d'appels* ou *Automatic Call Distributor* (ACD). En plus du routage, l'ACD gère les files d'attente et recueille diverses statistiques telles que le nombre d'abandons et les temps d'attente moyens des clients. L'ACD opère suivant une *politique de routage* pour associer les appels aux agents. L'ACD est un équipement qui existe depuis longtemps dont la principale politique utilisée autrefois était celle du *premier entré, premier servi* ou *First Come, First Served* (FCFS), appelée aussi *First-in, First-out* (FIFO). Aujourd'hui, les ACDs sont des ordinateurs avec des politiques programmables par les gestionnaires des centres d'appels. Ceci permet d'établir des politiques complexes et dynamiques, notamment pour les centres avec plusieurs types d'ap-

pels et des agents polyvalents où les décisions peuvent dépendre du nombre d'habiletés, des listes de préférences, du taux d'occupation, etc. Un routeur effectue du *routage par compétence* lorsqu'il a besoin de l'information sur la liste d'habiletés de chaque agent. Un client ne pouvant pas être servi immédiatement est placé dans une file d'attente. Si le client est trop impatient, il raccroche et il est considé un *abandon*. Nous pouvons supposer que les routeurs modernes possèdent des files d'attente ayant des capacités "illimitées". Les clients bloqués ou ayant abandonné peuvent également *recomposer*. Les statistiques de recomposition ne sont généralement pas calculées par l'ACD. La figure 1.1 montre les différents processus des appels entrants, omettant le système IVR.

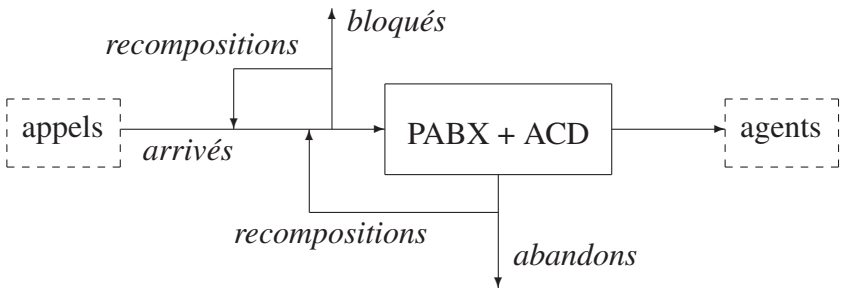

Figure 1.1 – Diagramme des processus des appels entrants, omettant l'IVR.

1.1.3 Qualité de service

La *qualité de service* du centre d'appels est mesurée par ses *niveaux de service* ou *Service Level* (SL). Il existe plusieurs mesures de niveaux de service dépendamment de l'objectif du centre d'appels. Par exemple, le nombre d'appels bloqués à cause d'un manque de lignes téléphoniques libres serait une mesure importante pour les centres d'urgence. Une autre mesure possible est le temps d'attente moyen ou *Average Speed of Answer* (ASA). Cependant, la donnée moyenne peut donner une mauvaise interprétation de la réalité. Supposons que la moitié des appels ait été servie immédiatement et que l'autre moitié ait eu un temps d'attente inacceptable, ASA pourrait donner un niveau de service moyen qui serait jugé acceptable. Nous ne pouvons pas différencier ce cas d'un centre où tous les appels reçoivent une durée d'attente moyenne acceptable. Soient X le

nombre d'appels servis et W_i la durée d'attente de l'appel $i \in \{1,2,\ldots,X\}$. Le SL basé sur le temps moyen d'attente est :

$$\mathrm{SL}^1_{\mathrm{ASA}} = \frac{\sum_{i=1}^{X} W_i}{X}. \tag{1.1}$$

S'il y a présence d'abandons, soient A le nombre d'abandons et U_j la durée d'attente avant d'abandonner l'appel $j \in \{1,2,\ldots,A\}$, le calcul devient :

$$\mathrm{SL}^2_{\mathrm{ASA}} = \frac{\sum_{i=1}^{X} W_i + \sum_{j=1}^{A} U_j}{X+A}. \tag{1.2}$$

Le niveau de service le plus grandement utilisé est parfois appelé *Telephone Service Factor* (TSF). Cette mesure représente la fraction des appels ayant attendu à l'intérieur d'une durée donnée nommée le *temps d'attente acceptable* ou *Acceptable Waiting Time* (AWT). Soit X_b le nombre d'appels servis ayant attendu au plus AWT, $X_b \leq X$, la formule du SL est :

$$\mathrm{SL}^1_{\mathrm{TSF}} = \frac{X_b}{X}. \tag{1.3}$$

Dans le cas où nous considérons les abandons, soient A et A_b le nombre d'appels ayant abandonné et le nombre ayant abandonné avant un temps d'attente de AWT, respectivement, $A \geq A_b$. La mesure devient :

$$\mathrm{SL}^2_{\mathrm{TSF}} = \frac{X_b}{X + (A - A_b)}. \tag{1.4}$$

Il existe d'autres variantes telle que :

$$\mathrm{SL}^3_{\mathrm{TSF}} = \frac{X_b + A_b}{X + A}. \tag{1.5}$$

Les SL peuvent être calculés par période, par types d'appels, agrégés sur plusieurs et sur tous les appels. Dans le chapitre des exemples numériques, le SL est calculé par l'équation (1.4).

L'objectif répandu dans l'industrie est celui du 80/20 qui indique qu'il faut répondre

à 80% des appels en 20 secondes ou moins. Ces objectifs sont imposés par l'entreprise ou par des institutions gouvernementales. Au Canada, par exemple, le Conseil de la Radiodiffusion et des Télécommunications Canadiennes (CRTC) impose la règle du 80/20 aux compagnies de téléphone pour certains types d'appels dont l'accès au bureau d'affaires et l'accès au centre de réparation [17]. Un échec au respect de cette règle peut résulter en une pénalité de plusieurs, voire des centaines de milliers de dollars dépendamment de la gravité.

La mesure du TSF peut suggérer des pratiques douteuses pour améliorer les performances. Au lieu de servir les clients par la politique du premier entré, premier servi, il serait plus avantageux de servir les appels dont le temps d'attente n'a pas dépassé le AWT avant les autres qui n'amélioreront pas le SL. Une solution pour éviter cette pratique est d'imposer une règle limitant le nombre d'abandons. Un abandon signifie possiblement la perte d'un client et de revenu par conséquence. Un gestionnaire pourrait vouloir limiter le *niveau d'abandons* $(A/[X+A])$ à 5% des appels entrants par exemple.

Les SL sont mesurés différemment pour les appels sortants. La performance du centre pourrait être mesurée selon le nombre d'appels sortants réussis et le nombre de *mismatchs*.

1.1.4 Gestion d'un centre d'appels

La gestion d'un centre d'appels est effectuée sur plusieurs niveaux de décisions :

— Les *décisions stratégiques* telles que le budget alloué au centre, les types de service à offrir et l'objectif du centre sont prises par la direction de l'entreprise.

— La gestion du budget, l'achat et l'utilisation des équipements ainsi que l'embauche des employés et de leur formation sont des *décisions tactiques* prises par le directeur du centre. La planification implique également la prévision de la demande future.

— Dans l'étape de la gestion de la main-d'oeuvre, appelée aussi *Workforce Management* (WFM), l'affectation des agents sur un horizon de quelques jours, semaines ou mêmes mois constitue une *décision de planification*.

— Les *décisions de contrôle journalières* sont prises par les chefs d'équipe pour

réagir à différentes situations au courant d'une journée.

— Les *décisions de contrôle en temps réel* sont prises par le routeur pour assigner les appels aux agents.

À chaque niveau de décision, nous pouvons poser la question : « Est-ce que ce choix est optimal ? » Un cadre supérieur pourrait décider entre la création de plusieurs centres d'appels locaux, un grand centre unique ou confier partiellement ou la totalité des appels à une entreprise de sous-traitance. Il doit aussi décider de l'importance du centre d'appels au sein de la compagnie : restreindre à un service de soutien, établir une division de vente à distance, chercher des nouveaux clients par des appels sortants, ... Ces questions demandent des études approfondies sur les besoins de l'entreprise et les décisions prises sont difficiles à changer une fois exécutées.

Le nombre de lignes téléphoniques à louer représente aussi un problème d'optimisation. Les lignes sans frais aux clients (1-800) sont généralement assez coûteuses pour l'entreprise et il est parfois plus profitable que ces lignes soient occupées que libres.

1.1.4.1 Gestion de la main-d'oeuvre

Dans plusieurs centres d'appels, 60% à 70% des dépenses sont consommées par le coût de la main-d'oeuvre [21]. Il est donc crucial d'avoir une bonne gestion du personnel. La gestion peut-être divisée en deux étapes : la *prévision des demandes futures* et la *conception d'horaires*.

Tout d'abord, l'industrie des centres d'appels est un secteur où la rotation du personnel est généralement élevée et les nouveaux agents requièrent un temps de formation avant de pouvoir répondre aux appels. La planification et l'embauche des agents sont des décisions prises plusieurs semaines ou mois à l'avance. La prévision du nombre d'appels est donc une étape importante et un problème difficile qui dépend de plusieurs facteurs. Le lancement d'une nouvelle campagne de publicité, l'arrivée de la période des Fêtes et la vente d'un nouveau produit sont des causes qui font fluctuer le nombre d'appels. Il est donc important d'avoir une bonne communication entre les départements des ventes et de publicité et le centre d'appels.

À partir de ces prévisions, il faut déterminer le nombre d'agents nécessaires et conce-

voir les horaires. Dans un centre à plusieurs types d'appels, il faut également déterminer les différents groupes d'agents. La taille du problème grandit exponentiellement avec le nombre de types d'appels : pour n types d'appels, il existe $2^n - 1$ groupes possibles. La présence des généralistes est nécessaire pour atténuer les effets causés par les arrivées aléatoires. Si les temps d'inter-arrivée et les durées de service étaient constants, les généralistes seraient probablement moins importants. Heureusement, les centres d'appels profitent généralement d'économies d'échelle (*economy of scale*) et il est souvent préférable d'avoir plus de spécialistes dans un grand centre et un plus faible ratio de généralistes, en plus d'une réduction des effectifs.

La gestion de la main-d'oeuvre ne s'effectue pas seulement sur un horizon à moyen terme de quelques semaines ou à long terme de quelques mois. Il n'est pas rare que le nombre d'agents présents au travail soit inférieur à celui planifié à l'horaire pour des raisons telles que les absences ou les congés de maladie. Une autre raison pour un ajustement des horaires est celle de la variance sur la quantité d'appels entre les journées qui est parfois élevée dans certains cas. Avramidis et al. [5] ont observé l'existence d'une corrélation sur le nombre d'appels d'une période à l'autre dans une journée et proposent quelques modèles de prévisions. Si le nombre d'appels était plus élevé que la moyenne durant l'avant-midi, cette hausse risque fort d'être présente durant l'après-midi. Le gestionnaire du centre appels devrait augmenter l'effectif de son personnel en rappellant les agents disponibles sur appel ou en annulant des périodes de formation. Dans le cas où l'achalandage est plus faible que prévu, le gestionnaire peut réduire le coût de son personnel de travail en offrant des congés non payés ou en envoyant en formation certains agents. Évidemment, un coût supplémentaire est généralement relié à ces corrections.

L'objectif du gestionnaire d'un centre d'appels uniquement entrants est de prévoir une affectation d'agents au coût minimal qui satisfait des critères sur la qualtié du service, établis par une loi ou par les dirigeants de l'entreprise. Le critère le plus utilisé est celui du TSF avec la règle de 80% des appels servis en 20 secondes ou moins. Lors de la conception des horaires, il y a d'autres contraintes à respecter qui sont parfois issues d'une convention collective avec les employés. Une journée est divisée en plusieurs périodes. Il faut réserver un temps pour dîner et quelques périodes de pause. L'entreprise

peut exiger de son côté qu'il ne puisse pas y avoir deux pauses consécutives en moins de 2 heures de travail par exemple. Ce mémoire se concentrera sur ce type de centres.

Dans un centre d'appels mixtes, il peut y avoir plusieurs objectifs : minimiser le coût salarial des agents tout en satisfaisant aux contraintes précédentes et maximiser le nombre d'appels sortants. Les appels sortants sont souvent des appels de télémarketing. Certains pays imposent des restrictions sur les heures durant la journée où ces appels sortants sont permis. Au Canada, le CRTC impose aux centres utilisant un dispositif de composition prédictive un ratio d'abandon maximal de 5% par mois [18]. Effectuer des appels sortants est généralement une tâche qui déplaît aux agents. Ainsi, il est parfois inscrit dans la convention collective le maximum d'heures de travail par jour accordées aux appels sortants par agent.

Avec l'accroissement des centres de sous-traitance, une entreprise pourrait également décider d'externaliser une partie des appels en signant un contrat pour 150 000 appels par mois, par exemple. Le processus d'arrivée des appels étant aléatoire, il y a le risque d'avoir une journée très achalandée, ce qui résulterait en un mauvais service sur l'ensemble des appels. L'entreprise voudrait alors externaliser les appels excédentaires durant ces journées plus actives. Cependant, ce sont généralement les centres de sous-traitance qui imposent des restrictions dans les contrats pour éviter ces risques. Supposons qu'un centre reçoit entre 10 000 et 15 000 appels par jour. Si 5 000 appels sont externalisés, le gestionnaire devra alors gérer un centre avec un taux d'arrivée variant entre 5 000 et 10 000 appels par jour, ainsi une variance plus considérable.

1.1.4.2 Efficacité du personnel

Pour évaluer l'efficacité du personnel, les gestionnaires vérifient généralement leur *taux d'occupation*. Agrégée sur tous les agents, soient D la durée de travail total, D_s la durée passée à servir un client et D_l la durée de temps inoccupé en attente d'un nouvel appel. Le taux d'occupation est calculé suivant la formule :

$$O = \frac{D_s}{D_s + D_l}. \tag{1.6}$$

À cause du temps de post-traitement des appels, $D \geq D_s + D_l$. Une augmentation de la productivité signifie en principe une réduction des coûts. Ainsi, un gestionnaire préfère maximiser l'utilisation de ses ressources tout en fournissant une bonne qualité de service. Connaissant la formule (1.6), un agent pourrait imaginer augmenter sa production en allongeant le temps des post-traitements après chaque appel avant qu'il soit prêt à recevoir un prochain appel. Un moyen pour éviter cela est de mesurer également le *facteur d'efficacité des agents* calculé par :

$$E = \frac{D_s + D_l}{D}. \tag{1.7}$$

1.2 État de l'art

Dans ce mémoire, nous nous concentrons sur le problème d'affectation des agents pour un centre d'appels exclusivement entrants. Cette section présente différents outils et la littérature pouvant aider à résoudre le problème d'affectation.

1.2.1 Modélisation d'un centre d'appels

Pour résoudre le problème de gestion d'un centre d'appels, il faut avoir accès aux mesures de performance de différentes solutions. Or, un centre d'appels est un système difficile à modéliser dont les données statistiques sont parfois non disponibles ou bien souvent agrégées par période, e.g., demi-heure, dans l'ACD.

Le *processus d'arrivée* des appels n'est pas un processus Poisson homogène (taux déterministe), bien qu'on fasse souvent cette hypothèse dans la pratique par souci de simplicité mathématique. Les études récentes suggèrent plutôt un processus doublement stochastique, e.g., Poisson-Gamma, où le *taux d'arrivée* des appels est une variable aléatoire par période [5, 8, 27]. Les taux d'arrivée varient souvent selon l'heure ainsi que la journée de la semaine. Une corrélation positive entre les périodes et entre les journées a aussi été observée dans plusieurs analyses.

La *durée de service* d'un appel est souvent considérée comme étant une variable aléatoire suivant une distribution exponentielle. Brown et al. [8] suggèrent que la distribution

soit plutôt lognormale.

Plusieurs autres processus sont moins étudiés ; ce sont souvent des données partiellement disponibles ou conditionnelles à certains événements. Le *temps de patience* détermine le temps qu'un client est prêt à attendre avant d'abandonner. Cependant, l'ACD ne peut enregistrer le temps de patience que pour les clients ayant effectivement abandonné. Pour les autres clients, nous savons seulement que leur temps de patience était supérieur à leur temps d'attente. Il a été observé qu'une fraction des clients raccrochent immédiatement lorsqu'ils doivent entrer dans une file d'attente. Cette statistique recueillie par l'ACD concerne seulement les clients qui n'ont pas pu être servis immédiatement. La probabilité de recomposition après un abandon fait aussi partie des statistiques partiellement observables.

Les outils disponibles pour approximer les performances d'un centre d'appels sont les outils de simulation qui sont précis, mais lents, et les approximations basées sur des modèles analytiques qui sont rapides, mais imprécises pour les centres d'appels modernes. Bien entendu, il est important d'avoir un outil à la fois précis et rapide, en particulier pour les logiciels d'optimisation.

1.2.1.1 Outils de simulations

La simulation est un outil très flexible qui demande généralement une connaissance moins avancée des mathématiques que les modèles analytiques. L'écart grandissant entre l'évolution des centres d'appels modernes et le développement des modèles analytiques est une des raisons principales de la popularité ainsi que de la nécessité des outils de simulation. Cependant, la simulation est un programme complexe qui exige un effort de développement considérable. Basée sur la collecte de statistiques, la simulation requiert des temps d'exécution importants pour réduire le bruit et l'intervalle de confiance des mesures.

Il existe des simulateurs commerciaux tels que *Arena Contact Centers* [36] et *cc-Prophet* [34] munis d'une interface utilisateur conviviale, d'une visualisation graphique de la simulation, d'un générateur de rapports automatisé, etc. Par contre, leurs outils de modélisation sont souvent limités par rapport aux besoins de centres d'appels modernes

et les temps d'exécution sont très lents.

Pour les logiciels d'optimisation étudiés dans ce mémoire, nous utilisons un simulateur sans interface utilisateur construit à partir d'une bibliothèque Java pour la simulation de centres de contacts [11]. La flexibilité, la facilité d'interaction avec d'autres programmes en Java et le temps d'exécution environ 25 fois plus rapide que le logiciel Arena [9] étaient des facteurs importants dans notre choix.

1.2.1.2 Approximations basées sur des modèles analytiques

Au début du vingtième siècle, un mathématicien danois appelé Agner Krarup Erlang fut un pionnier dans la théorie des files d'attente. Travaillant dans l'entreprise *Copenhagen Telephone Company*, il modélisa un simple centre d'appels composé d'un type d'appel avec un taux d'arrivée Poisson, une distribution exponentielle pour la durée de service, un nombre s d'agents et aucun abandon. Ce modèle est appelé $M/M/s$. Il formula les équations, souvent appelées Erlang B et C, pour calculer la probabilité qu'un appel entrant ne trouve pas immédiatement un agent libre dans les cas sans et avec une file d'attente. La formule d'Erlang B suppose l'absence de file d'attente et donne la *probabilité de blocage* d'un appel entrant. En présence d'une file d'attente, la politique de routage correspondante est celle du premier arrivé, premier servi et la formule d'Erlang C donne la *probabilité de délai*, $\mathbb{P}[W > 0]$. Cette formule a été étendue par après pour calculer $\mathbb{P}[W > t]$ pour une durée $t \geq 0$.

Il existe aujourd'hui de nombreuses études dans la littérature sur les centres avec un seul type d'appel et qui traitent des processus d'arrivée et de distributions des durées de service différents ainsi que la possibilité d'abandon. Tous ces modèles se basent sur les chaînes de Markov en temps continu, *Continuous Time Markov Chains* (CTMCs). Ces modèles souffrent de la malédiction de la dimensionalité où le nombre d'opérations augmente exponentiellement avec le nombre de types d'appels et de groupes d'agents. Cette analyse n'est donc pas applicable dans la pratique pour les centres à plusieurs types d'appels avec des agents polyvalents.

La recherche s'est donc plutôt dirigée vers les modèles d'approximation. Jusqu'à présent, la majorité des travaux était concentrée sur les systèmes avec perte, où un centre

ne possède pas de file d'attente, à cause de leur simplicité comparée aux systèmes avec délai. Koole et Talim [31] donnent une approximation à l'aide d'une décomposition exponentielle des taux d'arrivée. Cette approximation a été améliorée en remplaçant celle-ci par une décomposition hyper-exponentielle [20]. Chevalier et al. [15] présentent une approximation basée sur la méthode de Hayward-Fredericks qui est une extension de la méthode de randomisation équivalente ou *Equivalent Random Method* (ERM). Avramidis et al. [4] présentent une extension au modèle de Koole et Talim [31] en ajoutant l'approximation des probabilités de délai.

Bien qu'il soit possible d'obtenir de bonnes approximations des probabilités de blocage, il demeure difficile d'approximer les probabilités de délai, malgré la présence d'une corrélation entre ces deux mesures. Toutefois, les approximations peuvent être de bons guides dans les algorithmes d'optimisation.

1.2.2 Algorithmes d'affectation des agents

Il existe encore peu d'articles portant sur l'affectation des agents dans un centre d'appels moderne.

Atlason et al. [3] présentent un algorithme d'optimisation pour un type d'appel et un groupe d'agents sur plusieurs périodes. L'optimisation cherche à déterminer le nombre d'agents pour chaque quart de travail. L'algorithme est sous la forme d'un problème linéaire qui utilise une méthode de coupes pour éliminer les solutions irréalisables grâce à des sous-gradients générés par simulation, ce qui est une adaptation de la méthodologie de Kelley [28].

Bhulai et al. [6] présentent un algorithme d'optimisation d'horaires pour plusieurs types d'appels et groupes d'agents, mais contraint uniquement par le niveau de service global. L'algorithme est divisé en deux étapes. Premièrement, il détermine le nombre d'agents nécessaires dans chaque groupe pour chaque période. Cette étape résout le problème du dual lagrangien en relaxant la contrainte du SL global puis en trouvant le multiplicateur optimal à l'aide d'une recherche de bisection accompagnée d'une recherche locale. La deuxième étape résout un problème linéaire pour déterminer le nombre d'agents dans chaque groupe et chaque quart de travail.

Wallace et Whitt [39] supposent une liberté sur l'affectation des habiletés aux agents et sur la politique de routage. Dans leurs exemples, ils ont trouvé que des solutions avec au maximum 2 habiletés par agent pouvaient être toutes aussi performantes que celles où les agents possèdent toutes les habiletés. Attribuant le même nombre d'habiletés par agent (donc un coût identique), leur algorithme n'est pas adapté pour optimiser en fonction du coût des agents. De plus, un gestionnaire n'a pas toujours la possibilité de modifier les habiletés d'un groupe ainsi que le modèle de routage à cause des règles administratives, de l'inflexibilité ou la désuétude des équipements, ou le manque de temps pour la formation des groupes d'agents non disponibles, par exemple.

Harrison et Zeevi [22] présentent une optimisation basée sur un modèle par approximation fluide. Les auteurs veulent à minimiser le coût total avec des coûts de pénalité pour les abandons dans un centre d'appels avec un processus d'arrivée doublement stochastique. Le centre n'est soumis à aucune contrainte dure telle que le seuil minimal sur le niveau de service. La performance du centre est traduite par un coût à l'intérieur de la fonction objectif. Il est toutefois difficile de convertir un problème ayant des contraintes dures et d'y associer les coûts de pénalité appropriés.

Ceẓik et L'Ecuyer [14] proposent une adaptation et une extension à Atlason et al. [3] pour les centres ayant plusieurs types d'appels et groupes d'agents en gardant la politique de routage fixe. Cette méthode peut tenir compte des contraintes sur le SL global ainsi que pour chacun des types d'appels. À cause de la complexité qui découlerait de l'ajout des quarts de travail et des périodes multiples, l'algorithme présenté ne résout que sur une période, ou sur des quarts de travail d'une durée d'une journée complète.

Avramidis et al. [4] présentent une méthode de recherche randomisée par voisinage pour résoudre un problème identique à Ceẓik et L'Ecuyer [14]. Le nombre de solutions évaluées étant particulièrement élevé, l'algorithme utilise principalement une approximation analytique à la place de la simulation.

1.3 Projet de maîtrise

La gestion de la main-d'oeuvre pour les centres d'appels modernes est un problème d'optimisation difficile à résoudre. Les études dans ce domaine sont encore récentes et peu d'algorithmes ont été proposés à ce jour.

Dans le cadre de ce mémoire, je me suis intéressé à la minimisation du coût du personnel polyvalent dans un centre composé de plusieurs types d'appels et groupes d'agents pour une seule période, vu la complexité du problème dans le cas de plusieurs périodes. Le but était de concevoir des logiciels d'optimisation pour trouver l'allocation optimale tout en satisfaisant des contraintes sur les niveaux de service, avec des temps d'exécution relativement courts pour être efficace dans l'industrie.

Au cours de ma maîtrise, j'ai implémenté, expérimenté et analysé deux logiciels d'optimisation pour l'affectation du personnel dans les centres d'appels. Le premier logiciel est une optimisation par la programmation linéaire et la simulation. Le deuxième est une optimisation par une recherche aléatoire par voisinage aidée par un modèle d'approximation ainsi que par la simulation. Ce modèle d'approximation pour les centres d'appels pour plusieurs types et groupes d'agents polyvalents est aussi implémenté dans le cadre de ce mémoire.

1.4 Structure du mémoire

Ce mémoire est divisé en quatre parties. Dans le prochain chapitre, nous présentons le modèle mathématique du problème de conception d'horaires ainsi que la version du problème simplifié à une seule période. Dans le chapitre 3, nous présentons l'algorithme d'optimisation par la programmation linéaire et la simulation [14]. Nous discutons principalement de la difficulté de cet algorithme et de différentes méthodes heuristiques implémentées pour l'améliorer. Dans le chapitre 4, nous présentons l'algorithme de recherche randomisée par voisinage et le modèle d'approximation basé sur les chaînes de Markov en temps continu [4]. Dans le chapitre 5, nous analysons et comparons la performance de ces deux algorithmes sur trois exemples de différentes tailles, un exemple fictif et deux basés sur des centres d'appels réels. Nous incluons également des expériences

sur la sensibilité de chaque algorithme. Ce mémoire est accompagné, en annexe, par le guide d'utilisateur pour les logiciels Java implémentés.

1.5 Contribution personnelle

Ma contribution repose principalement sur l'implémentation et l'amélioration des logiciels d'optimisation.

Au chapitre 3, j'ai implémenté l'algorithme d'optimisation par coupes linéaires et la simulation, et j'ai expérimenté les diverses heuristiques auxiliaires aux coupes par sous-gradients, présentées dans les sections 3.3 et 3.4. Les performances de mon programme sont également rapportées dans la section des expériences numériques de Ceşik et L'Ecuyer [14]. J'ai étendu l'algorithme par la suite pour le cas où les durées de service dépendent à la fois du type d'appel et du groupe d'agents (section 3.2.2).

Au chapitre 4, j'ai implémenté l'approximation perte et délai et la recherche aléatoire par voisinage. J'ai également travaillé sur l'accélération de l'algorithme d'approximation (à la section 4.3.5 en faisant appel à la méthode de Gauss-Seidel et à l'aide de l'ordonnancement de l'évaluation des groupes d'agents à la section 4.3.6). J'ai participé au développement et à l'amélioration de la recherche randomisée en analysant diverses méthodes et paramètres, ainsi qu'en essayant différentes combinaisons de mouvements de recherche.

Finalement, au chapitre 5, j'ai analysé et comparé ces deux algorithmes d'optimisation avec différents problèmes inspirés des centres d'appels réels (ils ne sont pas tous rapportés dans ce mémoire), pour déterminer leurs points forts, leurs faiblesses et leur efficacité en pratique, où le temps d'exécution et la qualité des solutions sont des critères importants.

CHAPITRE 2

MODÉLISATION DU PROBLÈME

Une revue générale portant sur la description et la gestion d'un centre d'appels a été présentée dans le chapitre 1. Ce chapitre présente le problème principal de ce mémoire sous forme mathématique. Ce problème est celui de la gestion de la main-d'oeuvre pour un nombre de groupes donnés supposant une infinité d'agents disponibles par groupe. Dans notre problématique, le routage est une donnée invariable.

Notre modèle d'un centre d'appels est composé d'un ensemble de groupes d'agents $\mathscr{M} = \{1,2,\ldots,m\}$ et un ensemble de types d'appels $\mathscr{N} = \{1,2,\ldots,n\}$. Les types d'appels pouvant être servis par les agents du groupe j sont définis par l'ensemble $\mathscr{K}_j \subseteq \mathscr{N}$. Un agent du groupe $j \in \mathscr{M}$ est assigné à un horaire q parmi un ensemble d'horaires possibles $\mathscr{Q}_j$ et le coût de cet agent est $c_{j,q}$. Bien que les $\mathscr{Q}_j$ puissent être différents, les contraintes sur les quarts de travail sont généralement indépendantes du groupe d'agents.

Soit $\mathbf{Q}$ une matrice diagonale :

$$\begin{pmatrix} Q_1 & 0 & \ldots & 0 \\ 0 & Q_2 & & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \ldots & Q_m \end{pmatrix}$$

où la matrice binaire Q_j de dimension $P \times |\mathscr{Q}_j|$, s'il y a P périodes, correspond à la couverture des périodes de chaque horaire dans $\mathscr{Q}_j$. La durée d'une période varie selon les centres d'appels, 15 minutes par exemple. L'élément $Q_j(p,k) = 1$ si l'horaire k du groupe j couvre la période p, 0 sinon. Soit $\mathbf{y} = (y_{1,1},\ldots,y_{1,|\mathscr{Q}_1|},\ldots,y_{m,1},\ldots,y_{m,|\mathscr{Q}_m|})^{\mathrm{t}}$ le vecteur de décision correspondant au nombre d'agents de chaque groupe par horaire de travail. Par le produit $\mathbf{Qy} = \mathbf{x}$, nous obtenons le vecteur $\mathbf{x} = (x_{1,1},\ldots,x_{1,P},\ldots,x_{m,P},\ldots,x_{m,P})^{\mathrm{t}}$ correspondant au nombre d'agents par groupe par période. Bien entendu, $\mathbf{x}$ et $\mathbf{y}$ sont des vecteurs à valeurs entières.

Pour un type d'appel i, nous supposons un taux d'arrivée λ_i et une durée moyenne de service $1/\mu_{i,j}$ lorsque servi par un agent du groupe j. Les processus d'arrivée des appels sont présumés Poisson et les durées de service, des variables exponentielles. Si les taux de service sont indépendants des groupes d'agents (μ_i), la charge est de $\rho_i = \lambda_i/\mu_i$ erlangs. Supposant qu'il y ait des abandons, le temps de patience d'un client de type i est une variable aléatoire exponentielle de moyenne $1/\eta_i$. Nous supposons également que chaque appel ne requiert qu'un seul type de service, ce qui simplifie le modèle de routage.

2.1 Modèle de routage

La configuration et la politique de routage sont des éléments importants qui affectent la performance d'un centre d'appels. Les modèles de routage souvent utilisés sont les routages par compétence établis à partir de listes de priorités. Il peut y avoir une liste de priorité pour chaque type d'appel sur le choix des groupes d'agents (à l'arrivée d'un appel) et une liste pour chaque groupe d'agents sur le choix des types d'appels (à la fin d'un appel par un agent).

La figure 2.1 présente un exemple simple de routage d'un centre d'appels ayant 5 types d'appels (cercles) et 5 groupes d'agents (rectangles) démontrant l'ordre de visite des appels, à leur arrivée, vers différents groupes. Les agents du groupe 1 sont des purs spécialistes et ceux du groupe 5, des purs généralistes. Les appels du type 1 sont acheminés premièrement aux agents du groupe 1, puis du groupe 3 s'il n'y a aucun agent libre dans le groupe 1, et ainsi de suite jusqu'au groupe 5. Lorsqu'il n'y a aucun agent libre pouvant servir l'appel, cet appel entre dans une file d'attente. Si cet appel n'abandonne pas, un agent libéré ultérieurement le sortira de la file et le servira.

Le modèle de routage constitue une partie importante de l'élaboration de l'algorithme d'approximation. Dans l'exemple de la figure 2.1, nous pouvons approximer directement le flot des appels vers chaque groupe puisqu'il n'y a aucun croisement de routage. Nous utilisons le terme de *routage croisé* (*cross-routing*) lorsque les flots des appels de deux groupes sont interdépendants. La figure 2.2 montre un exemple d'un tel

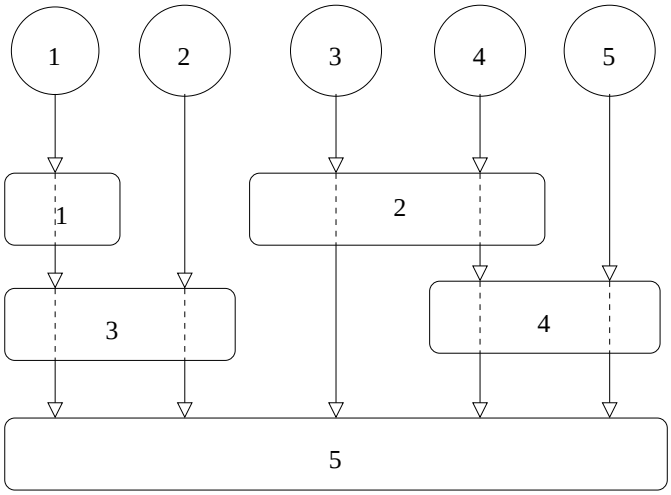

Figure 2.1 – Exemple de routage

cas. Le nombre total d'appels arrivant au groupe 1 dépend du travail du groupe 2 qui dépend lui-même du groupe 1.

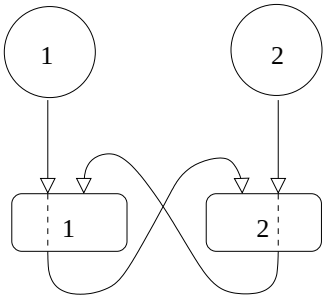

Figure 2.2 – Exemple d'un routage croisé

2.2 Modèle d'optimisation du problème d'horaires

Dans le chapitre 1, nous avons présenté quelques contraintes possibles. Dans notre problématique, nous ne considérons que les contraintes sur le niveau de service de chaque type d'appel ainsi que sur le service global, agrégeant tous les appels. Ces contraintes sont parfois imposées par les gouvernements ou par l'entreprise même. Elles sont des règles définies par les paramètres "$l_{i,p}/\tau_{i,p}$", c'est-à-dire $(100 \times l_{i,p})\%$ des appels

du type i doivent être répondus dans un délai de $\tau_{i,p}$ durant la période p. Ces contraintes peuvent être agrégées : sur toutes les périodes et tous les appels "l/τ", sur toutes les périodes, mais par type d'appel "$l_{i,\bullet}/\tau_{i,\bullet}$" et sur tous les appels, mais par période "$l_{\bullet,p}/\tau_{\bullet,p}$". Puisque nous résolvons ce problème de manière stochastique, les SL et toutes autres mesures de performance sont des espérances mathématiques par rapport à un temps d'horizon infini. Par exemple, la formule 1.4 (page 6) du SL global devient, où g est la fonction d'espérance du niveau de service :

$$g(\mathbf{x}) = \frac{\mathbb{E}[X_b]}{\mathbb{E}[X + (A - A_b)]}. \tag{2.1}$$

Le modèle mathématique pour l'optimisation sur P périodes est le suivant, (tiré de Ceşik et L'Ecuyer [14]) :

$$\begin{aligned}
\min \quad & \sum_{j=1}^{m} \sum_{q=1}^{|\mathscr{Q}_j|} c_{j,q} y_{j,q} \\
\text{sujet à :} \quad & \\
& \mathbf{Qy} = \mathbf{x} \\
& g_{i,p}(\mathbf{x}) \geq l_{i,p} \qquad \forall i \in \mathscr{N}, p = 1, \ldots, P \\
& g_{\bullet,p}(\mathbf{x}) \geq l_{\bullet,p} \qquad p = 1, \ldots, P \\
& g_{i,\bullet}(\mathbf{x}) \geq l_{i,\bullet} \qquad \forall i \in \mathscr{N} \\
& g(\mathbf{x}) \geq l \\
& \mathbf{y} \geq 0 \text{ et entier}
\end{aligned} \tag{P1}$$

Remarquons que ce modèle ne peut pas être résolu directement par la programmation linéaire à cause des fonctions g. Dans les algorithmes d'optimisation, ces fonctions sont remplacées par des approximations obtenues par la simulation ou des analyses numériques. Dans la formulation (P1), nous supposons que g prend en paramètre le nombre d'agents par groupe et par période. Un évaluateur sophistiqué, tel que le simulateur de Buist et L'Ecuyer [11], peut prendre directement en entrée le nombre d'agents selon les

quarts de travail et ainsi éviter la contrainte $\mathbf{Qy} = \mathbf{x}$ et éliminer le vecteur $\mathbf{x}$ puisque les $g(\mathbf{x})$ sont remplacées par les $g(\mathbf{y})$.

2.3 Modèle d'optimisation pour l'affectation d'une seule période

Étant donnée la complexité des problèmes d'horaires, nous nous concentrons dans ce mémoire sur l'optimisation pour une seule période. Nous éliminons ainsi les contraintes sur les quarts de travail $\mathbf{Qy} = \mathbf{x}$ et réduisons la taille du problème (P1). Le modèle devient :

$$
\begin{aligned}
Z = \min \quad & \sum_{j=1}^{m} c_j x_j \\
\text{sujet à :} \quad & \\
& g(\mathbf{x}) \geq l \\
& g_i(\mathbf{x}) \geq l_i \qquad \forall i \in \mathcal{N} \\
& \mathbf{x} \geq 0 \text{ et entier}
\end{aligned}
$$

(P2).

CHAPITRE 3

OPTIMISATION PAR LA PROGRAMMATION LINÉAIRE ET LA SIMULATION

L'algorithme d'optimisation par la programmation linéaire et la simulation est décrit dans Ceşik et L'Ecuyer [14] et représente une adaptation de la méthode de Atlason et al. [3] pour plusieurs types d'appels et groupes d'agents, mais sur une seule période. Puisque les contraintes sur les niveaux de service sont non linéaires, (P2) ne peut être résolu directement par la programmation linéaire. L'algorithme repose sur l'idée que la fonction du niveau de service $g(\mathbf{x})$, pour un centre avec un type d'appel et $\lambda < x\mu$, possède une forme concave lorsque la fonction se rapproche de 1 [26]. Les contraintes non linéaires sont alors remplacées par une série de coupes linéaires générées à partir de sous-gradients de $g(\mathbf{x})$ afin d'éliminer les solutions irréalisables, une méthodologie décrite par Kelley [28]. Cependant, pour les centres ayant plusieurs types d'appels et des abandons, la concavité n'est pas assurée. La méthode de coupes par sous-gradients pour les centres avec plusieurs types d'appels est donc une heuristique.

Définition 3.1. Une fonction $g(x)$ est concave sur le domaine $[A,B]$ si pour les tous points $A \leq a,b \leq B$, l'inégalité suivante est vérifiée :

$$g(\delta a + (1-\delta)b) \geq \delta g(a) + (1-\delta)g(b), \quad 0 \leq \delta \leq 1.$$

Définition 3.2. Une fonction g est convexe si la fonction $(-g)$ est concave.

Le problème original (P2) (page 22) est remplacé par le problème linéaire (P3) suivant (tiré de Ceşik et L'Ecuyer [14]) :

$$
\begin{aligned}
\bar{Z} = \min \quad & \sum_{j=1}^{m} c_j x_j \\
\text{sujet à :} \quad & \\
& \mathbf{B}\mathbf{x} \geq \mathbf{b} \\
& \mathbf{B}_i\mathbf{x} \geq \mathbf{b}_i \qquad \forall i \in \mathcal{N} \\
& \mathbf{x} \geq 0 \text{ et entier}
\end{aligned}
$$

(P3).

Les contraintes sur les SL sont remplacées par les contraintes linéaires $\mathbf{B}\mathbf{x} \geq \mathbf{b}$ pour le service global et $\mathbf{B}_i\mathbf{x} \geq \mathbf{b}_i$ pour le service par type d'appel. Un vecteur $\mathbf{b}_i$ peut être nul lorsqu'il n'y a pas de minimum sur le SL du type d'appel i.

L'algorithme commence à partir d'un point irréalisable et génère des coupes linéaires à chaque itération jusqu'à l'obtention d'une solution réalisable. Le problème à l'itération v est alors (P3)$^{(v)}$ avec un nouvel ensemble de contraintes : $\mathbf{B}^{(v)}\mathbf{x} \geq \mathbf{b}^{(v)}$ et $\mathbf{B}_i^{(v)}\mathbf{x} \geq \mathbf{b}_i^{(v)}$. Ces coupes sont générées principalement grâce aux évaluations des sous-gradients par la fonction g. Étant donnée la sensibilité aux erreurs d'estimation et à l'inexistence d'approximation analytique précise, nous utilisons la simulation pour estimer les fonctions g. La génération des coupes à partir de g et une méthode pour trouver un bon point d'initialisation sont décrites dans les prochaines sections.

3.1 Génération de coupes par sous-gradients

Supposons que la fonction $g(\mathbf{x})$ soit concave sur le domaine $[\mathbf{x}', \infty[$ et que $\mathbf{q}(\mathbf{x}')$ soit un sous-gradient sous forme d'un vecteur au point $\mathbf{x}'$. Puisque g est seulement définie sur $\mathbf{x} \geq 0$ et entier, il peut exister plusieurs sous-gradients pour un même point. Nous générons une coupe lorsque ce point constitue une solution irréalisable, soit $g(\mathbf{x}') < l$. Nous décrivons la procédure pour la fonction g, mais elle est aussi valable pour g_i. Par l'hypothèse de concavité, nous avons l'inégalité valide suivante :

$$
g(\mathbf{x}) \leq g(\mathbf{x}') + \mathbf{q}(\mathbf{x}')^{\mathrm{t}}(\mathbf{x} - \mathbf{x}'), \quad \mathbf{x} \geq \mathbf{x}'. \tag{3.1}
$$

Nous cherchons **x** tel que :

$$l \leq g(\mathbf{x}) \leq g(\mathbf{x}') + \mathbf{q}(\mathbf{x}')^{\mathrm{t}}(\mathbf{x} - \mathbf{x}').$$

D'où l'inégalité linéaire,

$$\mathbf{q}(\mathbf{x}')^{\mathrm{t}}\mathbf{x} \geq l - g(\mathbf{x}') + \mathbf{q}(\mathbf{x}')^{\mathrm{t}}\mathbf{x}' \tag{3.2}$$

puisque les valeurs de $l, g(\mathbf{x}'), \mathbf{q}(\mathbf{x}')^{\mathrm{t}}$ et $\mathbf{x}'$ sont connues.

La figure 3.1 montre graphiquement la génération d'une coupe par sous-gradient dans un cas à une dimension (un groupe d'agents) avec une solution optimale $\lceil x^* \rceil$. Les points sont reliés afin de mieux visualiser la fonction g. Si x' est le point à l'itération courante, l'addition d'une nouvelle coupe éliminera toutes les valeurs de $x < x''$. Notons que si nous générons une coupe par le sous-gradient dans la partie convexe, par exemple au point $\tilde{x}$ au bas de la courbe, ceci éliminerait tous les points $x < \tilde{x}'$ et par conséquent la solution optimale. Les régions convexes sont souvent présentes lorsque g est faible. Nous avons développé des méthodes de coupes auxiliaires, présentées dans les prochaines sections, lorsque la solution courante se retrouve dans de telles régions.

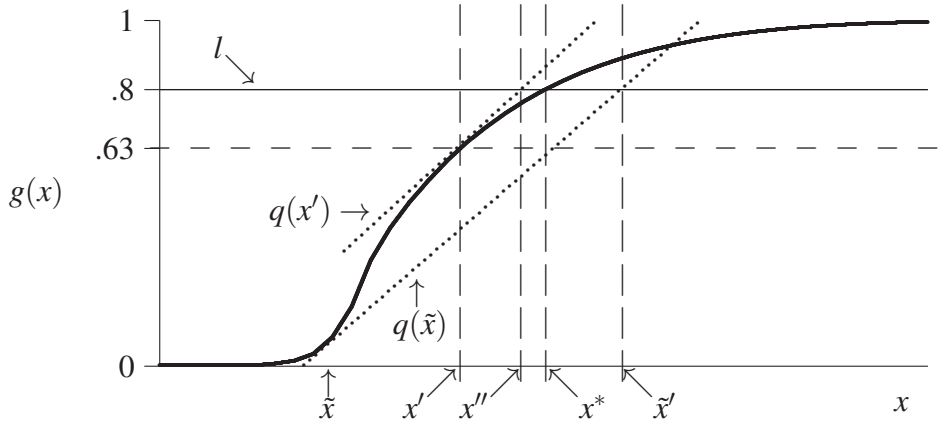

Figure 3.1 – Génération de coupes par sous-gradients sur la courbe du niveau de service par rapport à un groupe d'agents dans une région concave et une région convexe.

Puisque g est une fonction non continue, nous calculons le sous-gradient au point $\mathbf{x}'$

par la méthode des différences finies en incrémentant le nombre d'agents d'un groupe à la fois. Une évaluation (simulation) est requise pour déterminer chaque élément du vecteur $\mathbf{q}'(\mathbf{x}')$:

$$q'_j(\mathbf{x}') = g(\mathbf{x}' + \mathbf{e}_j) - g(\mathbf{x}'), \qquad \forall j \in \mathscr{M}. \tag{3.3}$$

Générer un sous-gradient demande donc m simulations. Cependant, il nous est impossible d'assurer que g soit concave à un point $\mathbf{x}'$ et que $\mathbf{q}'$ soit réellement un sous-gradient en ce point.

3.2 Initialisation de l'algorithme

L'algorithme commence avec une solution irréalisable, mais nous ne voulons pas qu'elle soit trop piètre, car les solutions avec un faible g risquent de se trouver dans des régions convexes du problème. Telle que montrée dans la figure 3.1, la génération d'une coupe par le sous-gradient dans une région convexe peut éliminer la solution optimale. Nous ajoutons donc des contraintes préliminaires pour couvrir un minimum de la charge des appels tel qu'il faut au moins $\alpha_i \rho_i$ agents capables de servir les appels du type i. Dans le cas sans abandon, α_i est au minimum à 1. Lorsque le taux d'abandon η_i est élevé, nous pouvons permettre que α_i soit légèrement plus bas que 1. Nous disposons de deux méthodes différentes pour résoudre ce problème de couverture. Il faut noter que ces méthodes ne considèrent pas la politique de routage utilisée et ne supposent aucune liste de priorité.

3.2.1 Graphe et flot maximal

La méthode décrite dans Ceẓik et L'Ecuyer [14] pour initialiser l'algorithme construit un graphe possédant $m+n+2$ sommets (voir la figure 3.2), soit un sommet par type d'appel, un par groupe d'agents et 2 sommets artificiels (source et puits). Les sommets sont reliés par des arcs de la manière suivante : de chaque type d'appel aux groupes d'agents pouvant le servir (ARC1) (autrement dit, il existe un arc entre le type d'appel i et le groupe j ssi $i \in \mathscr{K}_j$), de la source à chaque type d'appel (ARC2) et de chaque groupe d'agents au puits (ARC3). La capacité minimale est zéro pour tous les arcs. La

capacité maximale des arcs ARC2 correspond à $\alpha_i \rho_i$ où ρ_i est la charge du type d'appel i en erlangs. La capacité maximale des arcs ARC3 est égale au nombre d'agents dans chaque groupe respectivement et pour ARC1, elle est infinie. Lors de l'initialisation, nous pouvons supposer qu'il y a zéro agent. Cette méthode ne discerne pas les différents types de flots d'appels. Par conséquent, les durées de service $1/\mu_i$ sont supposées indépendantes des groupes d'agents.

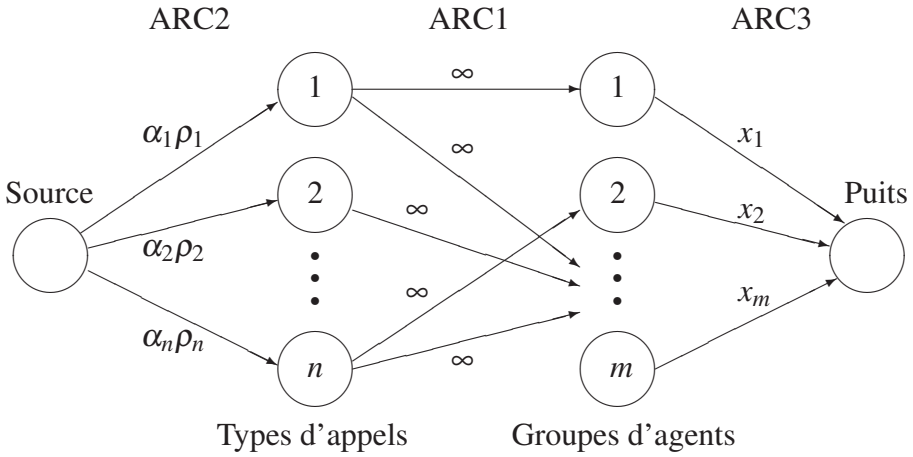

Figure 3.2 – Modèle d'un graphe à résoudre dans le problème du flot maximal indiquant la capacité supérieure de chaque arc, la capacité inférieure étant zéro.

Un problème de flot maximal est résolu à l'aide d'un solutionneur externe. Notons que la quantité de flot arrivant au puits ne peut pas être supérieure à $s = \sum_{i \in \mathscr{N}} \alpha_i \rho_i$ à cause de la capacité des arcs sortant de la source. Si la quantité de flot arrivant au puits est inférieure à s, alors il existe un ensemble d'arcs de type ARC2 qui ne sont pas à leur capacité maximale. Dénotons par $\mathscr{C}_1 \subseteq \mathscr{N}$ l'ensemble des types d'appels correspondant à ces arcs. Nous avons besoin d'une nouvelle contrainte qui incrémentera le nombre d'agents (par conséquent, les capacités des arcs ARC3). À cause de la polyvalence des agents, il n'est pas nécessairement optimal d'incrémenter uniquement parmi les groupes pouvant servir un type d'appel dans $\mathscr{C}_1$. Notons l'ensemble de ces groupes par $\mathscr{A}_1 = \{j : \mathscr{K}_j \cap \mathscr{C}_1 \neq \emptyset, j \in \mathscr{M}\}$. Nous pouvons optimiser indirectement en incrémentant des groupes qui ne sont pas dans $\mathscr{A}_1$. Par exemple, si un type d'appel $i \notin \mathscr{C}_1$ (couverture satisfaite) est servi par les groupes $j \in \mathscr{A}_1$ et $k \notin \mathscr{A}_1$, avec $c_j > c_k$, il serait plus intéressant

d'incrémenter x_k que x_j, tout en permettant aux agents du groupe j de se concentrer plutôt sur les appels de types $\mathscr{C}_1$. Dans l'implémentation, un algorithme d'étiquetage est utilisé pour trouver les ensembles $\mathscr{C} \supseteq \mathscr{C}_1$ et $\mathscr{A} \supseteq \mathscr{A}_1$. La contrainte linéaire à ajouter est :

$$\sum_{j \in \mathscr{A}} x_j \geq \left\lceil \sum_{i \in \mathscr{C}} \alpha_i \rho_i \right\rceil . \tag{3.4}$$

L'algorithme d'étiquetage implémenté sert à identifier une coupe minimale, voir la figure 3.3. Une coupe minimale partitionne les sommets d'un graphe en 2 sous-ensembles, soient $\mathscr{D}$ et $\bar{\mathscr{D}}$ avec le sommet source appartenant à $\mathscr{D}$ et le sommet puits à $\bar{\mathscr{D}}$, tels que la quantité de flot transitant entre les 2 sous-ensembles soit égale au flot maximal. Nous obtenons que $\mathscr{C} = \mathscr{N} \cap \mathscr{D}$ et $\mathscr{A} = \mathscr{M} \cap \mathscr{D}$, en omettant le sommet source.

Un centre d'appels peut avoir au maximum 2^n ensembles $\mathscr{C}$ distincts. Un tel cas se présente lorsque chaque type d'appel est servi uniquement par un groupe dédié et qu'il n'y a aucun agent polyvalent. Cependant, le nombre de contraintes requises est généralement proportionnel à n puisqu'il existe plusieurs contraintes dégénérées dues à la polyvalence des agents. Cette méthode, en résolvant le problème du flot maximal, est utilisée afin d'éviter d'ajouter toutes ces contraintes dégénérées au problème linéaire et inclure seulement celles qui s'avèrent nécessaires au cours de l'optimisation.

Un avantage de cette méthode (comparée à celle de la section 3.2.2) est de limiter le nombre de variables au nombre de groupes d'agents. Le problème de flot est résolu à part et puisqu'il est un problème en nombres réels, la résolution d'une instance est rapide avec un bon solutionneur.

Parce que cette méthode n'ajoute qu'une seule contrainte à la fois, il faut l'exécuter plusieurs fois jusqu'à ce que la couverture des charges soit satisfaite. L'inconvénient est le nombre d'appels et le temps total d'exécution du solutionneur pour résoudre les problèmes de flot maximal pour les centres d'appels de grande dimension. Malgré que la résolution d'une instance soit rapide, le temps total pour générer les contraintes nécessaires n'est pas toujours négligeable. La génération d'une seule contrainte par coupe minimale est une limitation qui augmente le nombre d'appels au solutionneur. Parfois, une contrainte pourrait être remplacée par plusieurs contraintes dominantes. Nous pou-

1. Résoudre le problème de flot maximal du graphe à l'aide d'un solutionneur externe.
2. Si la quantité de flot arrivant au puits est égale à $\sum_{i \in \mathscr{N}} \alpha_i \rho_i$, terminer la procédure.
3. Initier une pile vide $\mathscr{P}$.
4. Marquer la source et la placer sur $\mathscr{P}$.
5. Faire jusqu'à ce que $\mathscr{P}$ soit vide :
   (a) k = sommet sur le dessus de $\mathscr{P}$. Sortir k de $\mathscr{P}$.
   (b) Pour chaque arc (k, j) :
   Si j n'est pas marqué et la quantité de flot passant par l'arc (k, j) est inférieure à sa capacité supérieure, marquer j et le mettre sur $\mathscr{P}$.
   (c) Pour chaque arc (j, k) :
   Si j n'est pas marqué et la quantité de flot passant par l'arc $(j, k) > 0$, marquer j et le mettre sur $\mathscr{P}$.
6. Soit $\mathscr{D}$ l'ensemble des sommets marqués, nous le partitionnons en $(\mathscr{C}, \mathscr{A})$ tel que $\mathscr{C} = \mathscr{M} \cap \mathscr{D}$ et $\mathscr{A} = \mathscr{N} \cap \mathscr{D}$ (omettant le sommet source).
7. Ajouter la contrainte :

$$\sum_{j \in \mathscr{A}} x_j \geq \left\lceil \sum_{i \in \mathscr{C}} \alpha_i \rho_i \right\rceil .$$

Figure 3.3 – Algorithme pour générer une contrainte à partir d'une coupe minimale.

vons penser au cas où il n'y a aucun agent polyvalent et que les appels du type i ne sont servis que par le groupe i. Il serait plus rapide d'ajouter directement les contraintes $x_i \geq \alpha_i \rho_i$ sans résoudre le problème de flot. Par contre, pour un même problème, il est possible d'exporter et sauvegarder ces contraintes dans un fichier et de les réutiliser la fois suivante afin de réduire le temps d'initialisation.

Un autre désavantage est l'hypothèse d'indépendance des durées de service $1/\mu_i$ par rapport aux groupes d'agents. Ceci nous amène à formuler notre sous-problème autrement pour permettre d'avoir des $\mu_{i,j}$ différents pour différents groupes j dans la section 3.2.2.

3.2.2 Équations de couverture des charges pour des durées de service dépendantes des groupes d'agents

Lorsque la durée de service pour un type d'appel i est dépendante des groupes d'agents, nous ne pouvons pas utiliser la méthode de flot maximal décrite dans la section 3.2.1 utilisant la charge ρ_i pour résoudre le problème de couverture de charges. Dans ce cas-ci, μ_i est remplacé par $\mu_{i,j}$ pour le groupe j et il n'y a pas de ρ_i.

Soient $w_{i,j}$ le nombre d'agents du groupe j servant les appels du type i et $\mathscr{G}_i = \{j : i \in \mathscr{K}_j, j \in \mathscr{M}\}$ l'ensemble des groupes qui peuvent servir les appels du type i. Les équations à ajouter dans (P3) sont :

$$\begin{aligned} \sum_{i \in \mathscr{K}_j} w_{i,j} = &\quad x_j, \qquad \forall j \in \mathscr{M} \\ \sum_{j \in \mathscr{G}_i} \mu_{i,j} w_{i,j} \geq &\quad \alpha_i \lambda_i, \quad \forall i \in \mathscr{N} \\ w_{i,j} \geq &\quad 0, \qquad \forall i \in \mathscr{K}_j, \forall j \in \mathscr{M}. \end{aligned}$$

Le nombre de variables additionnelles est égal à la somme du nombre d'habiletés de chaque groupe ($\sum_{j=1}^{m} |\mathscr{K}_j|$) et le nombre d'équations additionnelles est $m+n$. Notons que les $w_{i,j}$ sont des variables réelles.

L'avantage est que le taux de service de chaque groupe peut être différent et que ce système d'équations statiques permet de modifier les coefficients sans générer de nouvelles contraintes. Contrairement à la méthode de la section 3.2.1, ces équations sont ajoutées lors de l'initialisation et il n'y a aucun sous-problème à résoudre.

L'inconvénient est l'ajout des variables $w_{i,j}$ au problème principal qui peut allonger le temps d'optimisation du solutionneur, en particulier si résolu en nombres entiers.

Dans le chapitre des exemples numériques, l'initialisation de cet algorithme est effectuée à l'aide de ces équations.

3.3 Autres méthodes heuristiques

En pratique, les algorithmes rencontrent souvent des difficultés dans les cas extrêmes. Dans certains cas, nous ne pouvons pas utiliser des coupes basées sur les sous-gradients

et la couverture de charges lors de l'initialisation n'est pas suffisante, alors il faut faire appel à d'autres heuristiques.

3.3.1 Lorsque le niveau de service d'un type d'appel est trop faible

Dans la section 3.2, nous avons présenté deux heuristiques pour débuter l'algorithme en satisfaisant une couverture de charges, espérant que les SL ne seraient pas trop faibles. Il n'est cependant pas trivial de trouver le bon α_i à utiliser et nous ne voulons pas générer un sous-gradient lorsque g_i est proche de zéro. Même s'il est possible qu'il n'y ait pas de minimum strict sur le SL par type d'appel, les centres d'appels n'accepteraient pas un SL de 10%. L'algorithme impose un seuil minimal ($l_{\min}$) sur g_i avant de pouvoir générer un sous-gradient. Dans un tel cas, nous choisissons le type d'appel $i = \arg\min_{k \in \mathcal{N}} \left\{ g_k(\mathbf{x}) : g_k(\mathbf{x}) < l_{\min} \right\}$ et incrémentons α_i. Ce qui mènera à la génération de nouvelles contraintes de couverture de charges.

Avec les équations présentées dans la section 3.2.2, le changement de α_i est direct, il suffit de changer le coefficient de la contrainte. Par contre, dans le cas de la méthode résolvant le flot maximal (section 3.2.1), le nombre de contraintes augmente et il est fort probable que certaines anciennes contraintes deviennent dominées par les nouvelles. Cependant, dans notre implémentation, nous ne faisons pas ce type de vérification. Certains solutionneurs, tel que Cplex, possèdent déjà des préprocesseurs pour détecter et simplifier ces cas. Soit $\mathcal{G}_i$ l'ensemble des groupes capables de servir les appels du type i, une alternative plus simple est d'incrémenter uniquement le nombre d'agents courant de $\mathcal{G}_i$ au lieu des contraintes de couverture, puisque les groupes dans $\mathcal{G}_i$ ont un impact direct sur g_i. Soient la solution courante $\mathbf{x}'$ et $a > 1$, la contrainte est :

$$\sum_{j \in \mathcal{G}_i} x_j \geq \left\lceil a \sum_{j \in \mathcal{G}_i} x'_j \right\rceil . \tag{3.5}$$

Ajustement des contraintes heuristiques

Il n'est pas trivial de configurer ces heuristiques de manière à limiter leur influence sur la solution finale. Si l'incrémentation de α_i ou la valeur a est trop faible, l'algo-

rithme risque d'itérer plusieurs fois avant de sortir de la région où la coupe basée sur le sous-gradient est dangereuse, car l'utilisation de sous-gradients lorsque le SL est faible présente un plus grand risque d'éliminer la solution optimale (voir la figure 3.1). De plus, il faut résoudre le problème linéaire puis simuler à chaque itération, ce qui peut devenir coûteux en temps d'exécution.

Nous avons établi une procédure de suivi de l'impact du changement sur g_i après chaque coupe heuristique. Nous ciblons une amélioration du g_i d'au plus 10% au-dessus du seuil $l_{\min}$ pour le type d'appel i ciblé. Nous utilisons une recherche binaire pour trouver ce paramètre avec un ajout minimum d'un agent. Cette procédure d'auto-correction (optionnelle dans notre programme) permet un choix plus souple sur la taille d'incrément de α_i et de a.

3.3.2 Sous-gradient faible ou nul

Il arrive que la taille du sous-gradient soit nulle ou presque nulle à une itération courante, souvent à cause du bruit de la simulation et dans les régions plates de g lorsque g est faible. Dans un tel cas, la coupe basée sur le sous-gradient ne serait donc pas applicable. En se référant à la figure 3.1, un sous-gradient nul n'intersectera jamais avec l.

Nous établissons donc un seuil minimal sur la taille de $\mathbf{q}$ avant d'ajouter une coupe. Si la taille de $\mathbf{q}$ est sous ce seuil, nous utilisons une autre coupe heuristique.

S'il y a un type d'appel tel que $g_i < l_{\min}$, alors nous ajoutons déjà une contrainte présentée dans la section 3.3.1. Sinon, une contrainte similaire couvrant tous les appels du centre sera ajoutée. Soit $\mathbf{x}'$ la solution courante et $a > 1$, la contrainte à ajouter est :

$$\sum_{j=1}^{m} x_j \geq \left\lceil a \sum_{j=1}^{m} x'_j \right\rceil . \tag{3.6}$$

3.3.3 Bruit de la simulation

La simulation permet d'estimer des fonctions complexes, mais la précision dépend du nombre de simulations effectuées et l'efficaté des coupes basées sur les sous-gradients

repose grandement sur cette précision. Cependant, dans la pratique, les gestionnaires de centres d'appels n'ont généralement pas assez de temps pour utiliser de longues simulations lors de l'optimisation. Il faut donc faire un compromis entre le temps d'exécution et la performance de l'optimisation.

Avec de courtes simulations, le bruit peut générer artificiellement des régions convexes qui peuvent donner de très mauvaises coupes. Il arrive même que g (et g_i) diminue lorsque nous ajoutons un agent additionnel. Si le simulateur estime que l'ajout d'un agent améliore peu le SL alors que c'est faux, la coupe par le sous-gradient résultant risque d'éliminer la solution optimale.

La figure 3.4 (inspirée d'une figure similaire dans Ceşik et L'Ecuyer [14]) montre l'imperfection de la courbe de g à cause du bruit de la simulation, comparée avec la courbe pointillée calculée par de longues simulations. Si nous générions une coupe aux points A à E ou à tout $x <$ A, la coupe éliminerait la solution optimale ou il n'existerait tout simplement aucune coupe si $q(x) \leq 0$.

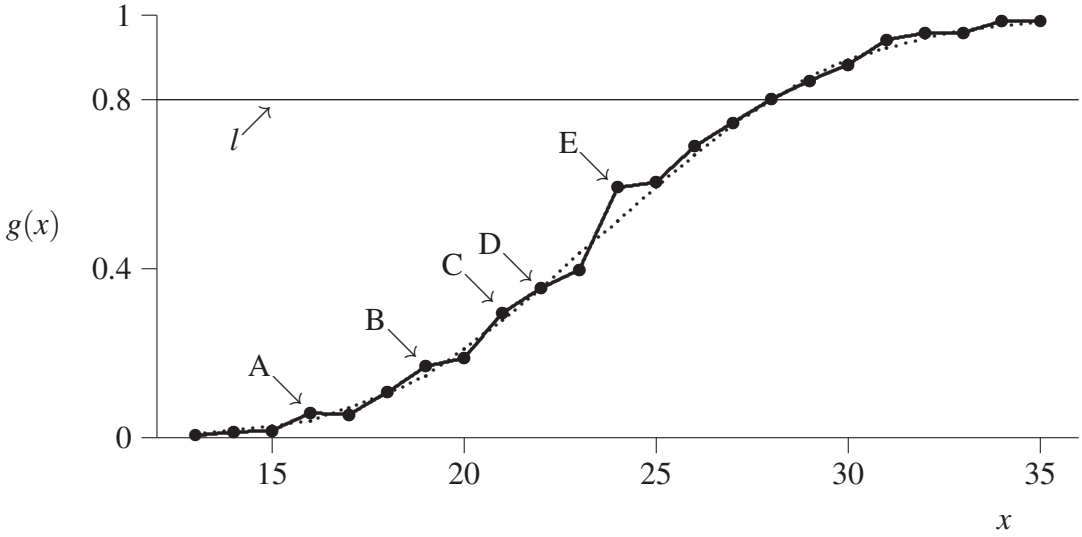

Figure 3.4 – Échantillon de $g(x)$ montrant l'imperfection de la courbe causée par le bruit de la simulation (solide) comparée avec une courbe avec peu de bruit (pointillée).

Une façon d'atténuer l'erreur du sous-gradient causée par le bruit sans augmenter le temps de simulation est d'utiliser un pas plus grand lors de la mesure du sous-gradient

par les différences finies. Nous ajoutons le paramètre du pas $d \geq 1$ à l'équation (3.3) :

$$q_j(\mathbf{x}) = \frac{g(\mathbf{x} + d\mathbf{e}_j) - g(\mathbf{x})}{d}. \tag{3.7}$$

Cependant, ce changement ne résout pas le problème de non-concavité lorsque g est faible et un d trop grand pourrait également éliminer la solution optimale à cause de la concavité de g.

3.4 Lorsque le temps de résolution est trop long

Le temps de résolution d'un problème (P3) en nombres entiers augmente exponentiellement en fonction de sa taille. Bien que le temps de résolution soit négligeable pour un centre d'appels de petite taille, le temps consommé par le solutionneur pour un grand problème devient rapidement le goulot d'étranglement du programme.

Deux relaxations au problème (P3) ont été testées pour contourner cette difficulté :

1. Appliquer la relaxation lagrangienne aux contraintes de coupes basées sur les sous-gradients.
2. Utiliser la relaxation LP en transformant le vecteur en nombres entiers $\mathbf{x}$ en un vecteur en nombres réels.

3.4.1 Relaxation lagrangienne

Nous avons appliqué la méthode de la relaxation lagrangienne (Ahuja et al. [2]) sur les contraintes de coupes basées sur les sous-gradients. De la formulation (P3), les coupes basées sur les sous-gradients sont représentées par la contrainte $\mathbf{Bx} \geq \mathbf{b}$ et les autres contraintes heuristiques par $\mathbf{Ex} \geq \mathbf{e}$. Pour raccourcir la notation, ces contraintes représentent à la fois celles pour le SL global et celles par type d'appel. À l'itération v, $(\text{P3})^{(v)}$ devient :

$$
\begin{aligned}
\bar{Z}^{(v)} = \min \quad & \mathbf{c}^{\mathrm{t}}\mathbf{x} \\
\text{sujet à :} \quad & \\
& \mathbf{B}^{(v)}\mathbf{x} \geq \mathbf{b}^{(v)} \\
& \mathbf{E}^{(v)}\mathbf{x} \geq \mathbf{e}^{(v)} \\
& \mathbf{x} \geq 0 \text{ et entier}
\end{aligned}
$$

et par la relaxation lagrangienne des coupes basées sur les sous-gradients :

$$
\begin{aligned}
\mathscr{L}^{(v)}(\theta) = \min \quad & \mathbf{c}^{\mathrm{t}}\mathbf{x} + \theta(\mathbf{b}^{(v)} - \mathbf{B}^{(v)}\mathbf{x}) \\
\text{sujet à :} \quad & \\
& \mathbf{E}^{(v)}\mathbf{x} \geq \mathbf{e}^{(v)} \\
& \mathbf{x} \geq 0 \text{ et entier}
\end{aligned}
\tag{P4}
$$

où $\theta = (\theta_1, \ldots, \theta_{|\mathbf{b}^{(v)}|})$ est le vecteur semi-positif, $\theta_k \geq 0$, des multiplicateurs de Lagrange.

Le dual de la relaxation lagrangienne, $\mathscr{L}^{(v)*}$, est défini comme suit :

$$\mathscr{L}^{(v)*} = \max_{\theta} \mathscr{L}^{(v)}(\theta).$$

La relation entre $\mathscr{L}^{(v)}(\theta), \mathscr{L}^{(v)*}, \bar{Z}^{(v)}$ et Z (en supposant qu'aucune mauvaise coupe n'élimine la solution optimale) est telle que :

$$\mathscr{L}^{(v)}(\theta) \leq \mathscr{L}^{(v)*} \leq \bar{Z}^{(v)} \leq Z.$$

L'objectif dans la relaxation lagrangienne est de trouver la solution la plus proche de $\mathscr{L}^{(v)*}$. Cependant, la difficulté est d'employer les bons multiplicateurs de Lagrange θ. Trouver les multiplicateurs optimaux revient à résoudre le dual lagrangien.

Optimisation par le sous-gradient de $\mathscr{L}^{(v)}(\theta)$

Nous avons implémenté la procédure heuristique itérative pour trouver θ, telle que présentée par Ahuja et al. [2]. Supposons qu'il existe une solution optimale unique $\bar{\mathbf{x}}$ pour le problème $\mathscr{L}(\bar{\theta})$ (les indices v sont omis pour simplifier). Cette heuristique suppose que $\bar{\mathbf{x}}$ reste une solution optimale pour $\mathscr{L}(\bar{\theta}+\varepsilon)$ où ε est un petit vecteur. $\mathscr{L}(\theta)$ est différentiable au point $\bar{\theta}$ et la valeur de la dérivée est $(\mathbf{b}-\mathbf{B}\bar{\mathbf{x}})$. Le terme *sous-gradient* est utilisé, car $\mathscr{L}$ n'est pas nécessairement différentiable lorsqu'il y a plusieurs solutions (si $\mathbf{x}$ varie avec ε). La fonction économique $\mathbf{c}^{\mathbf{t}}\mathbf{x}+\theta(\mathbf{b}-\mathbf{B}\mathbf{x})$ est linéaire en fonction des multiplicateurs θ et représente un hyperplan lorsque $|\theta|>2$. La fonction $\mathscr{L}(\theta)$ est ainsi définie par un ensemble d'hyperplans. L'heuristique est de se déplacer itérativement sur θ en direction du sous-gradient avec un pas de taille δ. À l'itération r de cette procédure, nous avons :

$$\theta^{(r+1)} = \theta^{(r)} + \delta^{(r)}(\mathbf{b}-\mathbf{B}\mathbf{x}^{(r)}).$$

Afin que la solution converge, cette méthode propose les critères suivantes :

$$\sum_{r=1}^{\infty} \delta^{(r)} = \infty$$

$$\lim_{r\to\infty} \delta^{(r)} = 0.$$

Un pas de taille $\delta^{(r)} = 1/r$ respecte ces critères. Le problème (P4) doit être résolu plusieurs fois jusqu'à convergence des solutions.

Cependant, ceci reste une méthode de relaxation. Il arrive que la solution reste inchangée en passant de l'itération v à $(v+1)$ par l'ajout de nouvelles contraintes dans $\mathbf{B}^{(v)}$ et $\mathbf{b}^{(v)}$. Dans un tel cas, l'algorithme entre dans une boucle infinie. Il faudrait modifier l'algorithme pour répondre à ce problème, mais nous n'avons pas poursuivi notre recherche dans cette direction. Nous avons plutôt choisi de poursuivre avec la relaxation LP.

Notons que le problème $\mathscr{L}(\theta)$ est très différent de celui de Bhulai et al. [6] où ils optimisent directement le problème (P2), mais contraint uniquement sur le SL global. Une grande différence est que leur problème relaxé demeure stochastique. Avec un seul mul-

tiplicateur de Lagrange, le multiplicateur optimal est trouvé par une recherche binaire et au lieu de résoudre un programme linéaire, ils utilisent une recherche par voisinage à chaque itération.

3.4.2 Relaxation LP du problème

Avec la technologie présente, les ordinateurs peuvent résoudre très rapidement les problèmes linéaires en nombres réels de très grandes dimensions. Les problèmes qui pouvaient prendre plusieurs heures deviennent négligeables avec la relaxation LP. À chaque solution **x** retournée par le solutionneur, chaque élément x_j est arrondi à $\lceil x_j \rceil$ étant donné que le simulateur ne prend que des nombres entiers. Une recherche locale est appliquée à la solution finale afin de réduire le coût additionnel causé par l'arrondissement au plafond lors de la dernière itération.

Recherche locale autour de la solution finale

Puisque l'algorithme termine à la première solution réalisable, il se pourrait que nous puissions facilement réduire le coût en cherchant dans le voisinage. La figure 3.5 présente une méthode de recherche locale très simple. Cet algorithme glouton réduit le nombre d'agents en commençant par les plus coûteux tant que la solution demeure réalisable.

Cette optimisation locale est surtout utile lorsque le problème est relaxé en un problème en nombres réels. D'après nos expériences, cette correction peut réduire le coût final de 1% à 2%. Par contre, l'effet est plutôt négligeable pour le problème en nombres entiers original.

3.5 Implémentation de l'algorithme

La figure 3.6 présente l'ordre d'exécution de l'algorithme. Le rectangle pointillé est l'étape de la couverture de charges lorsque nous utilisons l'algorithme de flot maximal présenté à la section 3.2.1. Ces procédures sont omises si nous utilisons plutôt le système d'équations de la section 3.2.2.

1. Trier les groupes d'agents en ordre décroissant par unité de coût dans une liste $\mathscr{L}$, $\mathscr{L}(1)$ étant le plus coûteux.
2. Tant que $\mathscr{L}$ n'est pas vide, faire :
   (a) Retirer le maximum d'agents du groupe $\mathscr{L}(1)$ tout en demeurant une solution réalisable.
   (b) Retirer $\mathscr{L}(1)$ de $\mathscr{L}$.

Figure 3.5 – Algorithme simple d'une recherche locale.

Nous avons besoin d'un simulateur et d'un solutionneur (deux solutionneurs dans le cas avec le problème de flot maximal). Notre implémentation utilise la librairie pour la simulation des centres de contacts développée en Java par Buist et L'Ecuyer [10]. Notre logiciel est compatible avec tout évaluateur de centres d'appels qui implémente l'interface définie dans cette librairie. Puisqu'il existe différents solutionneurs linéaires, nous avons créé une interface pour gérer les opérations élémentaires des problèmes linéaires.

Interfaçage avec les librairies de solutionneurs externes

Étant donné que nous ne voulons pas que notre logiciel soit dépendant d'un solutionneur spécifique, l'usager a la tâche d'implémenter l'interface entre sa librairie et le logiciel, s'il n'est pas déjà disponible.

Nous avons implémenté cette interface pour deux librairies de solutionneurs : Cplex [25] et OR-Objects 1.2.4 [19]. Cplex est un solutionneur commercial puissant, mais coûteux. Nous avons également implémenté l'interface du solutionneur pour la librairie de OR-Objects 1.2.4 qui est gratuite, mais moins développée et elle n'est plus maintenue depuis l'an 2001. Nous utilisons cette librairie seulement pour résoudre les problèmes linéaires en nombres réels.

Voici les opérations élémentaires qui peuvent être demandées :

— Initialiser les variables avec des bornes supérieures et inférieures.

— Assigner une fonction objectif de maximisation ou de minimisation.

— Ajouter des contraintes de type $\geq$, $=$ ou $\leq$.

— Modifier les contraintes existantes.

— Résoudre et recueillir la solution.

— Importer et exporter le problème linéaire vers un fichier.

Pour le moment, les variables ne peuvent être déclarées qu'à l'initialisation du problème, aucune ne peut être ajoutée par la suite.

À l'aide des fonctions d'exportation et d'importation, nous pouvons ajouter des contraintes manuellement et importer le problème modifié lors de l'initialisation du programme. OR-Objects 1.2.4 n'utilise que le format MPS alors que Cplex accepte les formats MPS et LP.

Exemple d'un problème linéaire :

$$\begin{aligned} \min \quad & x_0 + 7x_1 \\ & x_0 + x_1 \quad \geq 20.0 \\ & x_0, x_1 \quad \geq 0 \end{aligned}$$

```
* Metadatum lp.isMaximize "false"
ROWS
 N  OBJ
 G  cx0lb
 G  cx1lb
 G  c
COLUMNS
    x0        OBJ       1.0
    x0        c         1.0
    x0        cx0lb     1.0
    x1        OBJ       7.0
    x1        c         1.0
    x1        cx1lb     1.0
RHS
    RHS       cx0lb     0.0
    RHS       cx1lb     0.0
    RHS       c        20.0
```

Listing 3.1 – "Exemple d'un fichier MPS pour OR-Objects 1.2.4"

```
\Problem name: exemple.lp
Minimize
 obj: x0 + 7.0 x1
Subject To
 c1: x0 + x1 >= 20.0
Bounds
      x0 >= 0
      x1 >= 0
End
```

Listing 3.2 – "Exemple d'un fichier LP de Cplex"

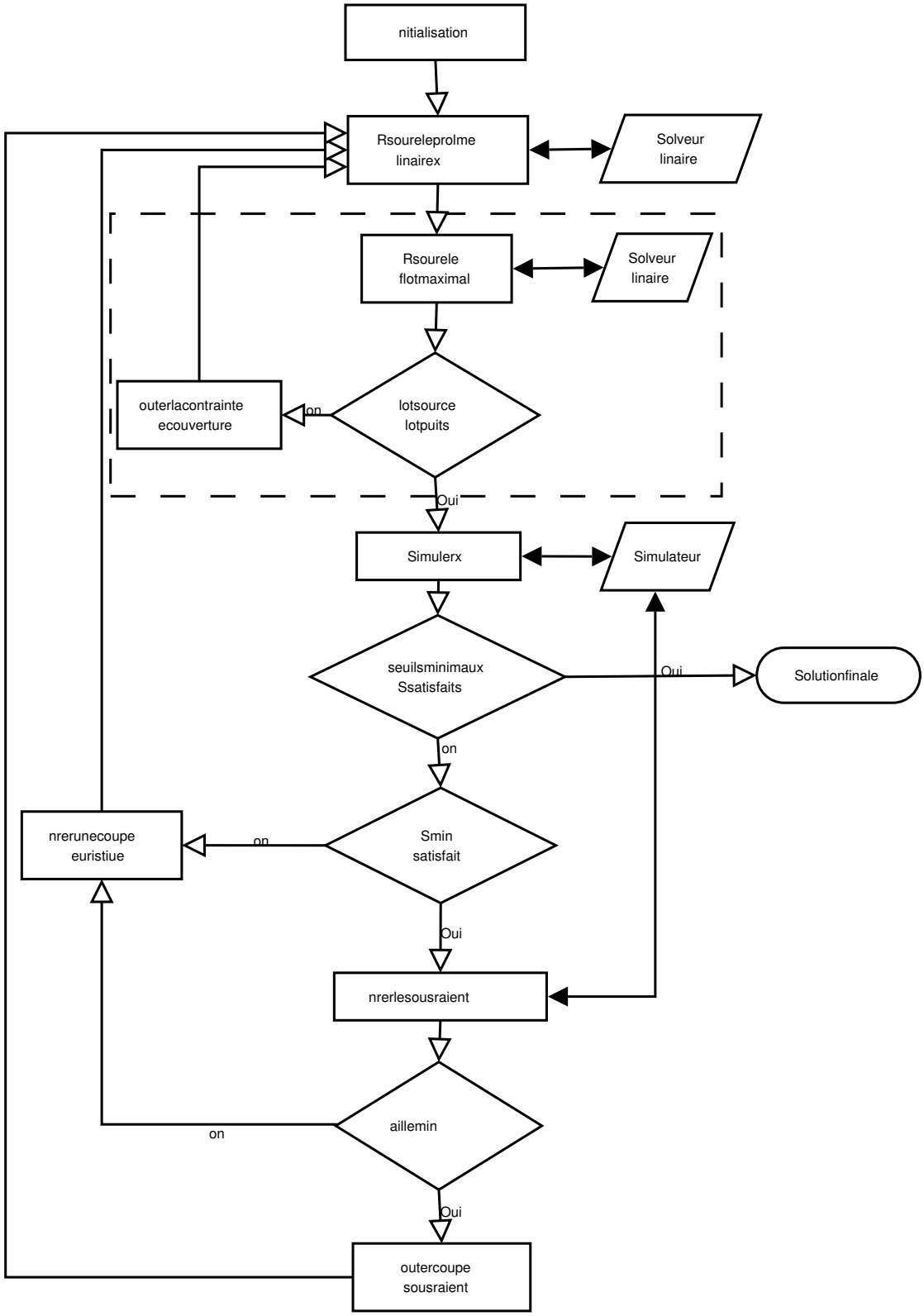

Figure 3.6 – Diagramme de l'algorithme d'optimisation par coupes linéaires et simulation.

CHAPITRE 4

OPTIMISATION PAR UNE RECHERCHE PAR VOISINAGE RANDOMISÉE

L'algorithme d'optimisation présenté dans ce chapitre est une méthode de recherche locale randomisée. À chaque itération, l'algorithme passe d'une solution à une autre via son voisinage tout en réduisant le coût d'affectation du personnel. Le nombre total de solutions possibles augmente rapidement avec la taille du problème ainsi que le nombre d'évaluations de g. Pour cette raison, la simulation est trop coûteuse pour être utilisée avec ce genre d'algorithme dans la pratique. Étant donné qu'il n'existe pas de formule analytique précise pour les centres d'appels avec agents polyvalents et routage par compétence (à part pour quelques configurations bien spécifiques de petits centres d'appels), g est évalué par une approximation grossière que nous avons développée. Cet algorithme d'approximation est basé sur la théorie des files d'attente où le système a supposément atteint un régime permanent (*steady-state*).

4.1 Processus de naissance et de mort

L'état d'un système de file d'attente avec un seul type de client (voir la figure 4.1) peut être représenté comme un processus stochastique $X(t)$, où $X(t)$ est le nombre de clients au temps t. En supposant qu'il puisse être modélisé par une chaîne de Markov en temps continu, le processus de naissance et de mort est le modèle analytique le plus élémentaire pour calculer les performances de ce système. Ce modèle suppose que les temps d'inter-arrivée des appels et les durées de service des appels sont des variables aléatoires de lois exponentielles (nous disons dans ce cas que le processus est markovien) et qu'il y a un nombre limité de serveurs. Il est souvent dénoté par $M/M/s$, où les M indiquent respectivement les processus d'inter-arrivée et de service markoviens et s exprime le nombre de serveurs. L'analyse de ce système dans un régime permanent correspond à celle d'un centre d'appels avec un seul type d'appel et une politique de routage du type premier arrivé, premier servi. Les formules analytiques pour le système

de file d'attente $M/M/s$ sont disponibles dans de nombreux livres dont Taylor et Karlin [38] et Hillier et Lieberman [23].

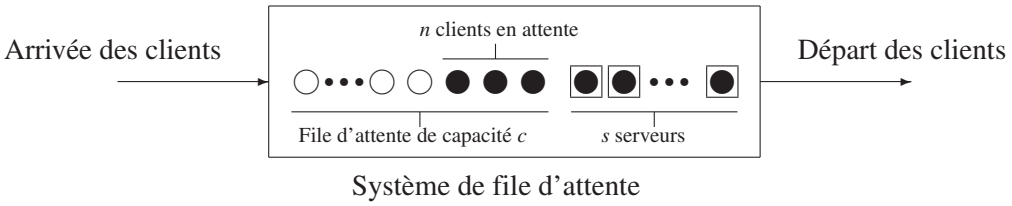

Figure 4.1 – Diagramme d'un système de file d'attente avec une capacité c, n clients en attente et s serveurs. Le nombre de clients dans ce système est $n+s$.

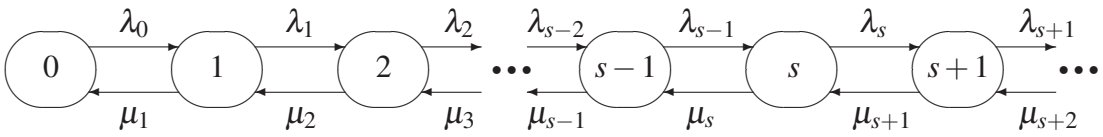

Figure 4.2 – Diagramme des taux de transition d'un processus de naissance et de mort.

La figure 4.2 montre les taux de transition entre chaque état du système (le nombre total de clients en service et en attente). Dans le cas simple d'une file $M/M/s$ de capacité infinie, où les clients n'abandonnent jamais, avec un taux d'arrivée des appels λ (le nombre potentiel de clients est supposé illimité), une durée moyenne de service $1/\mu$ et s agents en service, les taux de transition sont :

$$\lambda_k = \quad \lambda, \qquad k = 0,1,\ldots$$
$$\mu_k = \begin{cases} k\mu, & k = 1,2,\ldots,s \\ s\mu, & k = s+1, s+2,\ldots \end{cases}$$

Avec l'hypothèse que le système soit dans un régime permanent avec un taux de

service μ, $\pi_k(\mu)$ est la probabilité que le système se retrouve à l'état k. Nous avons que :

$$\pi_0(\mu) = \left(1 + \sum_{k=1}^{\infty} \prod_{v=1}^{k} \frac{\lambda_{v-1}}{\mu_v}\right)^{-1}. \tag{4.1}$$

Avec $\frac{\lambda}{s\mu} < 1$, nous obtenons :

$$\pi_0(\mu) = \left(\sum_{v=0}^{s-1} \frac{\lambda^v}{v!\mu^v} + \frac{\lambda^s}{s!\mu^s} \frac{s\mu}{s\mu - \lambda}\right)^{-1} \tag{4.2}$$

et pour $k > 0$:

$$\pi_k(\mu) = \frac{\lambda_{k-1}}{\mu_k} \pi_{k-1}(\mu). \tag{4.3}$$

À l'aide de ces probabilités, les performances de ce système peuvent être calculées avec les formules connues dont Erlang B et Erlang C. Cependant, l'analyse exacte d'un modèle de chaîne de Markov en temps continu pour plusieurs types d'appels et plusieurs groupes d'agents souffre de la malédiction de la dimentionalité : le nombre d'états augmente exponentiellement. Les politiques modernes de routage par compétence posent également un défi d'analyse difficile.

4.2 Modèle d'approximation avec perte

Koole et Talim [31] présentent un modèle d'approximation simplifié qui suppose que le centre d'appels ne possède aucune file d'attente, les appels sont perdus (aussi dits bloqués) lorsqu'il n'y a aucun agent libre pour y répondre immédiatement. Ce modèle se base sur le routage par priorité des appels. Soit $\mathscr{R}_i = \{r_i(1), r_i(2), \ldots, r_i(m_i)\}$ (avec m_i groupes pouvant le servir) l'ordre de visite d'un appel entrant de type i aux groupes d'agents, le groupe $r_i(j)$ est visité avant $r_i(j+1)$. À l'aide de cet ordre de priorité, ce modèle répartit les quantités d'appels parmi les groupes d'agents en utilisant les formules classiques de processus de naissance et de mort, dont la formule d'Erlang B.

Un modèle $M/M/s$ sans file d'attente possède les taux de transition suivants :

$$\begin{aligned} \lambda_k = &\quad \lambda, \quad k = 0, \ldots, s-1 \\ \mu_k = &\quad k\mu, \quad k = 1, \ldots, s. \end{aligned}$$

Les probabilités stationnaires sont :

$$\begin{aligned} \pi_0(\mu) &= \left(\sum_{v=0}^{s} \frac{\lambda^v}{v!\mu^v} \right)^{-1} \\ \pi_k(\mu) &= \frac{\lambda}{k\mu} \pi_{k-1}(\mu), \qquad k = 1, \ldots, s. \end{aligned}$$

La formule d'Erlang B donne la proportion des appels bloqués, ce qui correspond à la proportion du temps où tous les agents sont occupés. Soit $\rho = \lambda/\mu$, la formule est :

$$B(\rho, s) = \frac{\rho^s/s!}{\sum_{v=0}^{s}(\rho^v/v!)} = \pi_s(\mu) = 1 - \sum_{k=0}^{s-1} \pi_k(\mu). \tag{4.4}$$

Ce modèle approxime les performances en divisant l'analyse du centre d'appels par groupe d'agents plutôt que par type d'appel. Dans le cas simple d'un centre d'appels ayant seulement un type d'appel (l'indice du type d'appel est omis par simplicité) avec un taux d'arrivée λ et une durée de service d'une distribution exponentielle de moyenne $1/\mu$, ce modèle répartit les appels aux groupes suivant le routage par priorité. Cette répartition est calculée par un mécanisme de débordement du flot d'arrivée des appels. Soient $\gamma_{r(1)} = \lambda$ le flot global arrivant au groupe $r(1)$ et $\gamma_{r(j)}$ le flot d'appels arrivant au groupe $r(j)$, la relation entre les indices j et $j-1$ est :

$$\gamma_{r(j)} = \gamma_{r(j-1)} B(\rho_{r(j-1)}, x_{r(j-1)})$$

où $\rho_k = \gamma_k/\mu, k \in \mathscr{M}$ et une affectation d'agents $\mathbf{x}$. Le flot d'appels arrivant au groupe $r(j)$ correspond au flot qui a été bloqué au groupe $r(j-1)$, ainsi $\gamma_{r(j)} \leq \gamma_{r(j-1)}$. La probabilité de blocage de ce centre est déterminée par $B(\rho_{r(m)}, x_{r(m)})$.

Dans le cas d'un centre avec plusieurs types d'appels et d'agents polyvalents, le flot

d'appels du type i arrivant au groupe j est dénoté par $\gamma_{i,j}$. Lors des calculs de débordement avec la formule d'Erlang B pour un groupe j, les charges sont additionnées pour obtenir $\bar{\rho}_j = \sum_{i=1}^{n}(\gamma_{i,j}/\mu_{i,j})$, qui sera passé en paramètre à la fonction B.

Cependant, ce modèle d'approximation se base sur des hypothèses qui ne sont pas réalistes. Dans la pratique, les centres d'appels possèdent des files d'attente et leurs performances ne sont pas évaluées sur la proportion des appels bloqués, mais généralement sur la proportion qui a attendu moins de τ secondes.

Koole et al. [30] utilisent la relation entre la probabilité de blocage $B(\rho,s)$ obtenue par la formule d'Erlang B et la probabilité de délai $C(\rho,s)$ de Erlang C pour un système $M/M/s$:

$$C(\rho,s) = \frac{sB(\rho,s)}{s-\rho\,[1-B(\rho,s)]} \tag{4.5}$$

et pour obtenir le SL (voir Cooper [16]) :

$$\mathbb{P}[W \leq \tau] = 1 - C(\rho,s)e^{-\tau(s\mu-\lambda)}, \quad (\tau \geq 0, \rho > s). \tag{4.6}$$

Cependant, il est difficile de calculer le SL par type d'appel à l'aide de ces formules, car le paramètre s est inconnu et nous savons que la fonction du SL peut être très sensible à ce paramètre. Ainsi, cette approximation est utilisée pour évaluer le niveau de service global.

Calculer le SL à partir de la probabilité de blocage global peut mener à des erreurs d'analyse importante. Prenons par exemple un petit centre d'appels composé de 2 types d'appels avec des taux d'arrivée égaux et 2 groupes d'agents, soient un spécialiste et un généraliste. Le SL global est évidemment limité par le nombre d'agents généralistes. Supposons maintenant qu'il n'y ait aucun généraliste et un très grand nombre de spécialistes (suffisamment pour qu'un type d'appel n'ait aucune attente), le véritable SL global serait alors à 50%. Regardons ce qui se passerait dans les formules précédentes : les $B(\rho,s)$ sont analysées par type d'appel et la probabilité de blocage global serait de 50% (ce qui est exact). La probabilité de délai global $C(\rho,s)$ serait proche de 50%, mais puisque le nombre total d'agents s est très grand, l'équation (4.6) retournerait un SL global élevé tel que le SL $\to 100\%$ lorsque $s \to \infty$. Cette erreur pourrait être évitée si nous

pouvions calculer le SL par type d'appel et les agréger pour obtenir le SL global. Une extension à ce modèle d'approximation permettant de calculer le SL par type d'appel est présentée dans la prochaine section.

4.3 Modèle d'approximation avec perte et délai

Avramidis et al. [4] proposent des extensions au modèle d'approximation avec perte de Koole et Talim [31] en ajoutant des files d'attente et la possibilité d'abandon des clients. La durée de patience d'un client pour un appel du type i est une variable exponentielle de moyenne $1/\eta_i$. Nous appelons cette approximation perte et délai ou *Loss-Delay* (LD).

Ce modèle suppose le même modèle de routage que Koole et Talim [31] sauf que les appels du type i attendent au dernier groupe d'agents de la liste de priorité $\mathscr{R}_i$, soit au groupe $r_i(m_i)$. Nous supposons que les files d'attente sont associées aux groupes d'agents et un groupe peut posséder au plus une file d'attente. Différents types d'appels attendant pour un même groupe d'agents doivent partager la même file et ils sont répondus suivant la politique FCFS. Ainsi, le SL des appels du type i n'est déterminé que par le groupe $r_i(m_i)$. Ce modèle suppose également qu'il n'existe aucune égalité sur le rang de priorité. Dans le cas où la politique de routage du centre d'appels possède des égalités sur les rangs, un ordre de priorité distinct devra être généré pour l'approximation. Nous ne supposons pas que ce modèle est réaliste, mais ceci permet d'obtenir une approximation, parfois grossière, qui est utile dans les algorithmes d'optimisation (voir le chapitre 5).

Au départ, il faut déterminer le rôle de chaque groupe d'agents sur les types d'appels qu'il peut servir (déterminés dans $\mathscr{K}$). Chaque groupe possède les ensembles $\mathscr{P}$ (perte) et $\mathscr{D}$(délai) tels qu'il doit servir tous (incluant ceux en attente) les appels de type $i \in \mathscr{D}$ ou seulement ceux venant d'arriver si $i \in \mathscr{P}$ et i ne peut pas être à la fois dans les 2 ensembles. Ainsi, pour le groupe j, le type d'appel i appartient à l'ensemble $\mathscr{P}_j$ ssi $i \in \mathscr{K}_j$ et $r_i(m_i) \neq j$, et i appartient à l'ensemble $\mathscr{D}_j$ ssi $i \in \mathscr{K}_j$ et $r_i(m_i) = j$. Les groupes d'agents sont divisés en trois partitions : $\mathscr{M}_p, \mathscr{M}_{pd}$ et $\mathscr{M}_d$, soient respectivement *pure perte*, *perte et délai* et *pur délai*. Un groupe j est de type *pure perte* s'il n'est le dernier

groupe d'aucun type d'appel ; autrement dit, $\mathscr{P}_j \neq \emptyset$ et $\mathscr{D}_j = \emptyset$. Un groupe j est de type *pur délai* s'il est le dernier groupe dans le routage pour tout $i \in \mathscr{K}_j$, soient $\mathscr{P}_j = \emptyset$ et $\mathscr{D}_j \neq \emptyset$. Finalement, un groupe j est de type *perte et délai* si $\mathscr{P}_j \neq \emptyset$ et $\mathscr{D}_j \neq \emptyset$.

Dans le modèle de processus de naissance et de mort d'un groupe j ayant s agents, il y a deux types de taux de naissance, soient λ^p et λ^d pour les types d'appels appartenant à $\mathscr{P}_j$ et $\mathscr{D}_j$ respectivement. Lorsque tous les agents sont occupés, le taux de naissance est composé uniquement des appels de type délai. Pour un ensemble d'états fini $X(t) \in \{0,1,\ldots,s+c\}$ où c représente la capacité de la file d'attente, les taux de transition sont :

$$\lambda_k = \begin{cases} \lambda^p + \lambda^d, & k = 1,2,\ldots,s-1 \\ \lambda^d, & k = s,s+1,\ldots,s+c \end{cases}$$

$$\mu_k = \begin{cases} k\mu, & k = 1,2,\ldots,s-1 \\ s\mu + (k-s)\eta, & k = s,s+1,\ldots,s+c. \end{cases}$$

Les probabilités stationnaires sont :

$$\begin{aligned} \pi_0(\mu) &= \left(1 + \sum_{k=1}^{s+c} \prod_{z=1}^{k} \frac{\lambda_{z-1}}{\mu_z}\right)^{-1} \\ \pi_k(\mu) &= \frac{\lambda_{k-1}}{\mu_k}\pi_{k-1}(\mu), \qquad k = 1,2,\ldots,s+c. \end{aligned}$$

Les taux d'arrivée des appels sont agrégés selon les types perte ou délai. Lorsque tous les agents sont occupés ($X(t) \geq s$), il n'y a que les appels de type $i \in \mathscr{D}_j$ qui entrent dans le système. Par contre, lorsqu'il y a plus de s appels, il se peut qu'il reste des appels de type $i \in \mathscr{P}_j$ en service. L'agrégation des durées de service est un peu plus complexe et elle est présentée dans les sections suivantes.

4.3.1 Analyse des groupes de type *pure perte*

Pour les groupes dans $\mathscr{M}_p$, les paramètres sont $\lambda^d = 0$ et $c = 0$. Les appels sont débordés vers les autres groupes et la proportion de ces appels est calculée à l'aide de la formule d'Erlang B, l'équation (4.4).

4.3.2 Analyse des groupes de type *perte et délai*

Nous analysons deux cas : sans abandon avec des files d'attente d'une capacité infinie et le cas avec des abandons et des capacités finies. Le premier cas possède des simplifications de formules qui rendent le calcul plus simple et plus rapide.

4.3.2.1 Cas sans abandon

Dans le cas sans abandon, nous supposons que $\eta = 0$ et $c = \infty$. Pour les groupes dans $\mathscr{M}_{pd}$, la propriété PASTA (*Poisson Arrivals See Time Averages*) (voir Wolff [40]) implique que la proportion des appels bloqués est :

$$B(\mu) = \sum_{k=s}^{\infty} \pi_k(\mu) = \pi_s(\mu) \frac{s\mu}{s\mu - \lambda^d} \tag{4.7}$$

où

$$\pi_0(\mu) = \left(\sum_{v=0}^{s-1} \frac{(\lambda^p + \lambda^d)^v}{v!\mu^v} + \frac{(\lambda^p + \lambda^d)^s}{s!\mu^s} \frac{s\mu}{s\mu - \lambda^d} \right)^{-1}. \tag{4.8}$$

Contrairement aux taux d'arrivée agrégés λ^p et λ^d, il faut agréger les temps moyens de service pour les appels *pertes* et *délais* (μ^p et μ^d) pour obtenir le temps de service moyen μ du groupe pour l'ensemble des appels. L'agrégation est calculée en fonction du ratio ($w(\mu)$) du nombre d'appels *pertes* ou *délais* servis par ce groupe. Les appels de type *perte* bloqués sont ignorés, car ils seront servis par d'autres groupes.

$$w(\mu) = \frac{\lambda^d}{\lambda^d + \lambda^p(1 - B(\mu))} \tag{4.9}$$

Nous voulons donc trouver un μ en fonction du ratio des appels servis :

$$\frac{1}{\mu} = \frac{w(\mu)}{\mu^d} + \frac{1 - w(\mu)}{\mu^p}.$$

Ceci peut-être résolu par des méthodes de recherche de racines sur la fonction $h(\mu)$:

$$h(\mu) = \frac{w(\mu)}{\mu^d} + \frac{1 - w(\mu)}{\mu^p} - \frac{1}{\mu}. \tag{4.10}$$

Supposant que $\lambda^d > 0, \mu \geq 0$ et sachant que $B(\mu)$ est une probabilité d'une distribution continue, nous avons que $w(\mu) \in (0,1]$. Dans le cas sans abandon, si $\lambda^d \geq (s\mu^d)$ alors le système est instable et le calcul s'arrête. Supposant que le système soit stable et soit $\bar{\mu}^d = \max(\lambda^d/s, \mu^d)$, il existe une racine de $h(\mu)$ dans $[\min(\bar{\mu}^d, \mu^p), \max(\bar{\mu}^d, \mu^p)]$ (voir Avramidis et al. [4]).

Avec la racine μ^* de l'équation (4.10), la probabilité que la durée d'attente W d'un appel soit supérieure à τ est calculée comme avec une file $M/M/s$ (voir Cooper [16]) :

$$D(\mu^*, \tau) = \mathbb{P}[W > \tau] = \mathbb{P}[W > 0]\mathbb{P}[W > \tau | W > 0] = B(\mu^*)e^{-\tau(s\mu^* - \lambda^d)}. \tag{4.11}$$

4.3.2.2 Cas avec abandons

Dans le cas avec abandons, les appels de type $i \in \mathscr{D}_j$ en attente peuvent abandonner avec un taux $\eta_i > 0$ et la capacité de la file d'attente est limitée à c, donc il ne peut y avoir plus de $s+c$ appels dans le système.

La propriété PASTA (*Poisson Arrivals See Time Averages*) (voir Wolff [40]) implique que la fraction des appels bloqués est :

$$B_a(\mu) = \sum_{k=s}^{s+c} \pi_k(\mu). \tag{4.12}$$

Supposant que les clients du modèle $M/M/s$ ont des durées de patience qui suivent une loi exponentielle de moyenne $1/\eta$, Riordan [35] donne la probabilité que le temps d'attente W soit supérieur à τ conditionnellement au nombre X d'appels dans le système :

$$p_k(\mu, \tau) = \mathbb{P}[W > \tau | X = s+k] = \xi^{\phi+1} \sum_{z=0}^{k} \frac{(\phi)_z (1-\xi)^z}{z!} \tag{4.13}$$

où $\phi = s\mu/\eta$, $\xi = e^{-\eta\tau}$ avec les symboles de Pochhammer : $(\phi)_0 = 1$ et $(\phi)_z = (\phi)(\phi+1)\cdots(\phi+z-1)$ pour $z \geq 1$.

Le calcul du taux de service agrégé μ est similaire à l'équation (4.10) ; en soustrayant les abandons, car aucun temps de service n'a été dépensé pour les appels ayant

abandonné. Le taux d'arrivée moyen $\tilde{\lambda}^d$ pour les appels desservis du type *délai* est :

$$\tilde{\lambda}^d = \lambda^d[1 - \pi_{s+c}(\mu)] - \eta \sum_{k=1}^{c-1} k\pi_{s+k}(\mu). \tag{4.14}$$

Nous cherchons la racine μ^* de l'équation :

$$h_a(\mu) = \frac{w_a(\mu)}{\mu^d} + \frac{1 - w_a(\mu)}{\mu^p} - \frac{1}{\mu} \tag{4.15}$$

où

$$w_a(\mu) = \frac{\tilde{\lambda}^d}{\tilde{\lambda}^d + \lambda^p(1 - B_a(\mu))}. \tag{4.16}$$

Supposant que $\tilde{\lambda}_d > 0$, alors $w_a(\mu) \in (0,1]$. Supposant que $B_a(\mu)$ est une fonction continue, il existe une racine μ^* à la fonction $h_a(\mu)$ dans l'intervalle $[\min(\mu^d, \mu^p), \max(\mu^d, \mu^p)]$.

Avec la propriété PASTA et l'équation (4.13), la probabilité que le temps d'attente W d'un appel soit supérieur à τ est :

$$D_a(\mu^*, \tau) = \mathbb{P}[W > \tau] = \sum_{k=0}^{c-1} p_k(\mu^*, \tau)\pi_{s+k}(\mu^*) + \pi_{s+c}(\mu^*). \tag{4.17}$$

La fraction d'abandon des appels du type *délai* est calculée par le ratio du taux d'arrivée moyen des appels servis sur le taux d'arrivée total :

$$A(\mu) = 1 - \frac{\tilde{\lambda}^d}{\lambda^d}. \tag{4.18}$$

Puisque les appels du type *perte* $i \in \mathscr{P}_j$ sont envoyés vers un autre groupe d'agents lorsqu'ils ne peuvent être servis immédiatement par le groupe j, les probabilités d'abandon et de temps d'attente ne sont pas analysées avec ce groupe.

4.3.3 Analyse des groupes de type *pur délai*

L'analyse pour ces groupes est identique à l'analyse des groupes de type *perte et délai* (section 4.3.2), à l'exception que $\lambda^p = 0$, $\mu^p = 0$ et $\mu^* = \mu^d$.

4.3.4 Approximation du centre d'appels

Notre modèle d'approximation étant une extension de Koole et Talim [31], l'analyse du centre d'appels se fait par groupe d'agents. Avant de procéder aux calculs des formules de CTMC, il faut agréger certains paramètres.

Soit $\mathscr{R}_i$ l'ordre de priorité du type d'appel i aux groupes, défini dans la section 4.2, nous définissons la fonction $p(i,j)$ qui retourne le groupe précédant le groupe j dans l'ordre de priorité $\mathscr{R}_i$, nul si aucun. Dans le calcul du débordement de flot, les appels bloqués au groupe $p(i,j)$ deviennent les appels entrants au groupe j. Soit $\gamma_{i,j}$ le taux d'appels de type i entrant au groupe j et B_j la proportion des appels bloqués au groupe j, nous avons (nous ne distinguons pas les cas avec et sans abandon pour généraliser les notations) :

$$\gamma_{i,j} = \begin{cases} \lambda_i, & \text{si } r_i(1) = j \\ \gamma_{i,p(i,j)} B_{p(i,j)}, & \text{si } r_i(1) \neq j \text{ et } j \in \mathscr{R}_i. \end{cases}$$

Lors de l'analyse du processus de naissance et de mort du groupe d'agents j, il faut agréger de manière heuristique les données suivantes : $\lambda_j^d, \lambda_j^p, \mu_j^d, \mu_j^p, \eta_j^d$ et τ_j^d. Nous le faisons comme suit.

Si le groupe d'agents j possède un rôle de *perte* pour au moins un type d'appel ($j \in \mathscr{M}_p \cup \mathscr{M}_{pd}$), nous sommons les taux d'arrivée de type *perte* à ce groupe et la durée moyenne de service de type *perte* est agrégée proportionnellement aux taux d'arrivée (agréger selon les durées donne une moyenne plus conservatrice que si nous prenions la moyenne des taux de service) :

$$\lambda_j^p = \sum_{i \in \mathscr{P}_j} \gamma_{i,j} \qquad\qquad \mu_j^p = \left(\sum_{i \in \mathscr{P}_j} \frac{\gamma_{i,j}}{\lambda_j^p \mu_{i,j}} \right)^{-1}.$$

Si le groupe d'agents j possède un rôle de *délai* pour au moins un type d'appel ($j \in \mathscr{M}_d \cup \mathscr{M}_{pd}$), le taux d'arrivée de type *délai* et la durée moyenne de service agrégée de type *délai* sont calculés de la même manière que le cas *perte*. Le temps d'attente acceptable agrégé est également proportionnel aux taux d'arrivée. La patience est agrégée

selon les taux d'abandon au lieu des temps de patience, car avec ces derniers, les petits taux d'abandon auraient beaucoup d'importance et le système de file d'attente aurait un faible taux d'abandon. Prenons un exemple avec 2 types d'appels ayant des taux d'arrivée égaux et les taux d'abandon $\eta_1 = 120$ et $\eta_2 = 2$, soit un type d'appel avec une faible patience et l'autre, une grande patience. Supposons qu'il y a un même nombre d'appels en attente pour les 2 types, un taux agrégé de 61 (agrégeant selon les taux) représenterait mieux qu'un taux de 4 (agrégeant selon les durées de patience). Nous avons :

$$\lambda_j^d = \sum_{i \in \mathscr{D}_j} \gamma_{i,j} \qquad \mu_j^d = \left(\sum_{i \in \mathscr{D}_j} \frac{\gamma_{i,j}}{\lambda_j^d \mu_{i,j}} \right)^{-1}$$
$$\eta_j^d = \sum_{i \in \mathscr{D}_j} \frac{\gamma_{i,j} \eta_i^d}{\lambda_j^d} \qquad \tau_j^d = \sum_{i \in \mathscr{D}_j} \frac{\gamma_{i,j} \tau_i}{\lambda_j^d}.$$

Le SL du type d'appel i est calculé à l'aide du ratio $D_{r_i(m_i)}$ des appels ayant attendu plus de $\tau^d_{r_i(m_i)}$ au dernier groupe d'agents de la liste $\mathscr{R}_i$, puisque nous supposons à l'aide du modèle de perte que la proportion des appels i servis par les groupes $r_i(j), j < s_i$ ont été servis immédiatement :

$$g_i(\mathbf{x}) = 1 - \frac{\gamma_{i,r_i(m_i)}}{\lambda_i} D_{r_i(m_i)}. \tag{4.19}$$

Le SL global du centre d'appels est calculé en agrégeant les niveaux de service de chaque type d'appel :

$$g(\mathbf{x}) = \frac{\sum_{i=1}^{n} \lambda_i g_i(\mathbf{x})}{\sum_{i=1}^{n} \lambda_i}. \tag{4.20}$$

Le taux d'appels de type i servis par le groupe j, dénoté par $f_{i,j}$, correspond à la fraction non-bloquée si j n'est pas le dernier groupe de la liste et correspond à la fraction n'ayant pas abandonné si j est en effet le dernier groupe :

$$f_{i,j} = \gamma_{i,j}(1 - B_j), \quad \text{si } i \in \mathscr{P}_j \tag{4.21}$$
$$f_{i,j} = \gamma_{i,j}(1 - A_j), \quad \text{si } i \in \mathscr{D}_j. \tag{4.22}$$

Le taux d'abandon du type d'appel i correspond au nombre d'appels ayant abandonné au dernier groupe de la liste $\mathscr{R}_i$:

$$a_i = \gamma_{i,r_i(m_i)} A_{r_i(m_i)} = \gamma_{i,r_i(m_i)} - f_{i,r_i(m_i)}. \tag{4.23}$$

4.3.5 Implémentation de l'approximation perte et délai

Dans l'implémentation de l'approximation, nous avons conçu un algorithme itératif qui s'arrête lorsque les probabilités de blocage B_j convergent jusqu'à une différence inférieure au paramètre ε fourni par l'utilisateur. Dans le cas avec abandon, l'utilisateur peut spécifier la capacité c identique pour tous les groupes ou employer le choix automatique : $c_j = \max\{\delta\sqrt{x_j}, 10\}$, où δ est un paramètre fourni par l'usager. Au lieu de donner un grand c_j pour chaque groupe, le choix automatique permet d'économiser sur le temps de calcul. L'idée est que le nombre stationnaire de clients dans un système $M/M/\infty$ soit une distribution de Poisson dont l'écart-type correspond à la racine de la moyenne. Nous mettons un minimum de 10 en considération des petits centres d'appels.

Les données primordiales de cette approximation sont les probabilités de blocage B_j de chaque groupe d'agents. Avec B_j, nous obtenons les taux d'arrivée de chaque type d'appel à chaque groupe d'agents $\gamma_{i,j}$ et ceux-ci permettent d'agréger les autres paramètres afin d'évaluer les systèmes $M/M/s$ représentant les groupes d'agents. La figure 4.3 présente l'algorithme d'approximation par *itération des valeurs* (IV). Premièrement, les probabilités de blocage B_j sont initialisées à 0 pour chaque groupe d'agents $j \in \mathscr{M}$. À l'exception des taux d'arrivée exogène λ_i, les taux d'arrivée $\gamma_{i,j}$ entre chaque groupe sont initialisés à 0, car ils sont inconnus. Puisque les données sont agrégées en fonction des taux d'arrivée (voir section 4.3.4), les $\gamma_{i,j}$ sont mis à jour au début de chaque itération, suivis des autres paramètres. Chaque groupe est évalué (dans un ordre quelconque) pour actualiser sa probabilité de blocage B_j. L'algorithme s'arrête lorsque les erreurs de convergence de toutes les B_j sont inférieures à ε. Par la suite, il ne reste qu'à déterminer les niveaux de service.

Plus les listes de routage $\mathscr{R}_i$ seront longues, plus il faudra d'itérations. Pour donner un exemple simple, supposons un centre d'appels composé d'un type d'appel, m grou-

Paramètres de l'algorithme :
Vecteur du nombre d'agents par groupe : $\mathbf{x}$
Niveau de tolérance : ε
Nombre maximal d'itérations : κ
Contrôle de la capacité de la file d'attente : δ

1. Initialisation des variables :
   $v = 0$
   $\gamma_{i,j}^{(v)} = 0, \forall i \in \mathscr{N}, \forall j \in \mathscr{M}$
   $\gamma_{i,j}^{(v)} = \lambda_i$, si $r_i(1) = j$
   $B_j^{(v)} = 0, \forall j \in \mathscr{M}$
2. Mise à jour des taux d'arrivée :
   $v = v + 1$
   $\gamma_{i,j}^{(v)} = \gamma_{i,p(i,j)}^{(v-1)} B_{p(i,j)}^{(v-1)}, \forall i \in \mathscr{N}, \forall j \in \mathscr{M}$
3. Agréger les données pour chaque groupe d'agents $\forall j \in \mathscr{M}$:
   Si $j \in \mathscr{M}_p \cup \mathscr{M}_{pd}$: $\lambda_j^{p(v)}$ et $\mu_j^{p(v)}$
   Si $j \in \mathscr{M}_d \cup \mathscr{M}_{pd}$: $\lambda_j^{d(v)}, \mu_j^{d(v)}, \eta_j^{d(v)}$ et $\tau_j^{d(v)}$
4. Calculer les probabilités de blocage du groupe $\forall j \in \mathscr{M}$:
   $B_j^{(v)} = B(\mu_j^*)$ ou $B_a(\mu_j^*)$ si avec abandon,
5. Vérification des critères d'arrêt :
   Si $\forall j \in \mathscr{M}$: $|B_j^{(v)} - B_j^{(v-1)}| < \varepsilon$, alors aller à l'étape 6.
   Si $v \geq \kappa$, alors aller à l'étape 6.
   Autrement, retourner à l'étape 2.
6. Calculer les niveaux de service :
   $D_j = D(\mu_j^*, \tau_j^{(v)})$ ou $D_a(\mu_j^*, \tau_j^{(v)})$ si avec abandon, $\forall j \in \mathscr{M}$
   $g_i(\mathbf{x}) = 1 - \dfrac{\gamma_{i,r_i(m_i)}^{(v)}}{\lambda_i} D_{r_i(m_i)}, \forall i \in \mathscr{N}$
   $g(\mathbf{x}) = \frac{\sum_{i=1}^{n} \lambda_i g_i(\mathbf{x})}{\sum_{i=1}^{n} \lambda_i}$.

Figure 4.3 – Algorithme d'approximation perte et délai par itération des valeurs.

pes d'agents et un ordre de priorité $\mathscr{R} = \{r(1), \ldots, r(m)\}$. Supposons que l'algorithme évalue les agents suivant la liste $\mathscr{R}$, il faudrait au minimum m itérations ($m+1$ si nous comptons l'itération pour valider la convergence), parce qu'un seul B_j est mis à jour à chaque itération.

L'itération de Gauss-Seidel (GS), une variante de l'itération des valeurs, est une méthode bien connue utilisée dans le but d'accélérer la convergence. Une implémentation utilisant GS est facilement applicable en modifiant la méthode d'itération des valeurs pour toujours utiliser la plus récente valeur de chaque variable (en enlevant l'indice d'itération k) :

$$\gamma_{i,j} = \gamma_{i,p(i,j)} B_{p(i,j)}.$$

Ainsi, il ne suffirait que d'une itération (et une deuxième pour valider la convergence) dans l'exemple précédent. Cette économie peut se généraliser pour des exemples de centres d'appels ayant plusieurs types d'appels sans croisement de routage entre les types d'appels. Dans le cas où il y a un croisement dans l'ordre de priorité du routage, plusieurs itérations sont toutefois requises. Néanmoins, d'après nos expériences, l'implémentation avec GS converge plus rapidement. Par contre, différents ordres d'évaluation des groupes peuvent donner des résultats légèrement différents selon les critères d'arrêt et de convergence utilisés.

4.3.6 Ordonnancement de l'évaluation des groupes d'agents

Lorsqu'un centre d'appels ne possède aucun croisement dans le routage, l'algorithme n'a besoin que d'une seule itération. Dans le cas où il existe des croisements (appelés cycles), nous pouvons réduire le nombre d'itérations en identifiant ces cycles à l'aide d'un algorithme d'ordonnancement présenté dans cette section.

Représentons le routage des appels aux groupes d'agents par un graphe acyclique orienté ressemblant à une arborescence, mais où chaque sommet peut avoir plusieurs parents. Les groupes d'agents sont représentés par les sommets et les flots des appels du type i par les arcs orientés suivant l'ordre de priorité $\mathscr{R}_i$. Nous ajoutons un sommet racine représentant l'entrée exogène de tous les appels au centre et ajoutons un arc de

ce sommet vers tous les groupes à la tête d'au moins une liste de priorité, soit $r_i(1), i \in \mathscr{N}$. Le niveau d'un sommet dans une arborescence représente le nombre de générations séparant ce sommet de la racine. Soit $\mathscr{S}_z$ l'ensemble des groupes appartenant au niveau z, avec la racine au niveau 0, les groupes dans $\mathscr{S}_z$ ne dépendent aucunement des groupes de niveaux $> z$. Ainsi, un ordonnancement possible est d'évaluer les groupes d'agents par niveau en commençant par $\mathscr{S}_1$.

Par contre, lorsque le routage contient un cycle, le graphe n'est plus acyclique. Pour adapter l'algorithme à cette situation, nous changeons la définition d'un sommet pour un ensemble de groupes faisant partie d'un même cycle ou un seul groupe. Avec ce changement, nous maintenons la structure d'une graphe orienté acyclique. Il reste qu'il faut itérer (par IV ou GS) jusqu'à convergence pour les sommets représentant des cycles de plusieurs groupes. Toutefois, le nombre total d'itérations est réduit en itérant seulement le nombre de fois nécessaire par cycle au lieu de tous les groupes.

La figure 4.4 montre un exemple simple d'un centre d'appels avec 3 types d'appels et 7 groupes d'agents où le routage possède un cycle formé par les groupes 4, 5 et 6, identifié par un rectangle pointillé. L'ordre de priorité de chaque type d'appel est représenté par un chemin partant de la racine jusqu'au dernier groupe identifié par une flèche blanche. Pour cet exemple, un ordonnancement possible est : $(1,2,3,\{4,5,6\},7)$.

La figure 4.5 présente un exemple un peu plus complexe d'un centre d'appels avec 4 types d'appels, 11 groupes d'agents et un routage possédant 3 cycles. Cet exemple montre que les cycles peuvent aussi être sur des niveaux différents dans l'arborescence. Un ordonnancement possible est : $(1,2,\{3,4\},\{5,6,7,8\},9,\{10,11\})$.

Pour faire une analogie avec la théorie des graphes (voir Aho et al. [1]), un cycle est défini comme étant une composante fortement connexe. Les sommets d'un graphe orienté $G(V,E)$ peuvent être partitionnés en des ensembles V_j tels que deux sommets $w \in V_j$ et $v \in V_j$ si et seulement s'il existe un chemin menant de w à v et vice-versa. Les auteurs donnent un algorithme (Algorithme 5.4 [1]) pour partitionner V en des ensembles V_j avec un ordre de complexité de $O(\max(m+1,e))$ où $e = \sum_{i=1}^{n} s_i$, soit le maximum entre le nombre de sommets et d'arcs selon la définition du graphe original (où chaque sommet représente un groupe).

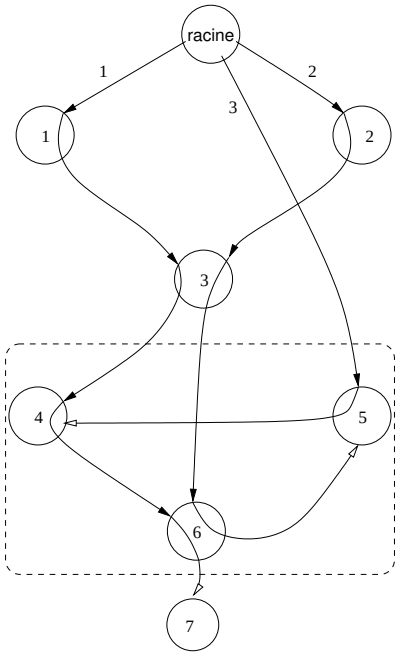

Figure 4.4 – Exemple d'un graphe orienté acyclique pour un centre d'appels avec 3 types d'appels, 7 groupes d'agents et un routage contenant un cycle.

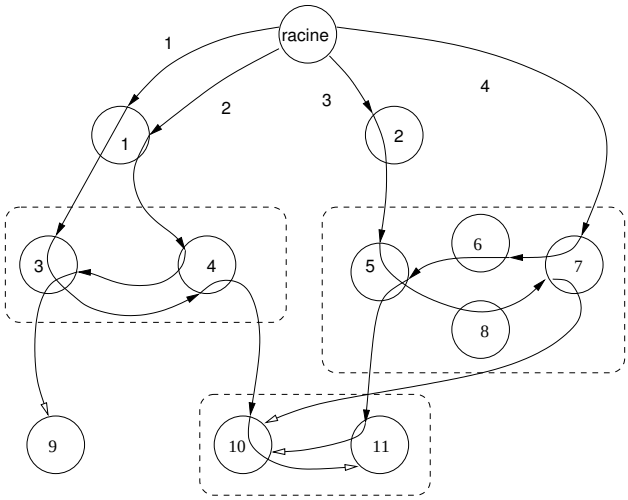

Figure 4.5 – Exemple d'un graphe orienté acyclique pour un centre d'appels avec 4 types d'appels, 11 groupes d'agents et un routage contenant 3 cycles.

À partir des partitions V_j, nous avons conçu un algorithme (voir la figure 4.3.6) pour trouver un ordonnancement d'évaluation des partitions dont le temps d'exécution est dans $O(e)$. Les duplications d'arcs (entre 2 paires de sommets identiques) sont éliminées. Cet algorithme est une adaptation de l'algorithme de recherche en largeur (*breadth-first search*) dans le cas où nous ne connaissons pas le niveau des partitions dans l'arborescence, sauf celui de la racine. Certaines structures de données sont utilisées :

— $\mathscr{L}$ est la liste d'ordonnancement des V_j qui sera passée à l'algorithme d'approximation perte et délai.

— $\mathscr{C}$ est un vecteur tel que l'élément $\mathscr{C}(j)$ contient le nombre d'arcs arrivant au sommet V_j. Les arcs dupliqués ont été éliminés auparavant.

— $\mathscr{Q}$ est une queue qui contient les sommets à examiner (telle qu'utilisée dans l'algorithme de recherche en largeur). $\mathscr{Q}(1)$ représente l'élément à la tête de la queue.

— $\mathscr{D}$ est un vecteur d'ensembles de sommets tel que l'élément $\mathscr{D}(j)$ contient l'ensemble des sommets destinations des arcs sortant du sommet V_j.

Nous disons qu'un sommet est approuvé lorsque ses parents sont présents dans $\mathscr{L}$, avec la racine dans $\mathscr{L}$ au départ. Le vecteur $\mathscr{C}(j)$ contient le nombre de sommets parents restant à approuver avant que le sommet j puisse être approuvé à son tour. L'algorithme commence avec la racine de l'arborescence et approuve un sommet lorsque $\mathscr{C}(j) = 0$. Lorsqu'un sommet j a été approuvé, les $\mathscr{C}(i), \forall i \in \mathscr{D}(j)$, sont décrémentés de 1 puisqu'un sommet parent vient d'être approuvé. Étant donné que le graphe est acyclique et connexe, en partant de la racine, il est assuré que l'algorithme approuvera tous les sommets selon un certain ordre et cet ordre sera l'ordonnancement fourni à l'algorithme d'approximation perte et délai.

D'après nos expériences, l'algorithme d'approximation LD est plus rapide avec l'ordonnancement pour les centres d'appels sans croisement de routage. Dans le cas où le routage possède des cycles, il n'y a pas de gagnant clair entre l'utilisation de GS avec ou sans ordonnancement, car la vitesse de convergence est dépendante de l'ordre d'évaluation des groupes d'agents. Cependant, il reste que la méthode GS est plus rapide que celle par IV. Notons que la vitesse de calcul de LD est négligeable et que la différence

```
liste fonction Ordonner(𝒞, 𝒟)
    ℒ = ∅
    𝒬 ← {racine}
    tant que (𝒬 ≠ ∅ )
        pour j ∈ 𝒟(𝒬(1))
            𝒞(j) ← 𝒞(j) − 1
            si (𝒞(j) = 0)
                ajouter j à 𝒬
            fin si
        fin pour
        ajouter 𝒬(1) à ℒ
        retirer 𝒬(1) de 𝒬
    fin tant que
    retirer ℒ(1) (retirer la racine)
    retourner ℒ
fin fonction
```

Figure 4.6 – La fonction *Ordonner* retourne la liste ordonnée $\mathscr{L}$ des sommets à évaluer par l'algorithme d'approximation LD.

entre GS et IV est insignifiante, à l'exception de centres d'appels de grande dimension où GS permet une légère économie sur le temps de calcul.

4.4 Méthodes de recherche randomisée

Nous avons bâti un algorithme de recherche randomisée, ou *Randomized Search* (RS), pour l'affectation des agents dans un centre d'appels pour une période, voir Avramidis et al. [4]. L'algorithme comprend trois étapes : l'initialisation, la recherche randomisée par voisinage (phase-1) et la correction locale par simulation (phase-2). Pour des raisons de temps d'exécution, la simulation est remplacée par LD comme évaluateur de performances du centre d'appels. Dans cette section, nous disons qu'une solution est réalisable selon l'évaluateur utilisé. L'approximation est employée pour les deux premières étapes, puis la simulation à l'étape finale. L'algorithme commence avec une solution réalisable, il enlève des agents ou déplace des agents d'un groupe vers un autre groupe moins dispendieux jusqu'à un optimum local et la solution finale est corrigée à l'aide

d'une recherche locale demandant peu de simulations.

4.4.1 Initialisation

La procédure d'initialisation cherche une solution réalisable puisque la recherche randomisée ne s'effectue que dans le domaine des solutions réalisables. Bien qu'il soit trivial de trouver une solution réalisable simplement en sur-allouant le nombre d'agents dans chaque groupe, d'après nos expériences, il est important d'avoir une bonne solution de départ, car elle influence sur la qualité de la solution finale. Nous avons alors besoin d'une bonne procédure d'initialisation. Dans cette section, nous présentons une initialisation donnant de bons résultats alors que différentes autres méthodes testées seront présentées dans le chapitre 5.

L'idée de cette méthode d'initialisation est de contrôler l'utilisation des agents spécialistes versus les généralistes. Premièrement, pour chaque type d'appel $i \in \mathscr{N}$, le flot d'appels est réparti parmi les groupes $j \in \mathscr{R}_i$, tel qu'une fraction β du flot soit allouée au groupe le moins dispendieux qui est généralement un spécialiste et que la fraction $(1-\beta)$ soit divisée également parmi les groupes restants.

Pour calculer le nombre initial d'agents d'un groupe, les flots d'appels alloués à ce groupe sont agrégés parmi les types $\mathscr{K}_j$. Le nombre d'agents correspond au nombre minimal d'agents nécessaires pour avoir un SL égal ou supérieur à un paramètre ξ. Les groupes sont considérés indépendants les uns des autres et leurs performances sont évaluées selon un système $M/M/s$ (un groupe de type *pur délai*, voir section 4.3.3). D'après nos expériences, l'affectation obtenue était quasiment toujours réalisable avec $\xi \geq l$, où l est le seuil du SL global. Un petit ξ risque de donner une affectation irréalisable et nous avons conçu deux procédures pour corriger la solution. La première procédure consiste à incrémenter le groupe qui possède le plus grand ratio de temps occupé par coût, $\arg\max_{j\in\mathscr{M}} \left\{ \left(\sum_{i\in\mathscr{K}_j} f_{i,j}/\mu_{i,j} \right) / (c_j x_j) \right\}$ (soit celui qui travaille le plus de temps par coût), si le seuil global n'est pas satisfait. Si le seuil du SL d'un type d'appel i n'est

pas satisfait, alors le groupe le plus dédié à ces appels :

$$j^* = \arg\max_{j \in \mathscr{R}_i} \left\{ \frac{f_{i,j}/\mu_{i,j}}{\sum_{k \in \mathscr{K}_j} f_{k,j}/\mu_{k,j}} \right\}$$

est incrémenté. La deuxième méthode incrémente tous les groupes d'agents en augmentant ξ par une proportion ζ déterminée par l'utilisateur.

4.4.2 Recherche randomisée par voisinage

L'étape de la recherche randomisée, la phase-1 de l'optimisation, débute avec une solution initiale réalisable. À chaque itération, il y a deux mouvements de taille q possibles applicables à la solution courante $\mathbf{x}^{(v)}$ avec $\mathscr{C}(q) = \{j : j \in \mathscr{M}, x_j^{(v)} \geq q\}$ étant l'ensemble des groupes ayant au moins q agents :

Retirer. Réduit le coût en retirant q agents d'un même groupe tout en gardant la réalisabilité. Le voisinage est : $\mathscr{V}_R(\mathbf{x}^{(v)}, q) = \{\mathbf{y} : \mathbf{y} = \mathbf{x}^{(v)} - q\mathbf{e}_j, j \in \mathscr{C}(q)\}$.

Déplacer. Réduit le coût en déplaçant q agents d'un groupe p vers un autre moins dispendieux tout en gardant la réalisabilité.
Le voisinage est : $\mathscr{V}_D(\mathbf{x}^{(v)}, q) = \cup_{p \in \mathscr{C}(q)} \mathscr{V}_D(\mathbf{x}^{(v)}, q, p)$ où $\mathscr{V}_D(\mathbf{x}^{(v)}, q, p) = \{\mathbf{y} : \mathbf{y} = \mathbf{x}^{(v)} - q\mathbf{e}_p + q\mathbf{e}_j, j \in \mathscr{P}(p)\}$ et $\mathscr{P}(p) = \{j : c_j < c_p, j \in \mathscr{M}\}$ est l'ensemble des groupes moins dispendieux que p.

Étant donné que l'algorithme ne recherche que dans le domaine réalisable, les voisinages $\mathscr{V}_R^r \subseteq \mathscr{V}_R$ et $\mathscr{V}_D^r \subseteq \mathscr{V}_D$ ne contiennent que les affectations réalisables. La solution choisie est celle qui réduit le moins le SL par coût sauvé :

$$\mathbf{x}^{(v+1)} = \arg\min_{\mathbf{y} \in \mathscr{V}_R^r \cup \mathscr{V}_D^r} \left\{ \frac{g(\mathbf{x}^{(v)}) - g(\mathbf{y})}{\mathbf{c}^t\mathbf{x}^{(v)} - \mathbf{c}^t\mathbf{y}} \right\}. \tag{4.24}$$

La recherche s'arrête lorsqu'il ne reste plus de mouvement possible pour $q = 1$, autrement dit $\mathscr{V}_R^r(\mathbf{x}^{(v)}, 1)$ et $\mathscr{V}_D^r(\mathbf{x}^{(v)}, 1)$ sont vides.

Cette recherche est monotone décroissante par rapport au coût des agents puisque les voisinages $\mathscr{V}_R^r$ et $\mathscr{V}_D^r$ ne contiennent que des solutions moins coûteuses (soient moins

d'agents ou des agents moins coûteux) que la solution courante.

Proposition 4.1. La recherche par voisinage, phase-1, termine après un nombre fini d'itérations.

Preuve. Puisque le coût décroît strictement à chaque itération, une solution ne peut être retenue qu'au plus une seule fois. De plus, le domaine des solutions réalisables est borné à un coût $\mathbf{c}^{\mathbf{t}}\mathbf{x} > 0$ tel qu'il faut au moins un agent, alors le nombre d'itérations doit être fini.

Dans notre implémentation, la taille du mouvement q est générée selon une loi exponentielle avec une moyenne égale à la médiane du vecteur $\mathbf{x}^{(v)}$. Cette règle a bien performé dans nos expérimentations. Nous avons essayé d'autres règles telles que fixer $q = 1$ ou suivant une loi uniforme. Il y a peu de sensibilité entre les différentes règles utilisées sur les solutions finales. Nous avons observé que les règles retournant plus fréquemment un petit q économisaient sur le nombre d'appels à l'évaluateur, car un q élevé risque souvent de donner des voisinages réalisables vides.

La taille du voisinage $\mathscr{V}_R(\mathbf{x}^{(v)}, q)$ est dans $O(m)$ et $\mathscr{V}_D(\mathbf{x}^{(v)}, q)$ est dans $O(m^2)$. À cause de la taille considérable de $\mathscr{V}_D(\mathbf{x}^{(v)}, q)$, l'algorithme n'évalue pas tout le voisinage. Il choisit au hasard un groupe pivot $p \in \mathscr{C}(q)$ et si le voisinage $\mathscr{V}_D^r(\mathbf{x}^{(v)}, q, p)$ n'est pas vide, alors la procédure choisira la prochaine solution $\mathbf{x}^{(v+1)}$ parmi ces voisins. Dans le cas où $\mathscr{V}_D^r(\mathbf{x}^{(v)}, q, p)$ est vide, nous ne voulons pas que l'algorithme examine une autre fois ce même voisinage pour $\mathbf{x}^{(v)}$. Pour ce faire, notre implémentation gère un vecteur $\mathbf{q}^*$ de taille m tel que q_p^* contient la plus petite taille q ayant généré un voisinage réalisable vide avec le groupe p comme pivot. Avec l'hypothèse que les fonctions $g_i(\mathbf{x})$ et $g(\mathbf{x})$ sont monotones par rapport à q avec $\mathbf{x} = \mathbf{x}^{(v)} - q\mathbf{e}_p + q\mathbf{e}_j, j \in \mathscr{P}(p)$, alors les voisinages sont aussi vides pour tout $q > q_p^*$.

D'après nos expériences, il est préférable de mixer les procédures *Retirer* et *Déplacer*. Comparativement au déplacement des agents, l'action de retirer des agents provoque généralement un impact plus important autant sur la réduction du coût que des SL. Nous avons obtenu de bonnes performances en donnant la priorité à la procédure *Retirer*. Pour un q donné, si $\mathscr{V}_R^r(\mathbf{x}^{(v)}, q)$ n'est pas vide, alors $\mathbf{x}^{(v+1)}$ sera choisie dans ce voisinage,

autrement l'algorithme cherche dans $\mathscr{V}_D^r(\mathbf{x}^{(v)}, q)$. Les premières itérations sont dominées par les mouvements *Retirer*, la recherche s'approche rapidement de la frontière de réalisabilité où il devient difficile de réduire le nombre d'agents, puis la recherche est dominée par la suite par les mouvements *Déplacer*. Avec le mixage des deux procédures, il arrive que l'algorithme puisse retirer des agents après avoir déplacé des agents.

En donnant la priorité à *Retirer*, d'après nos expériences, les solutions finales reflètent souvent la structure des solutions initiales. Notons que si q est une constante, alors la procédure *Retirer* sera déterministe. La diversité des solutions est plutôt donnée par la procédure *Déplacer*, mais son voisinage réalisable est souvent restreint par le manque d'agents à cause des mouvements *Retirer*.

Dans le cas où les deux procédures sont à priorité égale, la recherche commence généralement avec les mouvements *Déplacer* et dans certaines expériences, elle peut modifier entièrement la structure de la solution initiale, supposant qu'il y ait suffisamment d'agents. Avec les priorités égales, l'algorithme était plus robuste face aux mauvaises initialisations, cependant la correction avec la procédure *Déplacer* requiert énormément d'itérations, ce qui allonge beaucoup le temps d'exécution dans les grands problèmes. Cependant, nous avons trouvé que l'algorithme performait tout aussi bien et plus rapidement en donnant la priorité à *Retirer* avec une bonne méthode d'initialisation telle que présentée dans la section 4.4.1.

Vu l'influence de l'initialisation et la rapidité de la recherche avec LD comme évaluateur, nous suggérons d'effectuer plusieurs départs (*multi-starts*) sur le paramètre β qui contrôle la proportion des agents spécialistes et généralistes.

Il serait intéressant et probablement avantageux d'utiliser des méthodes qui n'imposent pas une amélioration du coût à chaque étape, comme la méthode tabu et le recuit simulé.

4.4.3 Correction et amélioration par simulation

Étant donné que la recherche randomisée requiert des évaluations rapides pour être efficace dans la pratique, l'approximation LD représente le choix approprié, mais elle donne des erreurs analytiques qui peuvent être substantielles lorsque comparées avec la

simulation. Par contre, même en remplaçant l'approximation par de très courtes simulations, les temps d'exécutions sont trop élevés en pratique et le bruit de la simulation amène l'algorithme à déclarer plus facilement des solutions faussement irréalisables (il ne suffit qu'une contrainte soit faussement violée). Cette étape sert à corriger la solution finale à l'aide de la simulation par deux procédures déterministes. À cause du temps d'exécution élevé de la simulation, nous avons choisi des algorithmes gloutons et au lieu d'évaluer les solutions voisines, les décisions sont calculées à partir des performances provenant des solutions courantes afin de limiter le nombre d'appels au simulateur.

Premièrement, il faut s'assurer que la solution soit réalisable selon la simulation. D'après nos expériences, il est plus efficace de rectifier les contraintes par type d'appel avant de corriger la contrainte du SL global puisqu'en améliorant le service d'un type d'appel, le SL global se trouve amélioré du même coup, alors que l'inverse n'est pas assuré. Si $\mathbf{x}^{(v)}$ est irréalisable, alors l'algorithme vérifie tout d'abord s'il y a des contraintes de SL par type d'appel violées. S'il existe un type d'appel $i^* = \arg\max_{i \in \mathscr{N}} \{l_i - g_i(\mathbf{x}^{(v)}) > 0\}$, alors la procédure incrémente le groupe qui a consacré la plus grande partie de son temps à répondre aux appels du type i^*. Le simulateur calcule le nombre moyen d'appels de chaque type servis par chaque groupe, soit $f_{i,j}$ où i est le type d'appel et j le groupe d'agents. Le temps moyen qu'un groupe j a passé à servir les appels du type i est calculé par : $f_{i,j}/\mu_{i,j}$. Puisque les calculs sont basés sur les performances d'une seule solution, les groupes n'ayant aucun agent sont automatiquement ignorés, car $f_{i,j} = 0$. Soit l'ensemble $\mathscr{R}'_{i^*} = \mathscr{R}_{i^*} \cap \{j : x_j^{(v)} \geq 1, j \in \mathscr{M}\}$ représentant les groupes ayant au moins un agent, le groupe j^* qui sera incrémenté est :

$$j^* = \arg\max_{j \in \mathscr{R}'_{i^*}} \frac{f_{i^*,j}/\mu_{i^*,j}}{\sum_{i \in \mathscr{K}_j} f_{i,j}/\mu_{i,j}}. \tag{4.25}$$

Dans le cas où seulement la contrainte globale est violée, l'algorithme incrémente le

groupe qui possède la meilleure productivité par coût :

$$j^* = \arg\max_{j \in \mathscr{M}} \frac{\sum_{i \in \mathscr{K}_j} f_{i,j}/\mu_{i,j}}{x_j^{(v)} c_j}. \tag{4.26}$$

La solution $\mathbf{x}^{(v)}$ est incrémentée d'un agent à chaque itération jusqu'à ce qu'elle devienne réalisable. Le nombre de simulations exécutées correspond au nombre d'incréments plus 1.

Dans le cas où LD sous-évalue la performance du centre d'appels (ou de certains types d'appels), certains agents pourraient être retirés pour économiser le coût tout en respectant les contraintes de SL. À partir de la solution réalisable obtenue par la simulation, cette procédure décrémente d'un agent à chaque itération jusqu'à ce qu'il n'y ait plus de solution voisine réalisable. Puisque la réduction est monotone, alors $x_j^{(w)} \leq x_j^{(v)}$ pour $w > v$. Si à une itération v, $\mathbf{x}^{(v)} - \mathbf{e}_j$ n'est pas réalisable, alors $\mathbf{x}^{(w)} - \mathbf{e}_j$ ne sera pas réalisable pour toutes les itérations ultérieures $w > v$, supposant la monotonicité de $g(\mathbf{x})$. Soit l'ensemble $\mathscr{L}$ initialisé à $\mathscr{L} = \{j : x_j^{(v)} \geq 1, j \in \mathscr{M}\}$ qui contient tous les groupes probables à décrémenter. Lorsqu'un groupe n'a plus d'agent, il est automatiquement retiré de $\mathscr{L}$. À chaque itération, l'algorithme calcule un facteur de dévouement en fonction du temps de travail consacré aux appels du type i par le groupe $j \in \mathscr{L}$:

$$d_{i,j} = \frac{f_{i,j}/\mu_{i,j}}{\sum_{k \in \mathscr{K}_j} f_{k,j}/\mu_{k,j}}. \tag{4.27}$$

Puisque la solution est réalisable, il existe des écarts non-négatifs sur les contraintes. Les groupes $j \in \mathscr{L}$ sont associés à ces écarts en les agrégeant en fonction de leur dévouement pour chaque type d'appel :

$$\chi_j = \sum_{i \in \mathscr{K}_j} d_{i,j}[g_i(\mathbf{x}^{(v)}) - l_i]. \tag{4.28}$$

Un coût est associé aux écarts par le produit $c_j \chi_j$, ainsi le groupe le plus coûteux sera choisi lorsque les écarts sont identiques. Les solutions sont évaluées par simulation en

suivant l'ordre décroissant des $c_j\chi_j$, $j \in \mathscr{L}$ et nous acceptons la première solution réalisable, puis nous passons à la prochaine itération. Lorsqu'une réduction d'un groupe j est irréalisable ou $x_j^{(v+1)} = 0$, alors ce groupe est retiré de $\mathscr{L}$. La procédure s'arrête lorsque $\mathscr{L}$ devient vide. Le nombre total d'appels au simulateur est borné supérieurement par le nombre de décrémentations plus m, le nombre de groupes.

Après cette étape, l'algorithme retourne la solution finale.

CHAPITRE 5

EXEMPLES NUMÉRIQUES

Pour étudier les performances de nos algorithmes d'optimisation, nous avons expérimenté nos logiciels sur différents exemples de centres d'appels. Nous débutons avec un centre d'appels simple (CCN) composé de 2 types d'appels et 2 groupes d'agents avec un modèle de routage de la forme de la lettre N. Par la suite, nous avons un exemple d'un centre d'appels de moyenne taille (CC1) considéré par l'entreprise Bell Canada ainsi qu'un grand centre d'appels (CC2) basé sur un centre réel de Bell Canada. Pour tous les exemples, le temps d'attente acceptable τ est de 20 secondes, globalement et par type d'appel, et le seuil minimal du SL global est $l = 0.80$. Les processus d'arrivée sont supposés Poisson et les lois de service et d'abandon sont exponentielles. Les taux de service sont dépendants uniquement du type d'appel et non du groupe d'agents. Tous les taux sont exprimés par heure.

Dans ce chapitre, nous emploierons les abréviations suivantes : LD pour l'approximation perte et délai (section 4.3), SIM pour la simulation, CP pour l'optimisation par coupes linéaires et simulations (chapitre 3), RS pour la recherche randomisée (section 4.4), RS_1 et RS_2 pour les phases 1 et 2 de RS. Par défaut, CP et RS_2 utilisent la simulation et RS_1 emploie LD. Pour les autres choix d'évaluateurs, nous ajoutons les suffixes /LD et /SIM. Par exemple, RS/SIM signifie que RS n'appelle que la simulation et jamais LD.

Dans la configuration du logiciel RS, les critères de convergence de LD par défaut sont $\kappa = 400$ et $\varepsilon = 10^{-4}$, ce qui était suffisant dans la majorité de nos expériences. À cause de la rapidité de RS, des démarrages multiples sont effectués par rapport au paramètre d'initialisation β. Dans la méthode d'initialisation, ξ est fixé à la même valeur que le seuil de SL global $l = 0.8$ et β varie de 0.2 à 0.9.

Dans la configuration de CP, le coefficient de charge α_i est initié à 1 pour les exemples sans abandon. Dans les cas avec abandons, α_i est réduit à 0.8 ou 0.9, car avec une valeur de 1, les solutions initiales possédaient déjà des niveaux de service élevés

tels que les SL de certains types d'appels étaient amplement satisfaits. Soit T la longueur des simulations en heures, nous avons fixé par défaut le pas pour la génération du sous-gradient à $d = 2$ lorsque $T \leq 50$ et $d = 1$ autrement.

Toutes les expériences ont été exécutées sur des ordinateurs avec des processeurs AMD Opteron 2.0GHz roulant sous Linux avec *Sun Java Development Kit* [37], version 1.4.2. Les problèmes linéaires sont résolus à l'aide de la librairie Cplex 9.0. Le simulateur utilisé est celui de Buist et L'Ecuyer [10].

5.1 CCN : Centre d'appels avec un routage en N

Nous commençons avec un centre d'appels de petite taille composé de 2 types d'appels et 2 groupes d'agents. L'avantage d'un petit exemple est qu'il est possible d'identifier la solution optimale. Un centre d'appels avec un routage en N (voir la figure 5.1) est un exemple des plus simples pour un centre avec des agents polyvalents.

Cet exemple peut représenter un centre d'appels avec des types de clients VIP (type 1) et normaux (type 2) où chacun possède son groupe d'agents, mais lorsque tous les agents du groupe 1 sont occupés, les appels importants (type 1) sont également servis par les agents du groupe 2. Par contre les agents du groupe 1 ne servent jamais les appels du type 2, car le centre ne veut pas nuire aux clients VIP. Une autre possibilité serait un centre bilingue où le type 1 représente la langue majoritaire et le type 2, une langue minoritaire, par exemple l'anglais et le français. Le centre d'appels emploie des agents unilingues anglophones (groupe 1) et des agents bilingues (groupe 2).

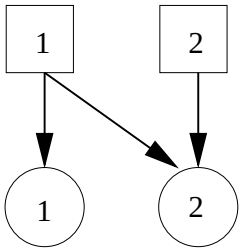

Figure 5.1 – Routage en N d'un centre d'appels avec 2 types d'appels (carrés) et 2 groupes d'agents (cercles).

Étant donné que le modèle d'approximation LD suppose une politique de routage qui n'est pas nécessairement identique à celle du centre d'appels, nous voulons également vérifier la sensibilité de LD par rapport à deux types de politiques différentes lorsqu'un agent termine un appel. La politique de routage pour l'arrivée des appels par liste de priorité demeure identique : $\mathscr{R}_1 = \{1,2\}$ et $\mathscr{R}_2 = \{2\}$. Un appel entrant du type 1 vérifie tout d'abord s'il y a un agent libre du groupe 1, sinon il vérifie dans le groupe 2 et s'il n'y a aucun agent libre, l'appel entre dans une file d'attente. Pour un appel entrant du type 2, il ne vérifie que s'il y a un agent libre du groupe 2 avant d'entrer dans une file d'attente. La première politique de service aux files d'attente est la règle classique du premier arrivé, premier servi, ou FIFO. Avec cette politique, l'agent répond à l'appel ayant attendu le plus longtemps parmi tous les types d'appels qu'il peut servir. La deuxième politique est construite selon une liste de préférence ou de priorité de la part des agents face aux types appels, puis les agents sélectionnent de manière FIFO parmi les appels d'un même type. Dans cet exemple, les agents du groupe 2 donnent la priorité aux appels du type 2, ainsi ils n'aident le groupe 1 que lorsque tous les appels du type 2 dans le système ont été servis. Nous pouvons supposer que cette politique est meilleure pour le type d'appel 2 que la politique uniquement FIFO, ce que nous observerons dans les prochaines figures.

Nous avons défini 6 exemples en combinant différents taux d'arrivée et politiques de routage. Il y a 3 configurations différentes des taux d'arrivée :

1 : $\lambda_1 = \lambda_2 = 160$ (taux symétriques)

2 : $\lambda_1 = 40, \lambda_2 = 160$ (taux asymétriques 1)

3 : $\lambda_1 = 160, \lambda_2 = 40$ (taux asymétriques 2)

et les deux politiques de routage :

F : FIFO

P : Préférence de l'agent.

Les différents exemples sont dénotés par {1,2,3}-{F,P} pour identifier la combinaison étudiée. Les paramètres constants sont les taux de service $\mu_1 = \mu_2 = 8$, les taux de patience (ou abandon dû à l'impatience) $\eta_1 = \eta_2 = 20$, les seuils minimaux sur les SL par type d'appel sont $SL_1 = SL_2 = 50\%$ et les coûts des agents $c_1 = 1$ et $c_2 = 1.2$.

Les simulations faites lors de nos expériences sont assez longues (horizon de 2 560

heures + 20% de réchauffement) pour que les largeurs des intervalles de confiance à 95% sur les niveaux de service soient généralement inférieures à 1%. Les figures 5.2 (global) et 5.3 (par type d'appel) montrent l'écart des SL entre les deux politiques de routage selon l'affectation des agents avec la configuration 1 des taux d'arrivée. Le nombre d'agents au groupe j est indiqué par l'axe x_j. Les figures confirment l'hypothèse que la politique de préférence des agents aide les appels du type 2 surtout lorsqu'il y peu d'agents du groupe 1. Lorsqu'il y a un surplus ou un grand manque d'agents, il n'y a aucune différence puisque tous les appels sont servis avant le délai τ ou la quasi-totalité a dû attendre un temps supérieur à τ. Dans la figure 5.3, il est intéressant de noter que le compromis d'aider les appels du type 2 avec la politique de préférence n'est pas symétrique entre la perte du SL du type 1 et le gain du type 2. Au pire, les appels du type 1 perdent 2% du niveau de service, alors que les appels du type 2 gagnent jusqu'à 30%. Ainsi, la politique de préférence des agents est meilleure que la politique FIFO au niveau global.

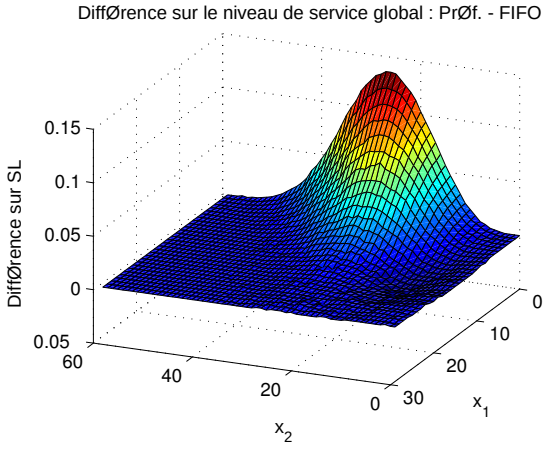

Figure 5.2 – CCN 1 : Différence sur le SL global entre la politique de préférence et FIFO. Les axes x_1 et x_2 représentent le nombre d'agents aux groupes 1 et 2 respectivement.

Les prochaines figures montrent les erreurs de l'approximation LD dans les cas 1-P et 1-F. La figure 5.4 indique que LD peut sur-évaluer et sous-évaluer jusqu'à 20% le niveau de service pour un même exemple. Le centre est opérable avec exclusivement des agents du groupe 2. Les contraintes sont satisfaites avec 50 agents du groupe 2, mais s'il n'y

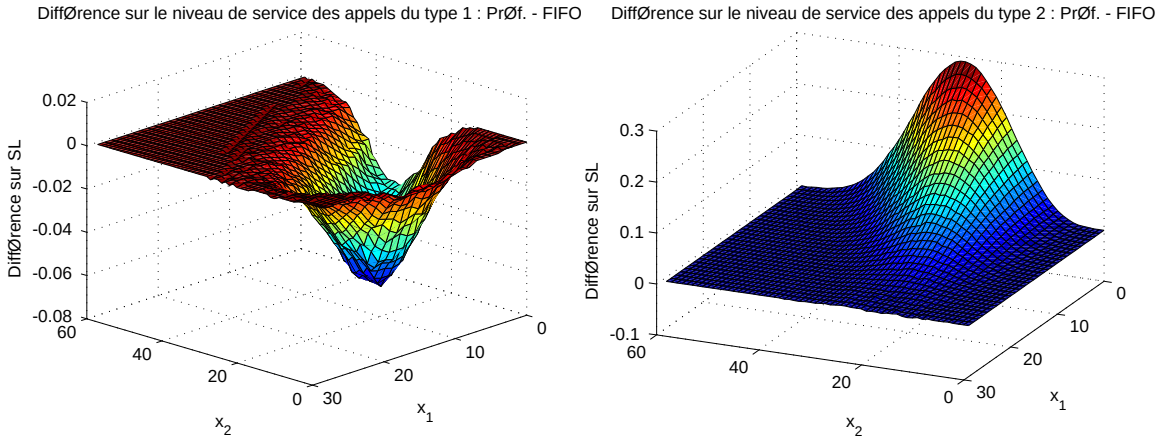

Figure 5.3 – CCN 1 : Différence sur le SL pour les types d'appels 1 (gauche) et 2 (droite) entre la politique de préférence et FIFO.

avait que des agents du groupe 1, alors le SL global serait borné à 50%. L'approximation surestime le SL des appels du type 1 jusqu'à 40% lorsqu'il y a principalement des agents du groupe 1. Cette erreur est similaire pour les deux politiques en comparant les figures 5.5 et 5.8. L'inverse se produit pour les appels du type 2 où l'approximation sous-estime le niveau de service. Alors que LD sous-estime jusqu'à 30% dans le cas de la politique de préférence (figure 5.6), l'erreur n'est au maximum que de 8% avec la politique FIFO (figure 5.9). La politique de routage de LD, supposant que tous les appels n'attendent qu'au groupe 2 et sont servis d'une manière FIFO, ressemble mieux à la politique FIFO dans cet exemple. La différence du SL du type 2 est la principale raison de la différence sur le niveau de service global entre les deux politiques de routage, voir les figures 5.4 et 5.7.

Nous avons résolu les exemples avec CP et RS avec des départs multiples sur le paramètre $\beta = \{0.2, 0.5, 0.7, 0.9\}$. La longueur des simulations pour CP et RS_2 était de 2560 heures. Étant donné la simplicité des exemples, nous avons aussi expérimenté RS en remplaçant LD par la simulation avec un horizon de 256 heures. Puisque RS commence dans les régions réalisables, l'erreur de LD est très petite et RS requiert généralement moins de 5 itérations autant pour RS/LD que RS/SIM.

Le tableau 5.1 montre les résultats de 3 réplications exécutées pour chaque exemple

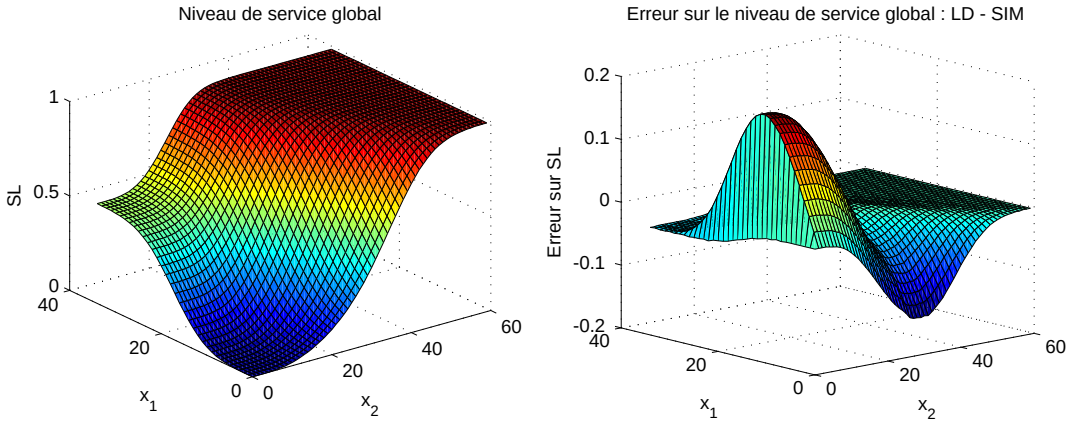

Figure 5.4 – CCN 1-P : SL global (gauche) et erreur de l'approximation LD (droite).

et chaque algorithme. Ce tableau contient le coût optimal trouvé par énumération du domaine réalisable, le coût minimal trouvé par chaque algorithme et le nombre de fois où chaque algorithme a trouvé l'optimum. L'optimum est unique pour chaque exemple. CP et RS/SIM sont comparables en réussissant à trouver l'optimum au moins une fois pour chaque exemple. Malgré que RS/LD n'ait jamais trouvé l'optimum, il était proche, tel qu'illustré pour l'exemple 1-P dans la figure 5.10. La figure (les axes représentant les nombres d'agents) montre la frontière du domaine réalisable telle que toutes les solutions sur et au-dessus de la frontière sont réalisables et toutes celles en-dessous sont irréalisables. L'optimum repose sur la frontière et il est dénoté par le symbole carré. Les solutions finales des 3 réplications de chaque algorithme sont identifiées par des flèches. Il arrive que plusieurs réplications terminent avec la même solution.

Pour ce petit exemple, les trois algorithmes (CP, RS/LD et RS/SIM) ont bien performé. De plus, les solutions évalués par LD dans RS/LD étaient dans des régions où il y avait peu d'erreurs d'approximation. Malgré que RS/SIM semble mieux performer que RS/LD, dès que les dimensions des exemples augmentent, RS/SIM devient rapidement trop coûteux en temps d'exécution.

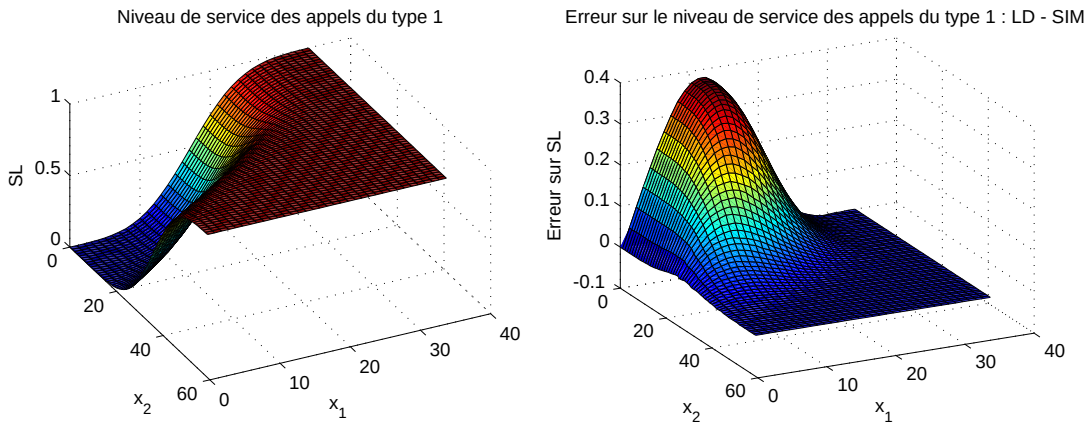

Figure 5.5 – CCN 1-P : SL du type d'appel 1 (gauche) et erreur de l'approximation LD (droite).

Exemple	Opt.	CP		RS/LD		RS/SIM	
		Min	# opt	Min	# opt	Min	# opt
1-F	47.0	47.0	1	47.2	0	47.0	1
1-P	46.8	46.8	2	47.2	0	46.8	1
2-F	32.0	32.0	1	32.6	0	32.0	3
2-P	31.8	31.8	3	32.6	0	31.8	2
3-F	28.2	28.2	2	28.4	0	28.2	3
3-P	28.2	28.2	1	28.6	0	28.2	2

Tableau 5.1 – CCN : Résultats des algorithmes CP, RS/LD et RS/SIM avec 3 réplications pour chaque problème. Montre le coût minimal trouvé et le nombre de réplications ayant trouvé l'optimum.

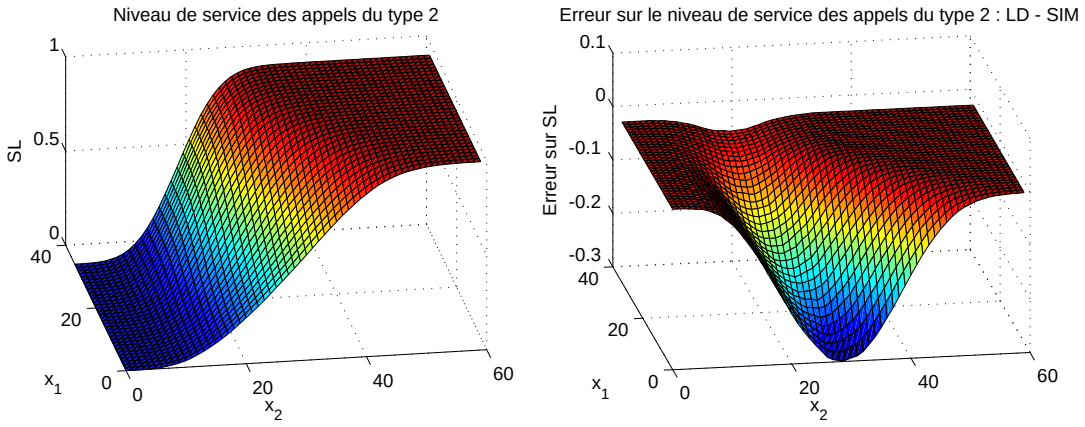

Figure 5.6 – CCN 1-P : SL du type d'appel 2 (gauche) et erreur de l'approximation LD (droite).

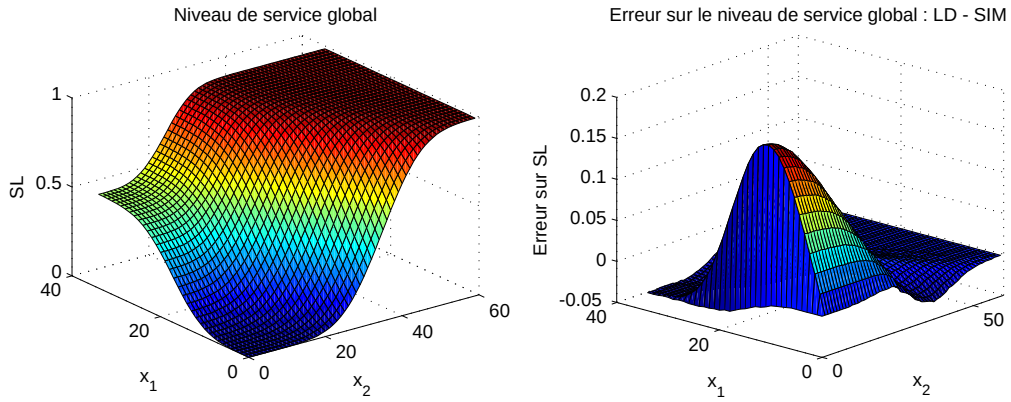

Figure 5.7 – CCN 1-F : SL global (gauche) et erreur de l'approximation LD (droite).

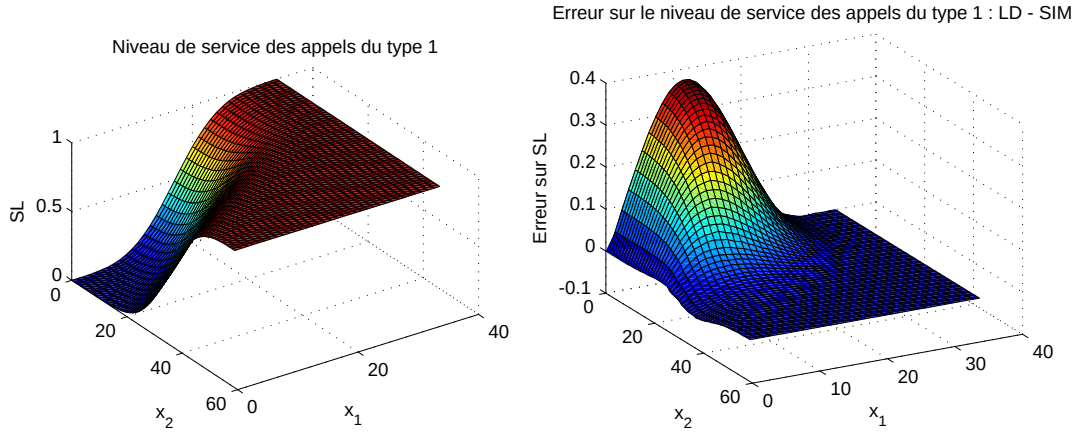

Figure 5.8 – CCN 1-F : SL du type d'appel 1 (gauche) et erreur de l'approximation LD (droite).

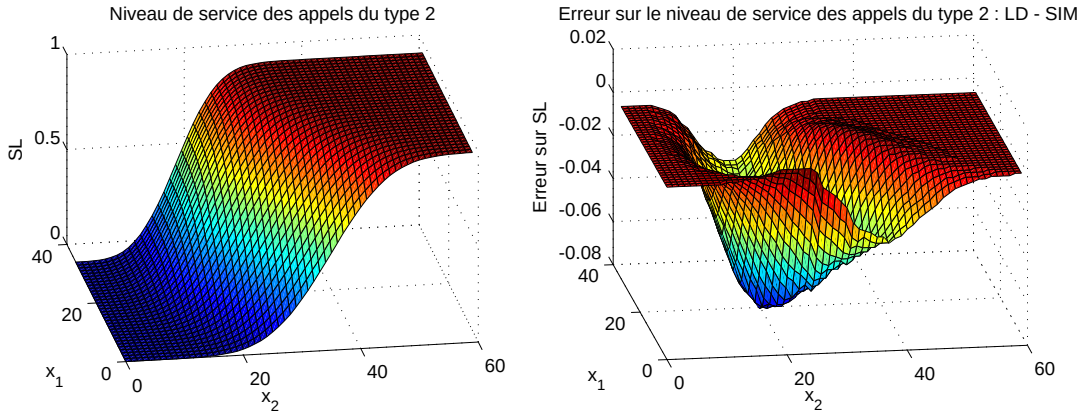

Figure 5.9 – CCN 1-F : SL du type d'appel 2 (gauche) et erreur de l'approximation LD (droite).

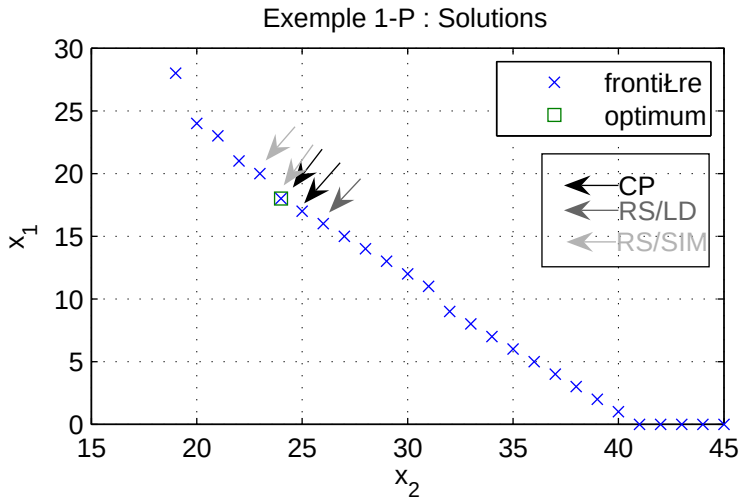

Figure 5.10 – CCN 1-P : Frontière des solutions réalisables accompagnée de l'optimum identifié par un carré. Les solutions finales des 3 réplications de chaque algorithme sont dénotées par des flèches. Les axes x_1 et x_2 représentent le nombre d'agents aux groupes 1 et 2 respectivement.

5.2 CC1 : Centre d'appels de moyenne taille

Cet exemple est également présenté dans Avramidis et al. [4]. Ce centre d'appels est composé de 7 types d'appels et 10 groupes d'agents avec 1 ou 2 habiletés. Le coût des agents ayant une seule compétence est de 1 et 1.05 pour 2 habiletés. L'ordre de priorité des appels entrants est : $\mathscr{R}_1 = \{1\}, \mathscr{R}_2 = \{1,3\}, \mathscr{R}_3 = \{2,4\}, \mathscr{R}_4 = \{5,4,3,6\}, \mathscr{R}_5 = \{7,6,8,9\}, \mathscr{R}_6 = \{9\}$ et $\mathscr{R}_7 = \{10,8\}$. Ce routage ne comporte aucun cycle tel que montré dans la figure 5.11. LD peut ainsi évaluer le centre d'appels avec une seule itération dont un ordre d'évaluation des groupes possible serait : $\{1,2,5,7,10,4,3,6,8,9\}$. Les agents répondent aux appels en attente avec la politique FIFO, à l'exception du groupe 1 qui donne la priorité aux appels du type 1 sur ceux du type 2 et du groupe 3 qui a une préférence sur les appels du type 2 face à ceux du type 4. Les taux d'arrivée sont $(\lambda_i)_{i=1}^7 = \{200,133,323,760,95,10,380\}$, les taux de service par type d'appel sont $(\mu_i)_{i=1}^7 = \{7.7,7.7,7.5,7.7,15,7,7,15\}$, les seuils minimaux des SL sont $(l_i)_{i=1}^7 = \{0.8,0.8,0.8,0.75,0.6,0.6,0.6\}$. Nous avons expérimenté le cas sans abandon et avec abandons ($\eta_i = 20, i \in \mathscr{N}$) que nous dénotons par CC1N et CC1A respectivement. Pour une raison de stabilité du simulateur, nous utilisons un taux d'abandon très faible ($\eta_i = 0.02, i \in \mathscr{N}$) pour CC1N et le nombre d'abandons est inférieur à 0.05% d'après nos résultats.

Nous avons résolu les deux cas avec différents budgets de CPU, avec l'algorithme CP et RS. CP fut exécuté sous forme d'un problème linéaire en nombres entiers ainsi que par la relaxation LP, dénotés par CP_{IP} et CP_{LP}. Dans le cas de CC1N, parfois CP_{IP} prenait trop de temps ; dans ces cas nous ne présentons que les résultats de CP_{LP}. Dans l'algorithme CP, nous avons fixé $\alpha_i = 1.0$ pour l'exemple CC1N et $\alpha_i = 0.9$ pour l'exemple CC1A, pour $i \in \mathscr{N}$.

Pour RS, nous avons effectué des départs multiples avec $\beta = \{0.2,0.5,0.7,0.9\}$ pour un petit budget de CPU et $\beta = \{0.1+0.1k, k = 1,\ldots,8\}$ pour un grand budget.

Le tableau 5.2 présente les performances des algorithmes RS et CP. Les expériences sont configurées en fonction de : la présence d'abandons (Ab.), l'algorithme utilisé (Algo.), le nombre de départs multiples pour RS ($|\beta|$), l'horizon de la simulation en

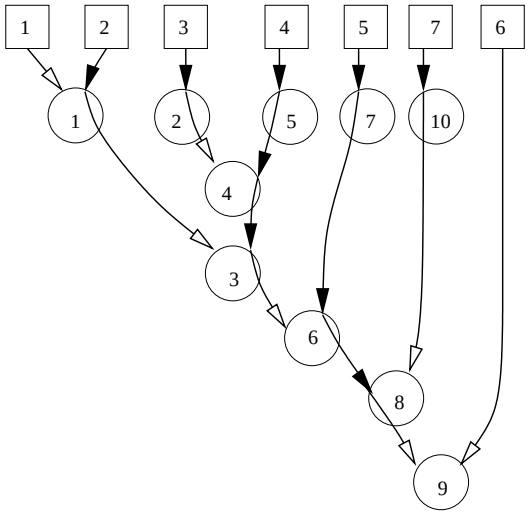

Figure 5.11 – CC1 : Schéma du routage par ordre de priorité des appels entrants ne comportant aucun cycle. Les types d'appels sont représentés par des carrés et les groupes d'agents par des cercles. Les flèches blanches pointent vers le dernier groupe de chaque liste de priorité $\mathscr{R}_\bullet$.

heures (T) et le budget de temps (CPU_{moy} en heures (h), minutes (m) et secondes (s)). Chaque cas représente un budget de CPU, variant d'une minute à plus d'une heure. Dans les résultats, le coût minimal et le coût médian sont conditionnels aux solutions finales réalisables parmi les 32 réplications selon de très longues simulations (horizon de 12 800 heures). Toutes les simulations débutent avec une période de réchauffement de 20% qui n'est pas incluse dans T, à l'exception des courtes simulations (≤ 50 heures) qui ont 40%. P_1 indique le nombre de réplications ayant trouvé une solution à un coût d'au plus 1% supérieur au coût de l'optimum empirique, peu importe la réalisabilité, P^* indique le nombre de réplications ayant obtenu une solution finale réalisable et P_1^* indique le nombre de réplications satisfaisant les 2 conditions. Les optima empiriques de CC1N et CC1A sont obtenus via plusieurs exécutions de CP et RS avec un grand budget de CPU et en prenant les meilleures solutions réalisables après de très longues simulations. Parfois, il a fallu des corrections manuelles pour obtenir ces optima, ce qui explique leur absence du tableau. L'indicateur $\bar{G}$ mesure la moyenne de l'écart relatif maximal entre la cible et son niveau de service parmi les solutions irréalisables,

$\bar{G} = 100 \times \max\{(l - g(\bar{\mathbf{x}}))/l, (l_{i^*} - g_{i^*}(\bar{\mathbf{x}}))/l_{i^*}\}$ où $i^* = \arg\max_{i \in \mathcal{N}}\{l_i - g_i(\bar{\mathbf{x}})\}$ et $\bar{\mathbf{x}}$ est une solution irréalisable.

Les figures 5.12 et 5.13 montrent la distribution des solutions finales des 32 réplications par rapport aux coûts optimaux empiriques à l'aide de diagrammes à surfaces (*box plots*). Un diagramme à surfaces divise une population en quartiles avec deux boîtes représentant les deux quartiles centraux et deux moustaches aux extrémités représentant les quartiles extrêmes. Dans la génération de ces diagrammes, chaque moustache a une longueur maximale de 1.5 fois la hauteur combinée des deux boîtes. Lorsqu'une donnée est en dehors de la couverture des moustaches, elle est identifiée comme étant une observation aberrante avec le symbole '+'.

RS_1 est très rapide vu la taille du centre. Le nombre d'itérations par valeur de β est généralement inférieur à 30 et le temps d'exécution de RS_1 est de moins de 3 secondes. Le reste du temps est dépensé dans la correction par simulation, RS_2. Malgré que l'ordonnancement des agents (section 4.3.6) dans LD accélère les temps d'évaluation, ce gain est négligeable.

Nous observons que RS et CP réussissent tous deux à trouver des solutions compétitives et qu'effectivement les chances d'obtenir une meilleure solution augmentent avec de plus grands budgets de temps d'exécution. Pour RS, le bruit de la simulation n'influence que la phase-2, RS_2. Le bruit de RS_1 est généralement similaire entre les différents cas. Par contre, CP est plus sensible au bruit de la simulation que RS. Avec de très courtes simulations, il arrive que CP produise une mauvaise coupe et résulte en une solution beaucoup plus coûteuse. Lorsque nous comparons CP_{IP} avec CP_{LP}, CP_{IP} converge, en fonction du budget de CPU, vers une meilleure solution plus rapidement que la méthode relaxée.

Le tableau 5.3 montre qu'il existe plusieurs optima locaux avec des coûts proches des solutions optimales empiriques et que ce ne sont que certaines contraintes qui sont critiques à la fois. Le taux d'abandon global pour CC1A est autour de 5.5%.

La figure 5.14 montre les erreurs de LD par rapport à SIM avec de longues simulations durant RS_1. Pour chaque solution courante d'une exécution de RS_1 donnée, nous comparons le SL global et celui pour les appels du type 2, qui était le type le plus contrai-

Ab	Cas	Algo.	$\lvert\beta\rvert$	T	CPU_{moy}	Coût min.	Coût méd.	P_1^*	P_1	P^*	$\bar{G}$
Non	1	RS	4	50	49s	242.65	243.40	5	27	6	4.2
		CP_{LP}	-	25	49s	242.10	244.80	4	18	15	5.3
	2	RS	4	160	2m36s	242.40	243.00	16	31	17	1.9
		CP_{LP}	-	80	2m38s	242.60	243.40	6	25	8	2.6
	3	RS	4	480	7m37s	242.40	243.55	10	27	15	0.8
		CP_{LP}	-	240	7m12s	241.85	243.35	15	26	19	1.6
	4	RS	8	640	21m46s	242.15	242.80	16	32	16	0.5
		CP_{LP}	-	640	17m37s	241.45	242.70	18	29	21	0.6
	5	RS	8	2560	84m21s	242.00	242.55	22	32	22	0.3
		CP_{LP}	-	2560	80m55s	241.35	242.30	27	32	27	0.5
Oui	1	RS	4	25	55s	224.40	224.85	2	30	4	1.7
		CP_{IP}	-	25	2m15s	222.95	225.75	3	16	11	3.0
		CP_{LP}	-	25	30s	225.05	227.08	0	14	6	2.6
	2	RS	4	50	1m33s	223.25	224.85	4	29	7	1.5
		CP_{IP}	-	50	2m46s	223.75	225.20	2	15	9	2.2
		CP_{LP}	-	50	57s	223.75	226.55	3	12	16	1.9
	3	RS	4	160	4m52s	223.25	224.45	8	27	13	0.7
		CP_{IP}	-	160	5m38s	223.40	225.10	7	20	14	0.8
		CP_{LP}	-	320	4m59s	222.70	224.72	8	18	16	0.4
	4	RS	4	640	17m10s	223.20	224.10	19	27	22	0.3
		CP_{IP}	-	640	15m29s	223.25	224.58	8	21	14	0.5
		CP_{LP}	-	1280	17m26s	223.05	224.70	14	22	23	0.6
	5	RS	4	1920	54m02s	223.25	224.40	15	23	24	0.2
		CP_{IP}	-	2560	59m12s	223.00	224.05	22	31	23	0.2
		CP_{LP}	-	3840	57m30s	223.85	224.85	12	17	24	0.3

Tableau 5.2 – CC1 : Performances des algorithmes RS et CP selon des budgets de CPU différents, basées sur 32 réplications.

Ab	Cas	Algo. Coût SL SL par type d'appel Vecteur d'affectation des agents
Non	*	- 241.30 0.804 ± 0.002 $(0.816, 0.850, 0.810, 0.808, 0.842, 0.609, 0.763)$ (42, 38, 16, 26, 69, 11, 0, 8, 3, 23)
	4	RS 242.55 0.801 ± 0.002 $(0.809, 0.828, 0.810, 0.868, 0.892, 0.763, 0.623)$ (42, 37, 12, 21, 86, 4, 3, 8, 4, 21)
		CP_{LP} 242.30 0.816 ± 0.002 $(0.831, 0.829, 0.838, 0.855, 0.869, 0.638, 0.696)$ (43, 38, 12, 25, 74, 12, 0, 11, 3, 19)
Oui	*	- 222.65 0.801 ± 0.001 (0.804, 0.805, 0.814, 0.789, 0.760, 0.621, 0.825) (39, 41, 12, 9, 81, 3, 0, 6, 4, 24)
	4	RS 224.10 0.804 ± 0.001 $(0.811, 0.839, 0.833, 0.813, 0.861, 0.760, 0.733)$ (39, 35, 14, 26, 70, 1, 0, 17, 5, 12)
		CP_{IP} 224.40 0.802 ± 0.001 $(0.824, 0.893, 0.799 \pm 0.002, 0.882, 0.686, 0.604, 0.629)$ (39, 23, 18, 40, 65, 4, 0, 2, 5, 23)
		CP_{LP} 224.50 0.805 ± 0.001 $(0.802, 0.886, 0.804, 0.863, 0.769, 0.625, 0.673)$ (38, 17, 20, 56, 52, 4, 0, 8, 4, 19)

Tableau 5.3 – CC1 : Solutions sélectionnées avec leur coût et leurs SL : les optima empiriques (cas "*") et les solutions médianes (16^{e} moins coûteuse parmi les 32 réplications) obtenues avec RS et CP en environ 17 minutes (cas 4). Les SL par type d'appel inférieurs à leur cible sont accompagnés d'un intervalle de confiance à 95%.

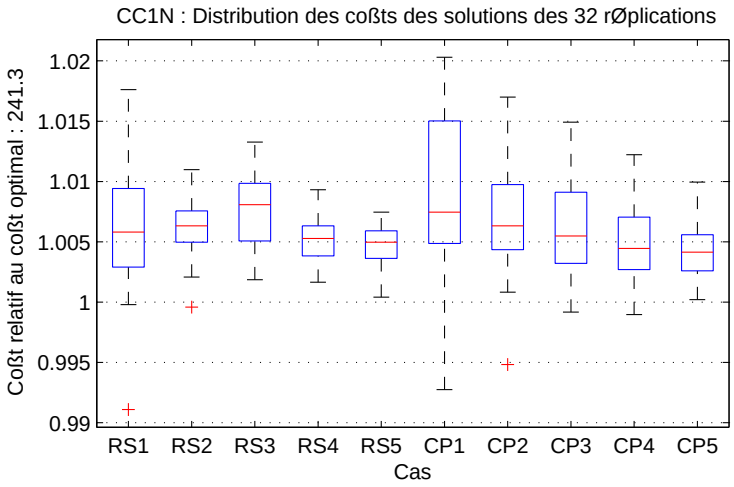

Figure 5.12 – CC1N : Diagrammes à surfaces représentant la distribution des coûts des solutions finales des 32 réplications relativement au coût optimal empirique.

gnant à la fin de RS_1. Comme pour l'exemple de la section 5.1, l'erreur de LD est plus petite dans les régions réalisables et augmente lorsque l'algorithme s'approche de la frontière de réalisabilité. Malgré que ces erreurs ne soient pas négligeables, nous nous apercevons que les courbes des niveaux de service $g(\mathbf{x})$ évoluent suivant des formes similaires, mais avec différentes amplitudes. Notre hypothèse est que malgré son inexactitude, LD réussit à bien guider RS_1 dans la sélection des voisinages.

La figure 5.15 montre les changements apportés par RS aux solutions initiales et à la fin de RS_2. Les points sont généralement proches de la ligne lorsque β est grand, alors qu'il y a plus de changements lorsque β est petit ($\beta = 0.2$). Il y a donc une corrélation entre β et l'homogénéité de la taille des groupes d'agents. Cette influence de la solution initiale est une raison pour utiliser de multiples départs sur β.

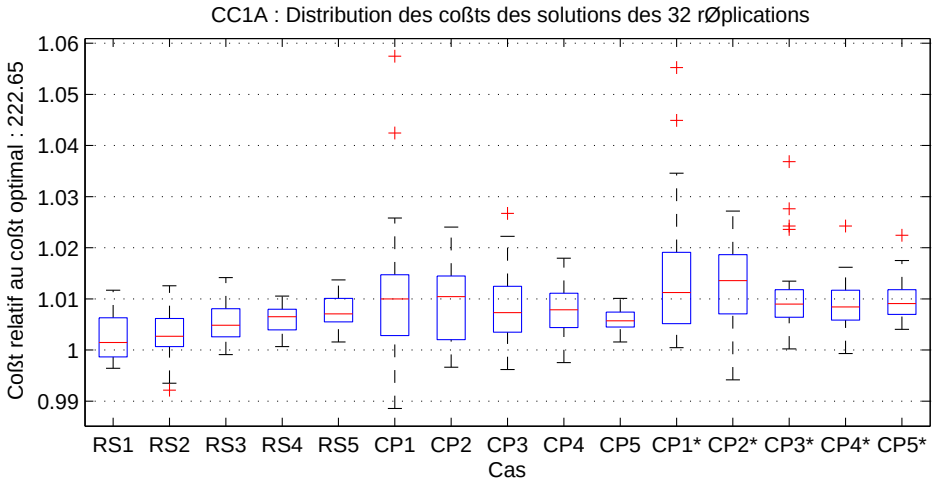

Figure 5.13 – CC1A : Diagrammes à surfaces représentant la distribution des coûts des solutions finales des 32 réplications relativement au coût optimal empirique. CP_{IP} est dénoté par CP et CP_{LP} par CP*.

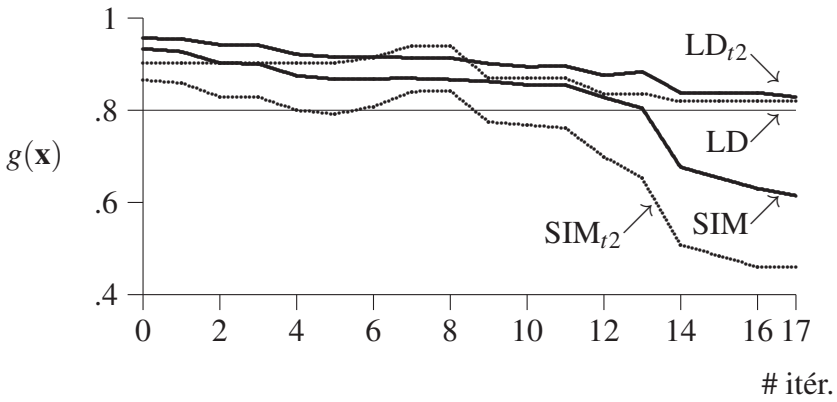

Figure 5.14 – CC1A : Pour chaque itération de RS_1 pendant une exécution, la courbe du SL global selon SIM et LD, et pour le type d'appel 2 ($t2$), le type le plus contraignant à la fin de RS_1 selon SIM.

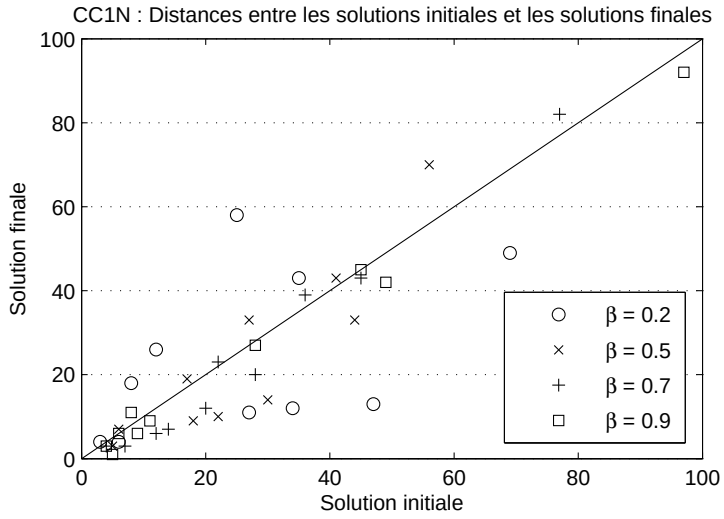

Figure 5.15 – CC1N : Distances entre les solutions initiales et les solutions finales pour chaque groupe d'agents, selon différents β. Les points au-dessus de la diagonale représentent un ajout d'agents et sous la diagonale, un retrait d'agents.

5.3 CC2 : Centre d'appels de grande taille

Ce grand exemple basé sur des données réelles, composé de 65 types d'appels et 89 groupes d'agents, est également présenté dans Ceẓik et L'Ecuyer [14] et Avramidis et al. [4]. Les taux d'arrivée varient de 1.046 à 416.6 appels par heure, les taux de service varient de 0.6667 à 22.5 et 600 pour un type d'appel. La charge du système est de 500 erlangs. Le nombre d'habiletés par groupe d'agents varie de 1 à 24 et le coût supplémentaire par habileté est de 5%, identique à l'exemple CC1, soit $c_j = 1 + 0.05(|\mathscr{K}_j| - 1)$. Les données originales n'incluaient que la règle de 80% en 20 secondes pour le SL global. Sans contrainte par type d'appel, nous obtenons des solutions dont les niveaux de service de certains types sont très mauvais. Généralement, un gestionnaire imposerait des critères sur le seuil du SL. Dans ce cas-ci, nous supposons un seuil minimal de $l_i = 0.5$ pour tous les types d'appels. Ceci laisse tout de même une bonne possibilité de manoeuvres pour satisfaire la contrainte globale $l = 0.8$.

Ce centre d'appels est divisé en 2 emplacements géographiques différents avec des groupes d'agents partageant souvent des habiletés identiques. Ainsi, les types d'appels et les groupes d'agents sont distingués par leur région. Dans cet exemple, une région possède 22 types d'appels et 15 groupes d'agents et l'autre région, plus grande, possède 43 types d'appels et 74 groupes d'agents. La politique de routage utilisée est celle du *spécialiste local*. Un nouvel appel entrant est assigné en priorité aux agents de la même région géographique et aux agents spécialistes tel qu'entre deux agents disponibles de groupes différents, celui ayant le moins d'habiletés (moins coûteux) est choisi. S'il n'y a aucun agent libre pouvant le servir, le routeur place l'appel dans une file d'attente. Après une durée d'attente de 6 secondes, le routeur vérifie s'il y a un agent libre parmi tous les agents des 2 emplacements pouvant servir cet appel, sinon l'appel demeure dans la file d'attente.

Lorsqu'un agent devient libre, il cherche premièrement s'il y un appel local en attente qu'il peut servir. Dans le cas où il n'y en a pas, l'agent cherche parmi les appels des 2 emplacements. Lorsqu'il y a plus d'un appel pouvant être servi, le choix est effectué suivant la politique FIFO sans distinction du type des appels.

Étant donné la grandeur de cet exemple et le nombre de voisinages à évaluer, le temps de calcul de LD n'est pas négligeable. Pour RS, des démarrages multiples sont effectués avec $\beta = \{0.6, 0.8\}$ pour un faible budget de CPU et $\beta = \{0.2, 0.5, 0.6, 0.7, 0.8, 0.9\}$ pour un grand budget. Pour CC2N, les formules sans abandon de LD permettent d'avoir des temps d'évaluations raisonnables. Par contre, LD avec abandons prenait au moins trois fois plus de temps avec les conditions d'arrêt par défaut. Une limite de temps a été fixée pour RS_1 sur l'ensemble des β. La limite était d'une heure pour les cas 1 et 2 (faible budget de temps), 2 heures pour le cas 3 et 12 heures pour le cas 4 (très long budget), voir le tableau 5.4. Nous montrons la différence de temps pour LD lorsque le paramètre de tolérance à l'erreur est augmenté de $\varepsilon = 10^{-4}$ (RS) à $\varepsilon = 10^{-3}$ (RS*). À cause du croisement de routage entre les 2 emplacements, les groupes d'agents se divisent en un faible nombre de partitions de graphes fortement connexes : 14 partitions d'un groupe, 1 partition de 3 groupes et 1 partition de 72 groupes. D'après nos expériences, il est préférable de simplement utiliser la méthode de Gauss-Seidel sans ordonnancement.

Pour CP, nous avons résolu le problème relaxé puisque même avec une limitation des temps de résolution de Cplex à 1 minute et un court temps de simulation $T = 50$, les durées d'exécution étaient énormes et leurs résultats étaient comparables pour un budget de temps de calcul similaire à la version relaxée. Dans la méthode d'initialisation de CP (section 3.2.2), nous avons fixé $\alpha_i = 1$ pour CC2N. Pour CC2A, puisqu'il y a peu de différence entre $\alpha_i = 0.8$ et 0.9 sur le coût final, nous avons choisi $\alpha_i = 0.8$. Un faible α_i donne une plus grande disparité sur les niveaux de service et par conséquent sur les taux d'abandon par type d'appel. Au niveau global, il y a peu de différence. Nous avons observé une sensibilité de CP face aux paramètres du pas du sous-gradient d et nous avons fixé $d = 2$ pour tous les cas dans CC2A. Comme nous résolvons le problème linéaire en nombres réels, la majorité du budget de CPU est consommée par le simulateur.

Le tableau 5.4 et les figures 5.16 et 5.17 montrent la difficulté du problème pour RS et CP. Les temps d'exécutions dépassent 20 minutes même avec de très courtes simulations. Les indicateurs P^* et $\hat{G}$ montrent la difficulté de trouver une solution finale réalisable, un problème relié à la durée des simulations T. Dans le cas de RS4* pour l'exemple CC2A,

Ab	Cas	Algo.	$\|\beta\|$	T	CPU_{moy}	Coût min.	Coût méd.	P_1^*	P_1	P^*	$\bar{G}$
Non	1	RS	2	80	24m52s	660.55	663.60	3	10	6	13.6
		CP	-	25	22m51s	668.75	668.75	0	4	1	20.0
	2	RS	2	320	45m44s	657.95	663.00	2	13	3	5.7
		CP	-	80	58m10s	677.20	677.20	0	6	1	15.7
	3	RS	6	1280	435m31s	657.20	659.50	9	15	9	1.9
		CP	-	960	567m52s	657.35	659.45	6	14	7	2.3
Oui	1	RS	2	25	52m01s	612.65	615.05	0	1	3	7.0
		RS*	2	25	30m08s	608.80	613.32	0	0	6	7.4
		CP	-	25	27m15s	631.10	634.65	0	0	3	8.6
	2	RS	2	80	59m35s	609.40	615.45	0	0	7	4.2
		RS*	2	80	35m49s	609.75	610.85	0	2	5	5.8
		CP	-	50	38m49s	614.40	622.60	0	0	4	10.1
	3	RS	2	160	96m56s	608.85	611.35	0	0	7	1.5
		RS*	2	160	46m12s	607.25	610.35	0	0	7	2.8
		CP	-	80	60m21s	616.25	621.95	0	0	3	8.3
	4	RS	6	640	804m25s	606.10	608.52	0	0	8	2.4
		RS*	6	640	420m16s	605.65	607.68	1	4	8	1.1
		CP	-	640	295m10s	605.20	613.25	1	5	5	2.0

Tableau 5.4 – CC2 : Performances des algorithmes RS et CP selon des budgets de CPU différents, basées sur 16 réplications.

deux réplications ont atteint la limite du budget de 12 heures. La durée moyenne des 10 autres réplications était de 332 minutes.

La durée moyenne de RS_1 dans CC2N avec $|\beta| = 2$ était de 17.5 minutes et 57.4 minutes pour $|\beta| = 6$. Le nombre moyen d'appels à LD par β était de 100 000. Pour CC2A, RS_1 avec la tolérance d'erreur de LD par défaut atteignait souvent la limite de temps accordé. Ceci est une explication de la différence entre les résultats trouvés par RS et RS^*, car RS ne pouvait pas toujours terminer sa recherche à l'intérieur du budget de temps accordé. Pour le cas 2 et la tolérance d'erreur par défaut, la durée moyenne était de 32 minutes et le nombre moyen d'évaluations LD était de 60 100. Dans le cas 4, ces moyennes étaient de 8.9 heures et 238 100 évaluations. Avec un relâchement de la tolérance (RS^*), les moyennes étaient de 25 minutes et 88 500 appels à LD et 2.4 heures et 267 300 évaluations pour les cas 2 et 4 respectivement. L'augmentation de la tolérance

dans LD peut donc réduire considérablement le temps d'exécution. Par contre, une tolérance trop grande serait nuisible, car les mesures de performances calculées pourraient être fortement erronées. L'arrêt prématuré de RS_1 dans le cas de RS expliquerait le nombre d'appels moyen à LD plus faible que RS^*. Tel que montré dans la figure 5.17, RS^* procure légèrement de meilleures performances que RS.

Le choix de β influence sur le nombre de voisins (moins il y a de généralistes, moins il y a de déplacements possibles d'agents avec la méthode *Déplacer*), ce qui affecte par conséquent le temps d'exécution. Comme RS ne fait que réduire le nombre d'agents ou échanger des agents d'un groupe vers un autre moins dispendieux, une initialisation avec un β proche de 1 donnerait une solution ayant moins de voisins qu'une solution issue d'un β plus petit. Dans nos expériences, RS avec $\beta = 0.8$ terminait plus rapidement qu'avec $\beta = 0.6$.

Pour CP, l'utilisation d'un grand T réduit généralement le nombre de coupes par sous-gradients. Dans l'exemple CC2A, il y avait environ 165 coupes pour le cas 1, 140 pour les cas 2 et 3 et 120 pour le cas 4. L'optimisation locale finale peut améliorer jusqu'à 2% la solution du problème relaxé dans chaque cas. Cependant, le nombre de simulations ne correspond pas au nombre de coupes multiplié par $m = 89$, car certaines coupes par type d'appel sont ajoutées simultanément. Le nombre total de simulations pour l'exemple CC2A dépassait rarement 1 200.

Les diagrammes à surfaces montrent que RS est moins sensible aux budgets de temps et à la longueur des simulations T que CP. Avec de courts budgets de CPU, RS semble mieux performer que CP. Cependant, CP réussit à trouver de meilleures solutions avec un très grand budget de temps (5 heures).

Le tableau 5.5 compare le coût et la réalisabilité des solutions médianes (8^{e} moins dispendieuse selon leur coût) des 16 réplications pour environ 45 à 90 minutes de temps de calcul avec les optima empiriques obtenus suite à de multiples réplications de RS et CP avec de très grands budgets de CPU (suivies parfois par des corrections manuelles). SL_{i^*} représente le niveau de service du type d'appel le moins satisfait : $i^* = \arg\min_{i \in \mathcal{N}} \{g_i(\mathbf{x}) - l_i\}$. Les niveaux de service sont accompagnés d'un intervalle de confiance à 95% et ceux en-dessous de leur cible sont indiqués en caractères gras. Les

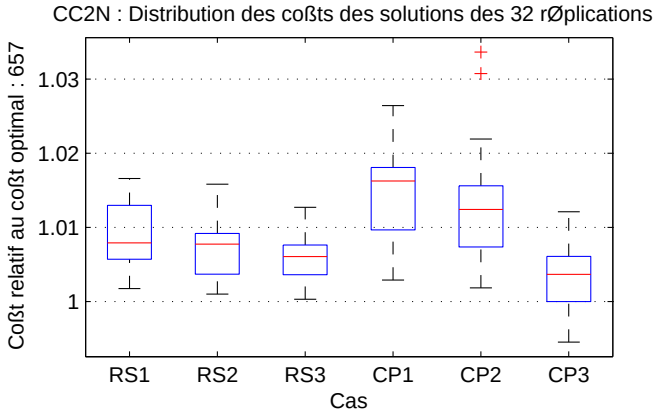

Figure 5.16 – CC2N : Diagrammes à surfaces représentant la distribution des coûts des solutions finales des 16 réplications relativement au coût optimal empirique.

Ab	Cas	Algo.	Coût	SL	SL_{i^*}
Non	*	-	657.00	0.850 ± 0.001	0.501 ± 0.004
	2	RS	661.80	0.867 ± 0.001	$\mathbf{0.487 \pm 0.006}$
		CP	664.95	0.868 ± 0.002	$\mathbf{0.36 \pm 0.02}$
Oui	*	-	600.00	0.812 ± 0.001	0.505 ± 0.003
	3	RS	610.55	0.865 ± 0.001	0.516 ± 0.003
		RS*	610.35	0.803 ± 0.001	0.500 ± 0.002
		CP	617.70	0.842 ± 0.001	$\mathbf{0.487 \pm 0.002}$

Tableau 5.5 – CC2 : Coûts et SL globaux des optima empiriques (cas "*") et des solutions médianes selon leur coût pour une durée d'exécution variant de 45 à 90 minutes.

solutions médianes de RS et CP sont loin des coûts optimaux pour les 2 cas présentés dans le tableau. La longueur de simulation T influence le niveau de réalisabilité, ainsi CP requiert une plus grande correction que RS. Ces chiffres indiquent que les seuils de SL pour les types d'appels sont plus difficiles à satisfaire que le niveau de service global. Le taux d'abandon global des solutions pour CC2A est autour de 7%.

Les figures 5.18 et 5.19 montrent l'évolution des niveaux de service. La solution courante de chaque itération durant RS_1, pour une exécution donnée, a été simulée sur un horizon de 1 280 heures. Les figures montrent l'erreur de LD par rapport à la simulation pour le SL global et pour les types d'appels 32 pour CC2N et 49 pour CC2A, qui étaient

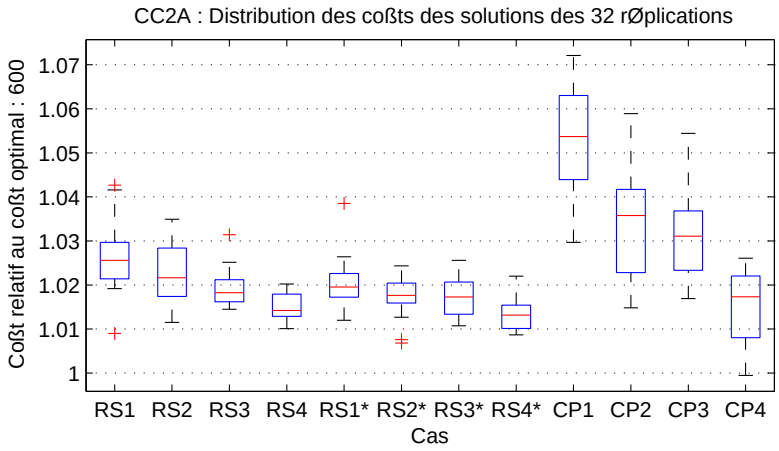

Figure 5.17 – CC2A : Diagrammes à surfaces représentant la distribution des coûts des solutions finales des 16 réplications relativement au coût optimal empirique. RS* dénote le cas avec une augmentation de la tolérance d'erreur dans LD.

les types d'appels les plus contraignants ($\arg\min_{i\in\mathcal{N}}\{g_i(\mathbf{x})-l_i\}$) à la fin de RS$_1$. Les observations sont semblables à celles de la figure 5.14 de l'exemple CC1A, où les courbes des niveaux de service entre LD et SIM suivent des formes similaires, mais les erreurs augmentent au fur et à mesure que les solutions s'approchent des régions irréalisables.

Lorsque nous comparons les exemples sans et avec abandons, une observation intéressante est que RS$_1$ s'arrête à $\pm 2\%$ des coûts finaux après RS$_2$ pour CC1N et CC2N, mais à -10% pour CC1A et CC2A. Ainsi, il y a eu plus de corrections de réalisabilité par simulation pour les exemples avec abandons. Puisque la solution à la fin de RS$_1$ était souvent irréalisable, la procédure d'amélioration par simulation (similaire à *Retirer*) était beaucoup moins utilisée et a retiré en général moins de 5 agents par solution.

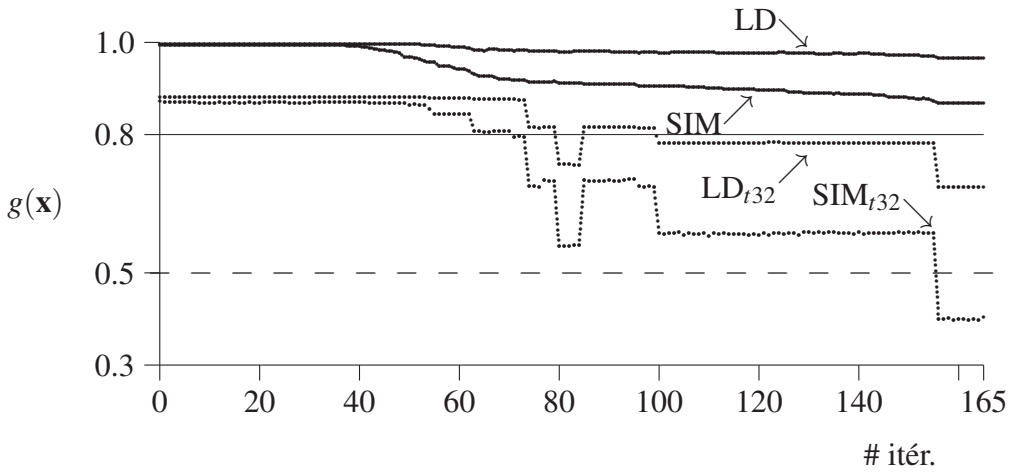

Figure 5.18 – CC2N : Pour chaque itération de RS_1 pendant une exécution, SL global selon SIM et LD, et pour le type d'appel 32 ($t32$), le type le plus contraignant à la fin de RS_1 selon SIM.

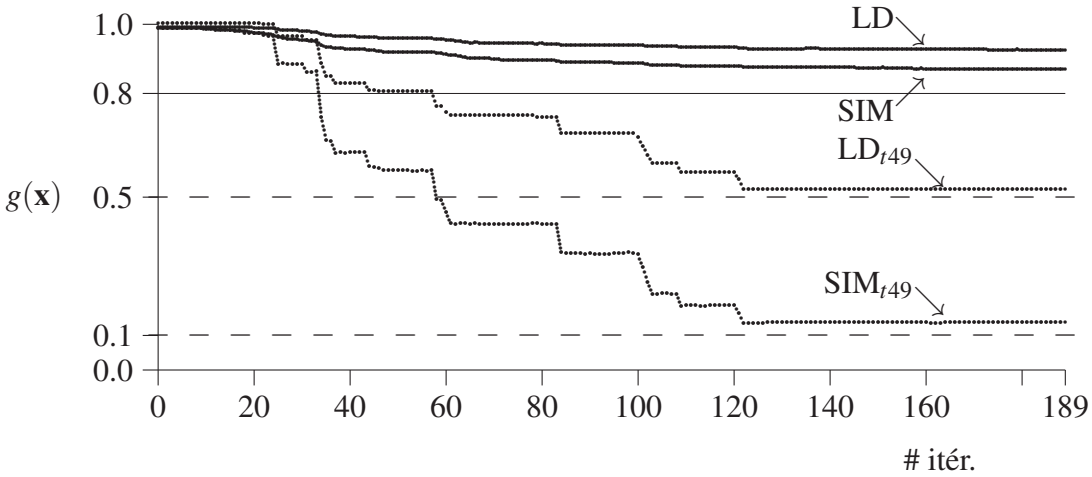

Figure 5.19 – CC2A : Pour chaque itération de RS_1 pendant une exécution, SL global selon SIM et LD, et pour le type d'appel 49 ($t49$), le type le plus contraignant à la fin de RS_1 selon SIM.

5.4 Analyse de sensibilité de RS

Dans cette section, nous analysons la sensibilité de RS avec différentes méthodes d'initialisation et examinons la sensibilité de RS_1 à d'autres générateurs de tailles de mouvements aléatoires.

5.4.1 Autres méthodes d'initialisation

Nous avons montré que RS était influencée par la solution initiale dans l'exemple CC1. Nous présentons ici les résultats obtenus en utilisant diverses autres méthodes d'initialisation :

1. **Géométrique**. Au lieu de répartir une fraction β des appels du type i vers le groupe le moins dispendieux et de diviser *également* le reste parmi les autres groupes dans $\mathscr{R}_i$, cette méthode répartit récursivement la proportion β aux groupes selon l'ordre croissant de leur coût par agent et le dernier (le plus dispendieux) reçoit le restant. Ainsi, les flots d'appels sont répartis en fonction du coût de chaque groupe. Pour que le groupe le plus dispendieux ne reçoive pas une quantité restante supérieure aux autres, il faut avoir un β tel que : $[1-\sum_{k=0}^{s_i-2}\beta(1-\beta)^k] \leq \beta(1-\beta)^{s_i-2}$ où $s_i = |\mathscr{R}_i|$. Une valeur de $\beta \geq 0.5$ satisfait ce critère pour tout $s_i > 1$.
2. **Sur-affectation.** Une méthode triviale pour générer une solution initiale réalisable est de sur-affecter le nombre d'agents dans chaque groupe.
3. **Aléatoire.** Les méthodes d'initialisation présentées jusqu'ici possèdent des caractéristiques spécifiques. Avec cette initialisation, nous voulons expérimenter une allocation aléatoire des agents qui donnera fort probablement une solution initiale irréalisable et nécessitera une correction. Pour ce faire, un nombre total aléatoire d'agents est réparti suivant une distribution (à plusieurs variables) de Dirichlet de paramètres $(\alpha_1, \ldots, \alpha_m)$ et de densité :

$$f(w_1, \ldots, w_m) = \left[\Gamma(\alpha_0) / \prod_{j=1}^{m} \Gamma(\alpha_j)\right] \prod_{j=1}^{m} w_j^{\alpha_j - 1},$$

où $\alpha_0 = \sum_{j=1}^{m} \alpha_j$ et $\Gamma(w+1) = w!$. Les variables $W_1, \ldots, W_m$ générées ont la propriété que : $\sum_{j=1}^{m} W_j = 1$. La solution initiale est générée comme suit :

(a) Nous générons les variables aléatoires $W_1, \ldots, W_m$ avec les paramètres $\alpha_i = \tilde{\alpha}$ pour $i \in \mathscr{M}$, chaque groupe sera donc généré de manière équivalente.

(b) Nous générons une variable aléatoire uniforme Y dans l'intervalle $[l,u]$ où l et u sont des paramètres.

(c) Avec $\rho = \sum_{i=1}^{n} \lambda_i/\mu_i$, les durées de service étant indépendantes des agents dans nos exemples, le nombre d'agents affectés au groupe j est :
$x_j = \text{arrondir}(\rho Y W_j)$.

La configuration est notée par D$\tilde{\alpha}y$ avec y représentant les valeurs $[l,u]$, soit a: $[0.8,1]$, b: $[1,1.5]$ ou c: $[1.5,2]$.

Nous avons également expérimenté une méthode alternative pour corriger les solutions irréalisables durant l'initialisation au lieu d'incrémenter un groupe d'agents à la fois. Premièrement, lorsque nous corrigeons une solution initiale irréalisable, nous voulons éviter qu'elle soit trop proche de la frontière de réalisabilité. Le paramètre $\nu \geq 0$ sert à contrôler la distance de la solution à la frontière. Ainsi, une solution initiale sera acceptée ssi $g(\mathbf{x}) \geq l + (1-l)\nu$ et $g_i(\mathbf{x}) \geq l_i + (1-l_i)\nu, \forall i \in \mathscr{N}$.

Deuxièmement, tous les groupes sont incrémentés d'un certain nombre d'agents à la fois au lieu d'un agent d'un seul groupe. Pour ce faire, ξ est incrémenté : $\xi = \tilde{\xi} + (1-\tilde{\xi})\zeta$, où $\tilde{\xi}$ est l'ancienne valeur de ξ, jusqu'à l'obtention d'une solution réalisable selon LD. Dans le cas de l'initialisation par Dirichlet, il n'y a pas de paramètre ξ. Les variables générées $W_1, \ldots, W_m$ ne changent pas, mais la charge du système ρ est augmentée de 2% jusqu'à l'obtention d'une solution initiale.

Les figures 5.20 et 5.21 montrent les résultats des différentes méthodes d'initialisation de RS pour les examples CC2N et CC2A. Les différents cas présentés sont :

Ay : Initialisation par défaut avec $\xi = 0.8$ et $\beta = y$.

By : Identique au cas Ay sauf que $\xi = 0.3$.

Cy : Initialisation géométrique avec $\xi = 0.8$ et $\beta = y$.

S : Méthode de sur-affectation avec 100 agents par groupe.

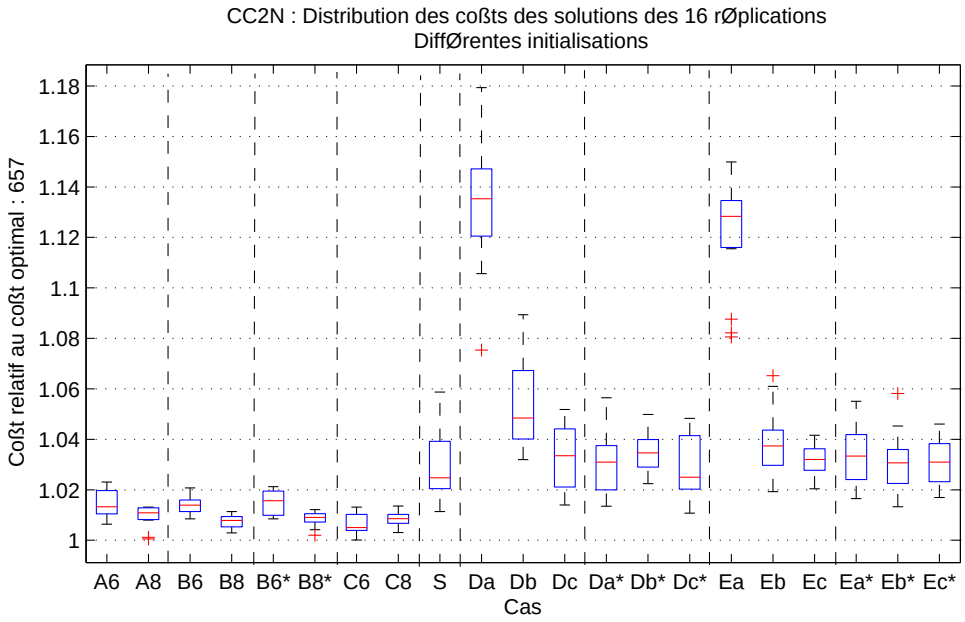

Figure 5.20 – CC2N : Diagrammes à surfaces comparant différentes méthodes d'initialisation dans RS, basés sur 16 réplications.

Dy : Initialisation avec Dirichlet : D1y.

Ey : Initialisation avec Dirichlet : D4y.

Le symbole "*" indique l'utilisation de la méthode alternative pour la correction avec $\nu = 0.05$ et $\zeta = 0.01$. Il n'y avait aucune correction à apporter dans les cas A, C et S.

Dans les deux exemples, les meilleures initialisations sont obtenues avec les méthodes par défaut et géométrique avec $\xi = 0.8$, donc pas de correction. Malgré que $\xi = 0.3$ ne semble pas dégrader l'efficacité de RS dans CC2N, la différence est marquante dans CC2A. Les initialisations aléatoires et la sur-affectation des groupes ont les pires performances, surtout lorsque la méthode de correction par défaut était utilisée (initialisation avec Dirichlet avec peu d'agents). Avec la correction alternative, puisque tous les groupes sont incrémentés à la fois, plusieurs groupes se trouvent sur-affectés et cette méthode devient similaire à la sur-affectation.

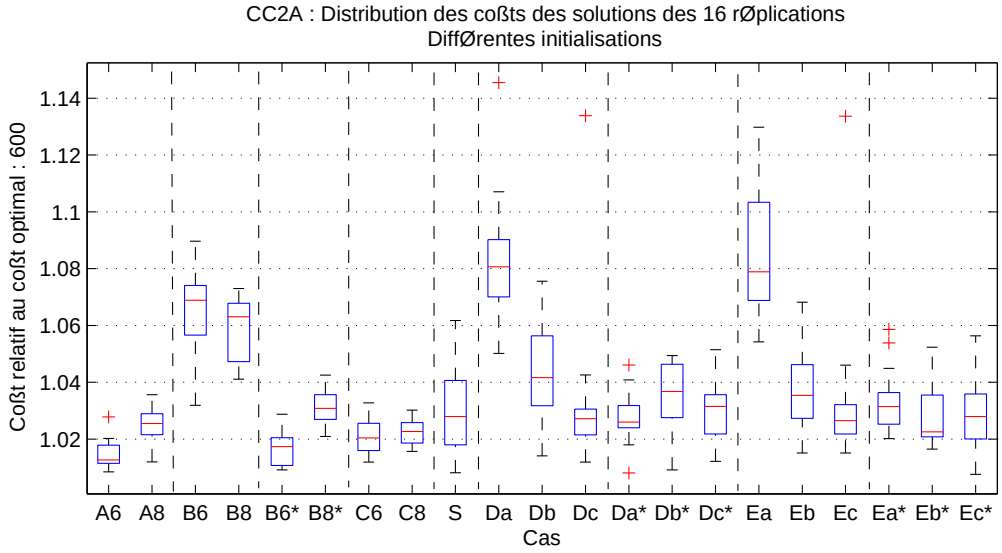

Figure 5.21 – CC2A : Diagrammes à surfaces comparant différentes méthodes d'initialisation dans RS, basés sur 16 réplications.

5.4.2 Différentes tailles de mouvement de recherche

Une partie de la randomisation vient de la génération d'une taille de mouvement aléatoire q à chaque itération de RS_1. Par défaut, l'algorithme génère $E \sim Exponentielle(\text{méd}(\mathbf{x}))$, une exponentielle où la moyenne est la taille médiane des groupes d'agents, et $q = \max(1, \text{arrondir}(E))$. Nous avons expérimenté d'autres méthodes pour le choix de q :

1. $q = 1$. Notons que la procédure *Retirer* devient déterministe. [1]
2. $q = \text{arrondir}(U)$ où $U \sim Uniforme(1, \max(\mathbf{x}))$, un choix aléatoire uniforme entre 1 et la taille du plus grand groupe d'agents. [U]
3. $E \sim Exponentielle(2)$, une distribution exponentielle fixe de moyenne 2. [E(2)]
4. $E \sim Exponentielle(y \times \text{méd}(\mathbf{x}))$ avec $y \in \{0.5, 1, 5\}$. Nous avons la taille par défaut avec $y = 1$. [E(*y*m)]

Les figures 5.22 et 5.23 montrent les résultats de ces méthodes sur les exemples CC1A, CC2N et CC2A sur 32 réplications. Pour mieux comparer les différents q, nous n'avons pas utilisé de démarrages multiples : $\beta = 0.7$ pour CC1A et 0.8 pour CC2 et

sans limite de temps pour RS_1. Les durées de simulation T sont 640, 320 et 160 pour CC1A, CC2N et CC2A respectivement.

Les résultats montrent que l'influence de q est plutôt minime comparée à l'initialisation de RS. Les différences de coût ne dépassent guère 1%. Notons que la solution initiale est identique dans tous les cas puisque la méthode d'initialisation est déterministe. Dans le cas où $q = 1$, une constante, la randomisation vient du choix aléatoire du groupe pivot dans la procédure *Déplacer*.

Dans l'exemple CC1A, RS_1 avec un petit q (cas 1 et E(2)) choisissait souvent les mêmes régions de voisinage et aboutissait à une solution locale moins optimale. Alors qu'avec un q plus grand, les voisinages étaient plus diversifiés. Par contre, un grand q signifie aussi le risque d'avoir plus de voisinages réalisables vides. Ceci peut ralentir RS, car le temps d'évaluation de LD dans certains problèmes peut être non négligeable, par exemple CC2. Le nombre d'appels à LD et le temps d'exécution de RS_1 pour le cas uniforme est plus du double que pour le cas $q = 1$. Nous aurions pu penser qu'un grand q aiderait à réduire le nombre d'itérations dans RS_1, mais les grands mouvements ne sont souvent possibles qu'au début de la recherche, le reste des itérations sont généralement des mouvements de petites tailles. Dans tous les 3 exemples, E(0.5m) et E(m) ont donné les meilleures performances.

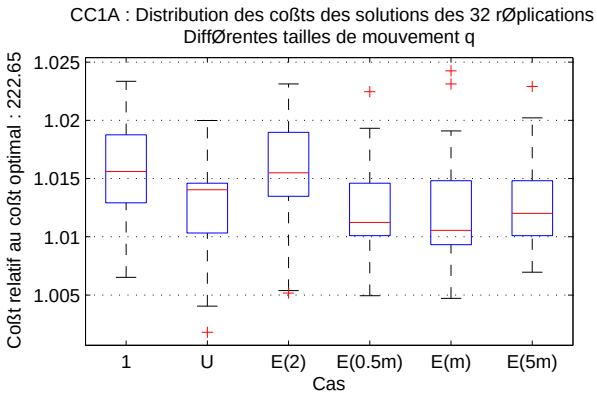

Figure 5.22 – CC1A : Diagrammes à surfaces comparant différentes méthodes de génération de la taille de mouvement q, basés sur 32 réplications.

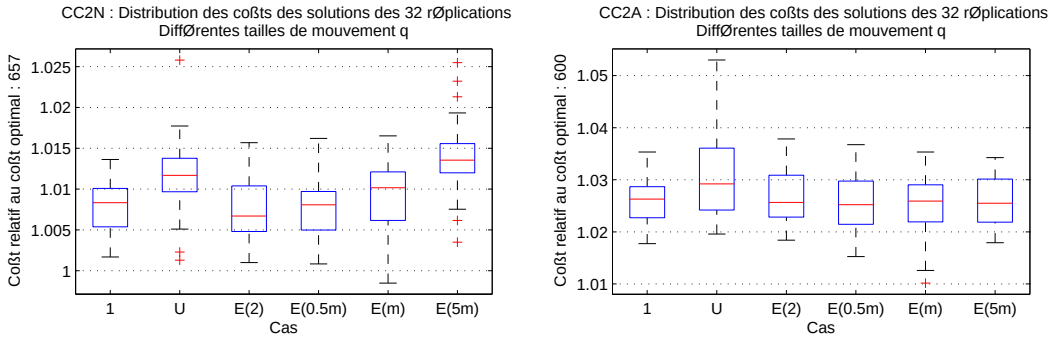

Figure 5.23 – CC2 : Diagrammes à surfaces comparant différentes méthodes de génération de la taille de mouvement q, basés sur 32 réplications.

5.5 Analyse de sensibilité de CP

Nous avons évalué la sensibilité de CP face aux paramètres de contrôle du pas d lors du calcul du sous-gradient par la méthode des différences finies (figure 5.24) et de contrôle de couverture de charge α_i pour chaque type d'appel (figure 5.25). Pour les cas CP1 à CP4 de l'exemple CC2A, nous avons expérimenté les valeurs de $d \in \{1,2,4\}$. Nous observons que d a effectivement un plus grand effet lors de courtes simulations (T variant de 25 à 80 heures). Alors qu'avec une longue simulation, chaque d est capable de trouver de bonnes solutions, mais $d = 4$ en trouve plus fréquemment. Pour CC2A, $d = 4$ aurait été un meilleur paramètre. Nous avons testé les cas CP1 à CP3 de l'exemple CC2A avec $\alpha_i \in \{0.5,0.7,0.8,0.9,1\}$ et $d_j = 4$. Les meilleurs résultats sont obtenus avec α_i entre 0.7 et 0.9. Il est à noter que α_i influence le taux d'abandon via le niveau de service. Un petit α_i risque de donner des niveaux de service plus proches des seuils et par conséquent, des taux d'abandon plus élevés pour certains types d'appels, alors qu'un plus grand α_i aurait un niveau d'abandon plus homogène entre les types d'appels. Cependant, les paramètres de CP et leur sensibilité diffèrent selon les exemples.

5.6 Autres méthodes d'optimisation

Les logiciels RS et CP (ainsi que LD et SIM) sont implémentés en plusieurs modules (ou *objets* dans le jargon de la programmation orientée objet). Ceci nous permet de com-

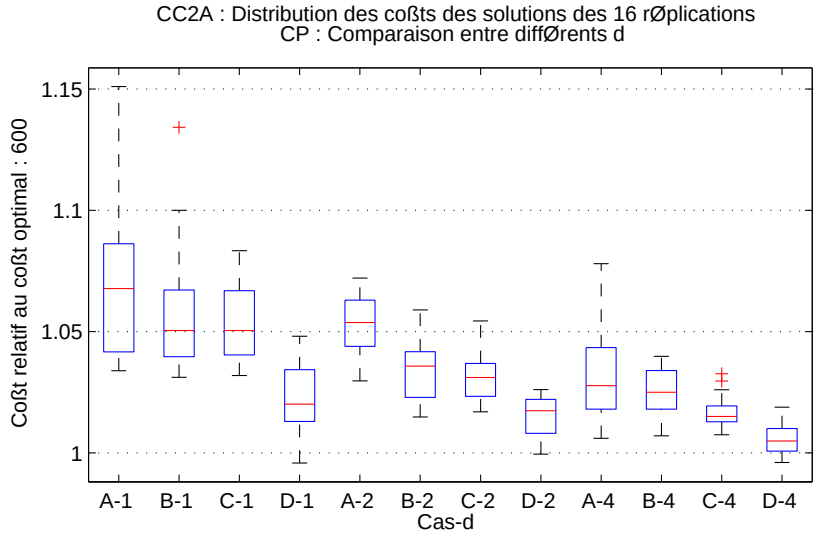

Figure 5.24 – CC2A : Diagrammes à surfaces comparant l'utilisation de différents d lors du calcul du sous-gradient dans CP. Les configurations CP1 à CP4 sont identifiées par les lettres de A à D respectivement suivis par la valeur de d utilisée.

biner différents modules tels que CP avec LD ou RS_1 avec SIM. Cette section présente quelques résultats obtenus par le mélange de certains modules.

5.6.1 Algorithme CP avec l'approximation LD

Dans cette expérience, SIM est remplacée par LD tout au long de CP. La solution finale est corrigée à l'aide de SIM avec la procédure RS_2 présentée à la section 4.4.3. Puisque LD est déterministe, le nouvel algorithme CP/LD est également déterministe. Les différences entre les réplications sont causées uniquement par le bruit de la simulation lors de la correction de la solution finale. Puisque CP est un algorithme qui travaille dans les régions irréalisables, nous utilisons la version de LD avec abandons. Dans les trois exemples, la longueur des simulations était de $T = 640$ heures. Le tableau 5.6 présente les résultats des 3 exemples basés sur 16 réplications. Les indicateurs sont le temps d'exécution de CP/LD (T_{CP}) et le coût trouvé (identique pour les 16 réplications), suivi du temps d'exécution total moyen ($\bar{T}$) et l'intervalle des coûts finaux trouvés après la correction par SIM, puis le pourcentage du coût (Opt) au-dessus de celui de l'optimum

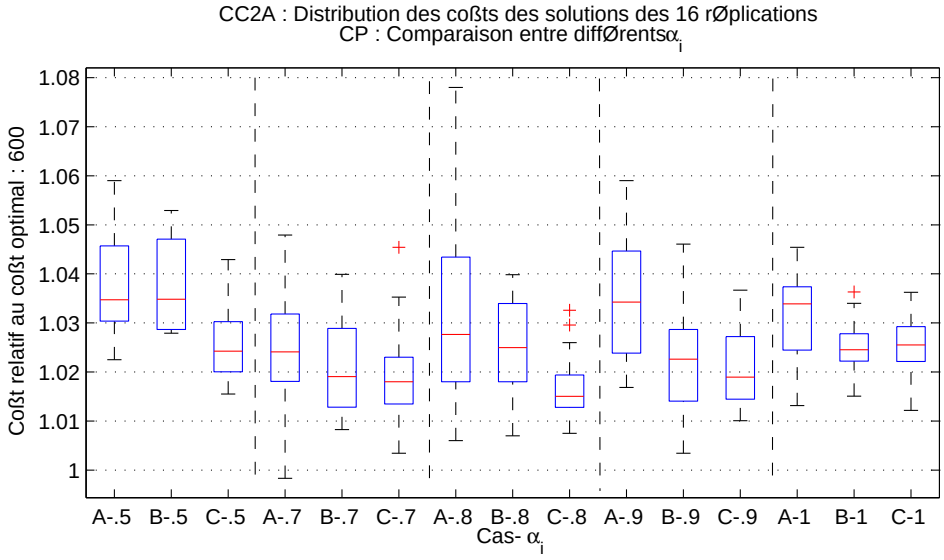

Figure 5.25 – CC2A : Diagrammes à surfaces comparant l'utilisation de différents coefficients de couverture de charge α_i dans CP. Les configurations CP1 à CP3 sont identifiées par les lettres de A à C respectivement suivis par la valeur de α_i utilisée, identique pour tout $i \in \mathcal{N}$.

empirique.

Ex	T_{CP}	Coût_{CP}	$\bar{T}$	Coût	Opt
CC1A	<1 sec	198.80	9 min	[226.80, 227.95]	2%
CC2N	14 sec	631.25	35 min	[662.45, 665.70]	1%
CC2A	14 sec	603.45	27 min	[616.40, 620.60]	3%

Tableau 5.6 – Résultats de la combinaison entre l'algorithme CP et l'approximation LD avec correction finale par SIM.

Pour des temps d'exécutions similaires à CP, ces résultats sont comparables en général et légèrement meilleurs pour les exemples CC2. De plus, la variance des coûts des solutions est beaucoup plus faible. Pour CC2A, CP/LD a fait 50 fois moins d'appels à LD que RS et le temps d'exécution avant la correction par simulation était moins d'une minute. La quasi-totalité du temps d'exécution était consacrée à la correction de la réalisabilité par simulation. Bien que la combinaison de CP avec LD semble bien fonctionner, il est à noter que l'algorithme CP/LD serait probablement limité par la fiabilité

de LD pour des exemples différents, car ce modèle d'approximation est beaucoup moins flexible que la simulation. Néanmoins, la rapidité (même pour de grands problèmes) pour obtenir des solutions "grossières" demeure intéressante.

5.6.2 Algorithme RS avec uniquement SIM

Vu l'erreur non négligeable de LD, cette section présente les résultats obtenus en remplaçant LD par SIM pendant toute l'exécution de RS. La solution initiale est toujours générée à l'aide des formules dérivées à partir des chaînes de Markov en temps continu. Dans la section 5.1, cet algorithme a mieux performé que RS/LD pour un petit exemple. Nous avons également expérimenté RS/SIM pour les exemples CC1 et CC2.

Pour CC1, LD est remplacé par SIM avec une longueur de simulation $T_1 = 25$ heures et 10 heures de réchauffement. Les autres paramètres étaient : $\beta = \{0.2, 0.5, 0.7, 0.9\}$ et $T = 640$ heures. Les temps moyens d'exécution de 32 réplications étaient de 29 minutes pour CC1N et 27 minutes pour CC1A. Même avec des courtes simulations, RS_1/SIM a pris un peu plus de temps que la deuxième phase RS_2/SIM, alors que RS_1/LD prenait moins de 20 secondes. Par contre, le nombre de corrections par simulation était légèrement réduit. Les coûts finaux des solutions étaient comparables à ceux trouvés par RS/LD autant pour CC1N que CC1A.

La différence entre RS/LD et RS/SIM était plus apparente dans l'exemple CC2, en particulier CC2A. Nous avons utilisé les mêmes paramètres que pour le cas 2 pour CC2N et le cas 3 pour CC2A, avec $T_1 = 25$ heures. Pour limiter le temps d'exécution, la taille des voisinages de chaque mouvement de RS_1/SIM était limitée à 10 voisins (choisis au hasard s'il y en avait plus de 10). Cette limitation du voisinage a été également testée pour RS_1/LD, les différences sur le coût étaient négligeables, mais il y avait une légère économie sur le temps de RS_1/LD. Les temps moyens d'exécution de 5 réplications étaient de 22 heures et 15 heures pour les exemples CC2N et CC2A respectivement, avec 97% du temps occupé par RS_1/SIM. Les coûts étaient de 2% à 3% au-dessus du coût optimal pour CC2N et de 4% à 6% pour CC2A, ce qui est légèrement moins performant que RS/LD. Le temps d'exécution très long rend cet algorithme peu attrayant dans la pratique.

5.6.3 Variations aux mouvements *Retirer* et *Déplacer* dans RS

En général, retirer un agent réduit davantage le niveau de service et le coût d'affectation que de réduire son nombre de tâches en le déplaçant vers un autre groupe. En donnant la priorité à *Retirer* sur *Déplacer* dans RS, nous voulons atteindre plus rapidement la solution optimale. Cela se produit effectivement dans RS : après une série de mouvements *Retirer*, le nombre total d'agents n'est réduit que parcimonieusement par la suite. La taille du voisinage avec un q donné pour *Retirer* est dans $O(m)$, de même pour *Déplacer* avec un pivot p donné. Dans cette section, nous avons agrandi le voisinage des mouvements pour essayer d'améliorer les solutions, principalement celles obtenues à partir d'une mauvaise initialisation.

Dans la variation 1, pour un q donné, tous les candidats pour le pivot sont examinés : l'algorithme cherche le meilleur déplacement de q agents possible. La taille du voisinage de *Déplacer* devient ainsi $O(m^2)$. Nous avons testé avec les exemples CC2N et CC2A avec de mauvaises initialisations générées par une distribution de Dirichlet. Malgré le temps d'exécution au moins 8 fois plus grand, cette variation ne semble pas apporter une réduction au coût.

Dans la variation 2, nous avons agrandi la taille du voisinage de la variation 1 en donnant une priorité *égale* pour les mouvements *Retirer* et *Déplacer* pour un q donné. À l'itération v, le voisin choisi est celui qui minimise : $[g(\mathbf{x}^{(v)}) - g(\mathbf{x})]/[\mathbf{c}^{\mathbf{t}}\mathbf{x}^{(v)} - \mathbf{c}^{\mathbf{t}}\mathbf{x}]$. L'union des deux mouvements donne un voisinage de taille $O(m^2 + m)$. Nous avons testé sur les mêmes instances que pour la variation 1. Un comportement intéressant est que dans tous les cas, RS_1 choisissait les mouvements *Déplacer* au début, puis les mouvements *Retirer* apparaissaient beaucoup plus tard. La série de déplacements au début lorsqu'il y avait un surplus d'agents a permis de modifier considérablement la structure de la solution. Dans un exemple de CC2N où une mauvaise initialisation avait donné une solution avec un coût 20% supérieur au coût optimal, la variation 2 a permis d'obtenir une bonne solution avec un coût de 658.85, soit à 0.2% au-dessus du coût optimal. Pour l'exemple CC2A, il y a eu une légère amélioration par rapport à la mauvaise initialisation, mais le coût était tout de même 3.4% au-dessus du coût optimal. Malheureusement,

ces variations ne sont pas applicables dans la pratique à cause du temps d'exécution élevé.

CHAPITRE 6

CONCLUSION

Les logiciels d'optimisation stochastique pour l'affectation des agents dans un centre d'appels, qui constituent les contributions principales de ce mémoire, sont des outils importants lors de la conception des horaires par les gestionnaires de centres d'appels. J'ai implémenté et expérimenté un algorithme d'optimisation par coupes linéaires et simulations (CP) et un algorithme de recherche randomisée (RS) utilisant un modèle d'approximation (LD) basé sur les chaînes de Markov en temps continu. Ce sont deux algorithmes très différents. CP explore les régions irréalisables et ajoute des coupes jusqu'à l'obtention d'une solution réalisable, alors que RS explore les régions réalisables jusqu'à un optimum local.

Malgré l'imprécision de notre algorithme d'approximation LD, les exemples numériques ont montré qu'il était un bon outil pour guider la recherche. Nos deux algorithmes CP et RS ont tous trouvé des solutions compétitives. Il y a cependant quelques différences de performance. La rapidité de RS, qui permet d'utiliser de plus longues simulations que CP, et l'évaluation sans bruit de LD augmentent la stabilité de RS (moins de variance). RS donne généralement de meilleurs résultats pour de courts budgets de temps d'exécution. D'autre part, l'augmentation du budget de temps (via l'augmentation de la longueur des simulations) permet d'avoir de meilleures coupes et pour de grands budgets, CP semble plus performant que RS. Nos exemples ont montré qu'il existe souvent plusieurs bonnes solutions avec différentes répartitions des agents.

Il n'existe présentement aucun algorithme d'approximation analytique précis et robuste pour les centres d'appels modernes. Considérant la diversité des politiques de routage possibles dont chacune requerrait une approximation analytique différente, la simulation demeure l'outil d'approximation le plus fiable et versatile. Malgré les résultats non-biaisés, la simulation retourne les approximations de performances avec un bruit estimé par des intervalles de confiance. La simulation est un outil d'évaluation coûteux en temps d'exécution dans un algorithme d'optimisation. En ne supposant aucune tech-

nique de réduction de variance, pour réduire le bruit d'un facteur de a, il faudrait a^2 fois plus de simulations. Obtenir des évaluations sans bruit est quasiment impossible dans la pratique. Puisque les solutions optimales se trouvent sur les frontières de réalisabilité, les solutions retournées par les logiciels d'optimisation risquent d'être irréalisables, tel que montrées dans le chapitre 5.

La polyvalence des agents et le partage d'habiletés à servir des types d'appels communs résultent en un problème combinatoire difficile pour une période. Cependant, la fonction de coût est souvent plate, de sorte qu'il existerait plusieurs optima locaux ayant des coûts proches du véritable optimum.

Dans les développements futurs, nous voulons étendre nos logiciels afin de résoudre le problème d'horaire pour une journée et sur plusieurs périodes. Des contraintes d'horaire s'ajoutent aux problèmes ; ainsi nous ne pouvons pas simplement optimiser chaque période indépendamment les unes des autres. Comme nous ne possédons pas de bonne approximation analytique pour une seule période, il existe encore moins de bonnes approximations analytiques pour l'évaluation d'une journée complète en tenant compte de l'effet transient entre les périodes.

BIBLIOGRAPHIE

[1] A. V. Aho, J. E. Hopcroft et J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, USA, 1974.

[2] R. K. Ahuja, T. L. Magnanti et J. B. Orlin. *Network Flows : Theory, Algorithms and Applications*. Prentice Hall, 1993.

[3] 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.

[4] A. N. Avramidis, W. Chan et P. L'Ecuyer. Staffing multi-skill call centers via search methods and a performance approximation. Soumis, 2006.

[5] A. N. Avramidis, A. Deslauriers et P. L'Ecuyer. Modeling daily arrivals to a telephone call center. *Management Science*, 50(7):896–908, 2004.

[6] S. Bhulai, G. Koole et G. Pot. Simple methods for shift scheduling in multi-skill call centers. Rapport technique, WS 2005-10, Free University, Amsterdam, 2005.

[7] A. J. Brigandi, D. R. Dargon, M. J. Sheehan et T. Spencer III. AT&T's call processing simulator (CAPS) operation desing for inbound call centers. *Interfaces*, 24 (1):6–28, Janvier-février 1994.

[8] 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.

[9] E. Buist. Conception et implantation d'une librairie pour la simulation de centres de contacts. Mémoire de maîtrise, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, 2005.

[10] E. Buist et P. L'Ecuyer. *ContactCenters : A Java Library for Simulating Contact Centers*, 2005. Guide d'utilisateur, disponible sur `http://www.ericbuist.com/contactcenters/`.

[11] E. Buist et P. L'Ecuyer. A Java library for simulating contact centers. Dans *Proceedings of the 2005 Winter Simulation Conference*, pages 556–565. IEEE Press, 2005.

[12] U.S. Census Bureau. Sector 56 : Administrative and support and waste management and remediation services : Establishment and firm, 2002. 1997 Economic Census. Tel que rapporté sur `http://www.census.gov`.

[13] U.S. Census Bureau. Sector 56 : Administrative and support and waste management and remediation services : Establishment and firm, 2005. 2002 Economic Census. Tel que rapporté sur `http://www.census.gov`.

[14] M. T. Ceẓik et P. L'Ecuyer. Staffing multiskill call centers via linear programming and simulation. *Management Science*, 2006. À paraître.

[15] P. 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.

[16] R. B. Cooper. *Introduction to Queueing Theory*. North-Holland, New York, deuxième édition, 1981.

[17] 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/archive/FRN/Decisions/2000/DT2000-24.htm`.

[18] Conseil de la radiodiffusion et des télécommunications canadiennes (CRTC). Règles de télémarketing par téléphone et par télécopieur, 2004. Voir `http://www.crtc.gc.ca/FRN/NEWS/RELEASES/2004/i040521.htm`.

[19] DRA Systems. OR-Objects 1.2.4, 2000. Librairie mathématique en Java. Voir `http://www.opsresearch.com`.

[20] G. J. Franx, G. Koole et A. Pot. Approximating multi-skill blocking systems by hyper-exponential decomposition. *Performance Evaluation*, 63:799–824, 2006.

[21] N. Gans, G. Koole et A. Mandelbaum. Telephone call centers : Tutorial, review, and research prospects. *Manufacturing and Service Operations Management*, 5: 79–141, 2003.

[22] 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.

[23] F. S. Hillier et G. J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, huitième édition, 2005.

[24] W. Hoschek. *The Colt Distribution : Open Source Libraries for High Performance Scientific and Technical Computing in Java*. CERN, Geneva, 2004. Disponible sur `http://dsd.lbl.gov/~hoschek/colt/`.

[25] ILOG, Inc. ILOG CPLEX : High-performance software for mathematical programming and optimization, 2006. Voir `http://www.ilog.com/products/cplex/`.

[26] 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.

[27] G. Jongbloed et G. Koole. Managing uncertainty in call centers using Poisson mixtures. *Applied Stochastic Models in Business and Industry*, 17:307–318, 2001.

[28] 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.

[29] G. Koole. Call center mathematics. En préparation, 2005.

[30] 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.

[31] G. Koole et J. Talim. Exponential approximation of multi-skill call centers architecture. Dans *Proceedings of QNETs*, pages 23/1–10, 2000.

[32] P. L'Ecuyer. *SSJ : A Java Library for Stochastic Simulation*, 2004. Guide d'utilisateur, Disponible sur `http://www.iro.umontreal.ca/~lecuyer/`.

[33] V. Mehrotra. Ringing up big business. *ORMS Today*, 24(4):18–24, Octobre 1997.

[34] NovaSim. ccProphet — simulate your call center's performance, 2003. Voir `http://www.novasim.com/CCProphet/`.

[35] J. Riordan. *Stochastic Service Systems*. John Wiley & Sons Inc., New York, 1962. The SIAM series in applied mathematics.

[36] Rockwell Automation, Inc. Arena simulation, 2005. Voir `http://www.arenasimulation.com`.

[37] Sun Microsystems, Inc. Java Technology, 2006. Voir `http://java.sun.com`.

[38] H. M. Taylor et S. Karlin. *An Introduction to Stochastic Modeling*. Academic Press, San Diego, troisième édition, 1998.

[39] R. B. Wallace et W. Whitt. A staffing algorithm for call centers with skill-based routing. *Manufacturing & Service Operation Management*, 7(4):276–294, 2005.

[40] R. W. Wolff. Poisson arrivals see time averages. *Operations Research*, 30:223–231, 1982.

ANNEXE I

ALGORITHME DÉTAILLÉ DE LA RECHERCHE RANDOMISÉE

Cette section présente en détails l'algorithme par défaut de la recherche randomisée (RS), introduit à la section 4.4. La phase 1 de l'algorithme est représentée par la fonction *RechercheRandomisée* qui appelle les procédures *Initialisation*, *Retirer* et *Déplacer*. La phase 2 est composée des procédures *SimAjouter* et *SimRetirer*, respectivement, pour corriger des erreurs d'approximation. En plus du SL, ces procédures utilisent la mesure $f_{i,j}$ qui est le nombre d'appels de type i servis par le groupe j.

```
vecteur fonction RechercheRandomisée
    global x = Initialisation(β, ξ)
    global q* = ∞ et q*_j = ∞, j ∈ M
    terminé = faux
    tant que terminé = faux
        Générer une taille de mouvement q ≥ 1
        si q < q*
            Retirer(q)
        sinon
            terminé = Déplacer(q)
        fin si
    fin tant que
    retourner x
fin fonction
```

Figure I.1 – La fonction *RechercheRandomisée* retourne un vecteur d'affectation réalisable, selon l'évaluateur utilisé.

vecteur fonction *Initialisation*(β, ξ)
// Distribuer les taux d'arrivée
pour $\forall i \in \mathscr{N}$
si $|\mathscr{R}_i| = 1$ (seulement un groupe peut servir)
$\gamma_{i,r_i(1)} = \lambda_i$
sinon
Déterminer le groupe j^* tel que $c_{j^*} \leq c_k, \forall k \in \mathscr{R}_i$
$\gamma_{i,j^*} = \lambda_i \times \beta$
$\gamma_{i,j} = \dfrac{\lambda_i - \gamma_{i,j^*}}{\mathscr{R}_i - 1}, \forall j \in \mathscr{R}_i \backslash \{j^*\}$
fin si
fin pour
pour $\forall j \in \mathscr{M}$
// Agréger les données
$\bar{\gamma} = \sum_{i \in \mathscr{K}_j} \gamma_{i,j}$
$\bar{\mu} = \left(\sum_{i \in \mathscr{K}_j} \frac{\gamma_{i,j}}{\bar{\gamma}\mu_{i,j}} \right)^{-1}$
$\bar{\tau} = \sum_{i \in \mathscr{K}_j} \frac{\gamma_{i,j}\tau_i}{\bar{\gamma}}$
$x_j = \arg\min_{s \geq 0}\{D(\bar{\mu}, \bar{\tau}) \geq \xi, \text{ pour un système } M/M/s : \bar{\gamma}/\bar{\mu}/s\}$
fin pour
// Assurer la réalisabilité de la solution initiale
tant que x est *irréalisable*
si $g(\mathbf{x}) < l$
$j^* = \arg\max_{j \in \mathscr{M}} \left\{ \dfrac{\sum_{i \in \mathscr{K}_j} f_{i,j}/\mu_{i,j}}{c_j x_j} \right\}$
sinon
$i^* = \arg\max_{i \in \mathscr{N}} \{l_i - g_i(\mathbf{x})\}$
$j^* = \arg\max_{j \in \mathscr{M}} \left\{ \dfrac{f_{i^*,j}/\mu_{i^*,j}}{\sum_{i \in \mathscr{K}_j} f_{i,j}/\mu_{i,j}} \right\}$
fin si
$\mathbf{x} = \mathbf{x} + \mathbf{e}_{j^*}$
fin tant que
retourner x
fin fonction

Figure I.2 – La fonction *Initialisation* de RS retourne un vecteur d'affectation réalisable des agents.

fonction *Retirer*(q)
// La solution courante $\mathbf{x}$ est une variable globale accessible.
// Trouver l'ensemble des groupes ayant au moins q agents
$\mathscr{C} = \{j : x_j \geq q, j \in \mathscr{M}\}$
si $|\mathscr{C}| = 0$
$q^* = q$
retourner
fin si
pour $\forall j \in \mathscr{C}$
Évaluer le vecteur $\mathbf{x}'_j = \mathbf{x} - q\mathbf{e}_j$ avec LD
si $\mathbf{x}'_j$ est *réalisable*
$\mathscr{F} = \mathscr{F} \cup \{j\}$
fin si
fin pour
si $|\mathscr{F}| = 0$
$q^* = q$
retourner
fin si
// Choisir le meilleur voisin réalisable
$j^* = \arg\min_{j \in \mathscr{F}} \left\{ \frac{g(\mathbf{x}) - g(\mathbf{x}'_j)}{c_j} \right\}$
$\mathbf{x} = \mathbf{x}'_{j^*}$
retourner
fin fonction

Figure I.3 – La fonction *Retirer* enlève q agents d'un groupe tout en gardant la réalisabilité de la solution.

booléen fonction *Déplacer*(q)
// La solution courante **x** est une variable globale accessible.
// Trouver l'ensemble des groupes ayant au moins q agents à déplacer
$\mathscr{P} = \{j : x_j \geq q, q < q_j^*, j \in \mathscr{M}\}$
si $|\mathscr{P}| = 0$
si $q = 1$, **retourner vrai**, **fin si**
retourner faux
fin si
Choisir un groupe pivot $p \in \mathscr{P}$
$\mathscr{G} = \{j : c_j < c_p\}$, les groupes moins chers que p
si $|\mathscr{G}| = 0$
$q_p^* = q$
retourner faux
fin si
pour $\forall j \in \mathscr{G}$
Évaluer le vecteur $\mathbf{x}'_j = \mathbf{x} - q\mathbf{e}_p + q\mathbf{e}_j$ avec LD
si $\mathbf{x}'_j$ est *réalisable*
$\mathscr{F} = \mathscr{F} \cup \{j\}$
fin si
fin pour
si $|\mathscr{F}| = 0$
$q_p^* = q$
retourner faux
fin si
// Choisir le meilleur voisin réalisable
$j^* = \arg\min_{j \in \mathscr{F}} \left\{ \frac{g(\mathbf{x}) - g(\mathbf{x}'_j)}{c_p - c_j} \right\}$
$\mathbf{x} = \mathbf{x}'_{j^*}$
retourner faux
fin fonction

Figure I.4 – La fonction *Déplacer* envoie q agents d'un groupe vers un groupe moins dispendieux tout en gardant la réalisabilité de la solution.

fonction *SimAjouter*
    // La solution courante **x** est une variable globale accessible.
    tant que x est *irréalisable*
        // Corriger par type d'appel avant
        si $\max\{l_i - g_i(\mathbf{x}) : i \in \mathscr{N}\} > 0$
            $i^* = \arg\max_{i\in\mathscr{N}}\{l_i - g_i(\mathbf{x})\}$
            $\mathscr{C} = \mathscr{R}_{i^*} \cap \{j : x_j > 0, j \in \mathscr{M}\}$
            $j^* = \arg\max_{j\in\mathscr{C}} \left\{ \frac{f_{i^*,j}/\mu_{i^*,j}}{\sum_{k\in\mathscr{K}_j} f_{k,j}/\mu_{k,j}} \right\}$ (où f est le nombre d'appels servis)
        sinon (la contrainte globale n'est pas satisfaite)
            $j^* = \arg\max \left\{ \frac{\sum_{i\in\mathscr{K}_j} f_{i,j}/\mu_{i,j}}{c_j x_j} : x_j > 0, j \in \mathscr{M} \right\}$
        fin si
        $\mathbf{x} = \mathbf{x} + \mathbf{e}_{j^*}$
    fin tant que
fin fonction

Figure I.5 – La fonction *SimAjouter* ajoute un agent jusqu'à l'obtention d'une solution réalisable.

fonction *SimRetirer*
// La solution courante **x** est une variable globale accessible.
// La liste $\mathscr{L}$ contient les groupes ayant la possibilité d'être réduits.
$\mathscr{L} = \{j : x_j \geq 1, j \in \mathscr{M}\}$
tant que $|\mathscr{L}| > 0$
$d_{i,j} = \dfrac{f_{i,j}/\mu_{i,j}}{\sum_{k\in\mathscr{K}_j} f_{k,j}/\mu_{k,j}}, i \in \mathscr{K}_j, j \in \mathscr{L}$
$\chi_j = \sum_{i\in\mathscr{K}_j} d_{i,j}[g_i(\mathbf{x}) - l_i]$
Soit π une permutation de $\mathscr{L}$ en ordre décroissant : $c_{\pi(1)}\chi_{\pi(1)} \geq c_{\pi(2)}\chi_{\pi(2)} \geq \ldots \geq c_{\pi(|\mathscr{L}|)}\chi_{\pi(|\mathscr{L}|)}$
pour $i = 1$ à $|\mathscr{L}|$
$\mathbf{x}'_{\pi(i)} = \mathbf{x} - \mathbf{e}_{\pi(i)}$
si $\mathbf{x}'$ est *réalisable*
$\mathbf{x} = \mathbf{x}'_{\pi(i)}$
si $x_{\pi(i)} = 0, \mathscr{L} = \mathscr{L}\backslash\{\pi(i)\}$, **fin si**
continuer tant que
sinon
$\mathscr{L} = \mathscr{L}\backslash\{\pi(i)\}$
fin si
fin pour
fin tant que
fin fonction

Figure I.6 – La fonction *SimRetirer* retire les agents par une procédure gloutonne jusqu'à un minimum local.

ANNEXE II

GUIDE D'UTILISATEUR DES LOGICIELS

Ceci est un guide d'utilisation des logiciels d'optimisation pour l'affectation des agents dans un centre d'appels téléphoniques pour une seule période (voir la section 2.3). Les deux algorithmes sont : l'optimisation par la programmation linéaire et la simulation (chapitre 3) et la recherche randomisée accompagnée de l'approximation perte et délai (chapitre 4).

Ces logiciels sont programmés en Java et ils sont regroupés dans un paquetage nommé : `umontreal.iro.lecuyer.ccoptim`. Ce dernier est divisé en plusieurs sous-groupes : `cp` (l'algorithme CP), `rs` (l'algorithme RS), `approx` (les fonctions d'approximation) et `util` (des fonctions utilitaires).

II.1 Installation

Pour utiliser les logiciels, l'usager a besoin d'installer le logiciel Java 5.0 [37] ou une version plus récente. Des librairies externes Java sont également nécessaires : les librairies de simulation SSJ [32] et ContactCenters [10], et la librairie de calculs mathématiques Colt [24]. Pour CP, il faut en plus un solutionneur linéaire externe. Pour l'instant, le programme CP est compatible avec les solutionneurs Cplex 8 et 9 [25] et OR-Object 1.2.4 [19] qui est gratuit, mais ne résout pas les problèmes en nombres entiers. Nous laissons à l'utilisateur l'installation de ces libraires et de créer ou modifier la variable d'environnement `CLASSPATH` en conséquence.

II.2 Exécution des programmes

Si l'usager a défini l'emplacement des libraires dans la variable d'environnement `CLASSPATH`, les logiciels peuvent être exécutés dans n'importe quel répertoire.

II.2.1 Optimisation par la programmation linéaire et la simulation

Le logiciel d'optimisation par la programmation linéaire et la simulation (CP) requiert 3 fichiers de paramètres qui sont détaillés dans la section II.3.

La commande pour exécuter le programme :

```
java umontreal.iro.lecuyer.ccoptim.cp.CuttingPlaneOptimizer ccParams.xml
simParams.xml cpParams.xml
```

où

— `ccParams.xml` est le fichier de configuration du centre d'appels.

— `simParams.xml` est le fichier de paramètres du simulateur.

— `cpParams.xml` est le fichier de paramètres du programme d'optimisation ; se référer à la section II.3.3.

II.2.2 Recherche randomisée

Le programme de la recherche randomisée (RS) requiert 4 fichiers de paramètres qui sont détaillés dans la section II.3.

La commande pour exécuter le programme :

```
java umontreal.iro.lecuyer.ccoptim.rs.RandomizedSearch ccParams.xml
ldParams.xml simParams.xml rsParams.xml
```

où

— `ccParams.xml` est le fichier de configuration du centre d'appels.

— `ldParams.xml` est le fichier de paramètres de l'approximation perte et délai (LD).

— `simParams.xml` est le fichier de paramètres du simulateur. Ce fichier est utilisé seulement par le simulateur.

— `rsParams.xml` est le fichier de paramètres de la recherche randomisée ; se référer à la section II.3.4.

II.2.3 Approximation perte et délai

Le programme d'approximation perte et délai (LD) requiert 2 fichiers de paramètres qui sont détaillés dans la section II.3.

La commande pour exécuter le programme :

```
java umontreal.iro.lecuyer.ccoptim.approx.LossDelayApprox ccParams.xml
ldParams.xml
```

où

— `ccParams.xml` est le fichier de configuration du centre d'appels.

— `ldParams.xml` est le fichier de paramètres de l'approximation perte et délai (LD).

II.3 Fichiers de paramètres

Tous les fichiers de paramètres utilisent le format XML. Un paramètre peut être du type booléen (`true` ou `false`), un nombre entier, un nombre réel ou du texte. Un paramètre peut aussi être un tableau de ces 4 types de données. Dans la description détaillée, le type de chaque paramètre est spécifié selon la légende : booléen (B), nombre entier (E), nombre réel (R) et texte (T).

— Pour écrire un paramètre, il faut encapsuler la valeur par le nom du paramètre.
Exemple : `<verbose> true </verbose>`

— Pour un tableau, les valeurs sont séparées à l'aide de la virgule (,) :
Exemple : `<arrivalRates> 10, 9, 10 </arrivalRates>`

— Pour mettre des commentaires, il ne suffit que de les encapsuler comme ceci :
`<!--` *commentaires* `-->`.
Exemple, pour mettre le paramètre `changeHeuristicCut` en commentaire :
`<!-- <changeHeuristicCut>true</changeHeuristicCut> -->`

II.3.1 Paramètres du centre d'appels

Les logiciels d'optimisation, d'approximation et de simulation utilisent le fichier de configuration du centre d'appels. Une documentation détaillée est disponible dans le guide d'utilisateur de la librairie ContactCenters [10].

II.3.2 Paramètres du simulateur

Ce fichier contient les paramètres de simulation et il est utilisé uniquement par le simulateur. Une documentation détaillée est disponible dans le guide d'utilisateur de la librairie ContactCenters [10], voir la section de `batchSimParams`. Ces paramètres permettent de contrôler la longueur et la qualité des simulations, il est important de comprendre leur utilisation. Nous incluons une description rapide de certains paramètres pour simuler un nombre fixe de simulations pour un système en régime permanent (*steady-state*), se referer à la documentation officielle pour plus d'information :

- `level` (R) : le niveau des intervalles de confiance lors du calcul des erreurs relatives et de l'affichage des résultats de la simulation. Valeur par défaut : `0.95`.
- `initNonEmpty` (B) : si `true`, la simulation sera initialisée avec $m \times$ `targetInitOccupancy` agents occupés et toutes les files d'attente vides, où m est le nombre total d'agents. Nous suggérons la valeur `true`.
- `targetInitOccupancy` (R) : si le simulateur débute dans un état *non vide*, selon `initNonEmpty`, la proportion des agents occupés correspond à `targetInitOccupancy`. Valeur par défaut : `0.9`.
- `batchSize` (R) : la taille d'un lot (*batch*) en nombre de périodes de simulation. La durée d'une période est définie dans le fichier de configuration du centre d'appels.
- `minBatches` (E) : nombre de lots à simuler et nombre minimal dans le cas d'un contrôle dynamique. Le nombre total de périodes simulées est : `batchSize`×(`minBatches`+`warmupBatches`).
- `warmupBatches` (E) : nombre de lots pour le réchauffement du simulateur. Les données statistiques ne sont pas recueillies durant ces périodes.

Exemple d'un fichier de paramètres du simulateur :

```
<batchSimParams
   level="0.95"
   targetInitOccupancy="0.9"
   initNonEmpty="true"
   batchSize="10"
   minBatches="10"
   warmupBatches="2">
</batchSimParams>
```

Listing II.1 – Exemple d'un fichier de paramètres du simulateur

II.3.3 Paramètres de l'optimisateur par la programmation linéaire et la simulation

Ce fichier de paramètres est seulement utilisé par le programme d'optimisation CP. Nous suggérons de se référer au chapitre 3 pour plus d'information sur certains paramètres. Le vecteur **c** de coût par groupe d'agents peut être défini à l'aide du paramètre `busyCost` dans le fichier de configuration du centre d'appels ou par `skillCost` dans ce fichier pour une attribution automatique du coût. Les paramètres importants sont suivis du symbole "*".

- `currentPeriod` (E) : sélectionner la période à optimiser. La première période correspond à 0.
- `skillCost` (R) : si le paramètre `busyCost` est > 0 dans le fichier de configuration du centre d'appels, le vecteur de coût des groupes d'agents prend les valeurs de `busyCost`. Sinon, le coût d'un agent est défini selon la règle suivante : $1 + \texttt{skillCost} \times (k-1)$, où k est le nombre d'habiletés de l'agent.
- `minEachQosH` (R) : les SL de tous les types d'appels doivent être au minimum à cette valeur pour que le programme génère des coupes par sous-gradients ; se référer à la section 3.3.1. Dans le cas où le centre d'appels est contraint uniquement sur le SL global, cette valeur peut être 0. Valeur par défaut : `0.1`.

— `cutPriority` (E) : si égal à 1, la priorité est donnée aux coupes par sous-gradients lorsqu'ils ne sont pas nuls. Autrement, la priorité est donnée aux coupes heuristiques, voir la section 3.3. Valeur par défaut : `0` (coupes heuristiques).

— `defaultSubStep*` (E) : la taille du pas par défaut pour générer les sous-gradients par la méthode des différences finies, ceci correspond à la variable *d* de la section 3.3.3. Valeur par défaut : `1`, mais elle dépend du centre d'appels. Nous recommendons une valeur ≥ 2 pour les courtes simulations ou les grands centres d'appels.

— `lowOverallSubStep*` (E) : quand le SL global est inférieur à `pivotOverallQos`, l'algorithme utilise un pas *d* de cette valeur au lieu de `defaultSubStep`. La raison est que lorsque le SL est faible, le bruit de la simulation risque d'être plus élevé. Valeur par défaut : `2`, mais elle doit être supérieure à `defaultSubStep`.

— `pivotOverallQos` (R) : si le SL global est inférieur à cette valeur, alors le programme utilisera un pas *d* égal à `lowOverallSubstep` au lieu de `defaultSubStep`. La raison est que lorsque le SL est faible, le bruit de la simulation risque d'être plus élevé. Cette valeur devrait être à mi-chemin entre 0 et le seuil du SL global.

— `pivotPerCallQos` (R) : si le SL global est égal ou supérieur à cette valeur et qu'il existe un type d'appel dont le SL ne satisfait pas son seuil, alors le programme générera un sous-gradient par type d'appel au lieu du niveau global. La justification de ce paramètre est que l'amélioration du SL par type d'appel augmente aussi le SL global. Par contre, le SL par type d'appel possède plus de bruit de simulation au début de l'algorithme. Cette valeur devrait être proche du seuil du SL global.

— `loadCoefficient*` (tableau de R) : coefficients des charges des types d'appels ; se référer aux sections 3.2.1 et 3.2.2. Valeur par défaut : `1.0`. Réduire légèrement cette valeur lorsqu'il y a présence d'abandons. Il est à noter que cette valeur peut affecter le taux d'abandon des appels. Si le tableau contient plus d'entrées que de types d'appels (*n*), seulement les *n* premières seront utilisées.

— `changeHeuristicCut` (B) : si `true`, alors le programme vérifiera l'impact de la dernière coupe heuristique ajoutée ; se référer à la section 3.3.1. Puisque

nous voulons optimiser par les coupes basées sur les sous-gradients, le programme veillera à ce que la coupe heuristique n'augmente le SL d'au plus 10%. Valeur par défaut : `true`.

— `solverClass` (T) : le nom de la classe et du paquetage de l'interface au solutionneur linéaire à utiliser sans le the suffixe '`.class`'. Les classes d'interface disponibles sont : OR-Objects 1.2.4, Cplex 8 et Cplex 9.
  — `umontreal.iro.lecuyer.ccoptim.cp.OR124Solver`
  — `umontreal.iro.lecuyer.ccoptim.cp.Cplex8Solver`
  — `umontreal.iro.lecuyer.ccoptim.cp.Cplex9Solver`

— `solveIP` (B) : dépend du solutionneur `solverClass` utilisé. Mettre la valeur à `true` pour résoudre les problèmes en nombres entiers. Il est à noter que le temps d'exécution peut être très élevé dépendamment de la difficulté du problème. Mettre à `false` pour des faibles budgets de temps.

— `solveWithMaxFlowCut` (B) : si `true`, alors le programme utilisera l'initialisation de la section 3.2.1, sinon il utilisera les équations de la section 3.2.2. Si les durées de service dépendent à la fois du type d'appel et du groupe d'agent, il faut utiliser les équations de la section 3.2.2 et mettre la valeur à `false`.

— `verbose` (B) : si `false`, alors le programme imprimera seulement la solution finale avec les résultats de la simulation et le coût. Si la valeur est `true`, alors le programme imprimera en plus une trace de toutes les itérations et des coupes linéaires ajoutées. Valeur par défaut : `false`.

Exemple d'un fichier de paramètres du logiciel CP :

```
<cuttingPlaneParams
   currentPeriod="0"            skillCost="0.05"
   minEachQosH="0.1"            cutPriority="0"
   defaultSubStep="2"           lowOverallSubStep="3"
   pivotOverallQos="0.65"       pivotPerCallQos="0.70"
   changeHeuristicCut="true"    solveIP="false"
   solverClass="umontreal.iro.lecuyer.ccoptim.cp.OR124Solver"
   solveWithMaxFlowCut="false"
   verbose="false" >
```

```
   <loadCoefficient>
      <row repeat="65">1</row>
   </loadCoefficient>
</cuttingPlaneParams>
```

Listing II.2 – Exemple d'un fichier de paramètres du logiciel CP

II.3.4 Paramètres pour la recherche randomisée

Ce fichier de paramètres est utilisé seulement par le logiciel RS. Nous suggérons de se référer au chapitre 4 pour des informations additionnelles sur certains paramètres.

- `findInitialStaffing` (B) : si `true`, le programme déterminera automatiquement la solution initiale (à l'aide des paramètres `rootL` et `lowerCostRatio`), voir la section 4.4.1. Si `false`, alors le programme utilisera l'affectation des agents définie dans le fichier de configuration du centre d'appels comme la solution initiale. Valeur par défaut : `true`.
- `useBusyCost` (B) : si `false`, le coût d'un agent sera déterminé en fonction du nombre d'habiletés qu'il possède (k) et son coût sera : $1 + \texttt{skillPremiumCost} \times (k-1)$. Si `true`, alors le vecteur de coût des agents sera déterminé par les paramètres `busyCost` définis dans le fichier de configuration du centre d'appels.
- `skillPremiumCost` (R) : le coût pour chaque habileté additionnel d'un agent.
- `maxRSCPUsec` (E) : le temps total maximal en secondes pour l'initialisation et la phase-1 de RS sur tous les départs multiples. En général, l'utilisateur n'aura pas besoin de cette contrainte. Nous suggérons de donner une très grande valeur.
- `rootL` (tableau de R) : correspond au paramètre ξ dans la section 4.4.1. Mettre plusieurs valeurs pour exécuter des démarrages multiples. Nous suggérons des valeurs proches du seuil du SL global.
- `lowestCostRatio` (tableau de R) : correspond au paramètre β dans la section 4.4.1. Une valeur de `0` initialisera avec uniquement des généralistes, alors qu'avec `1`, il n'y aura que des spécialistes. Mettre plusieurs valeurs pour exécu-

ter des démarrages multiples. Nous suggérons des valeurs autour de `0.8`.

— `randomSeeds` (tableau de 6 E) : définit les germes du générateur de nombres aléatoires utilisé par RS (générateur `MRG32k3a` [32]), mais ceci n'affecte pas le simulateur. Le générateur a besoin de 6 nombres entiers. Les germes par défaut seront utilisés si ce paramètre est omis.

— `verbose` (B) : si `false`, le programme imprimera seulement la solution finale, les niveaux de service et le coût final. Si `true`, alors le programme imprimera l'action à chaque itération : le coût, le SL global, la procédure utilisée et le nombre total d'agents.

Exemple d'un fichier de paramètres du logiciel RS :

```
<randomizedSearchParams
   findInitialStaffing="true"                useBusyCost="false"
   skillPremiumCost="0.05"                   maxRSCPUsec="300000000"
   verbose="true">

   <rootL> 0.8</rootL>
   <lowestCostRatio> 0.5</lowestCostRatio>
   <randomSeeds> 19992,6210,1414,595,56558,2146 </randomSeeds>
</randomizedSearchParams>
```

Listing II.3 – Exemple d'un fichier de paramètres du logiciel RS

II.3.5 Paramètres pour l'approximation LD

Ce fichier contient les paramètres pour l'approximation LD. Cette approximation ne peut évaluer qu'une seule période à la fois. Si l'utilisateur veut seulement évaluer une affectation d'agents, il faut entrer le vecteur d'affectation dans le fichier de configuration du centre d'appels.

— `currentPeriod` (E) : correspond à la période à évaluer. La période `0` correspond à la première période.

— `maxIteration` (E) : nombre maximal d'itérations, correspond au paramètre κ dans la section 4.3.5. Valeur par défaut : `400`.

— `absErrorTolerance` (R) : le niveau de tolérance d'erreur, correspond au paramètre ε dans la section 4.3.5. Valeur par défaut : `1e-4` (10^{-4}).

— `queueAbandon` (B) : devrait être défini selon le centre d'appels. Si `true`, alors le programme utilise les formules avec abandons, sinon il utilise les formules sans abandon.

— `queueCapacity` (E) : capacité de la file d'attente de chaque groupe d'agents. Ce paramètre est nécessaire si `queueAbandon` est à `true`. Une valeur < 0 utilisera une sélection automatique (se référer à la section 4.3.5) : $\delta\sqrt{m}$, où m est le nombre d'agents dans le groupe et δ est le paramètre `delta`. Valeur par défaut : `-1`.

— `delta` (R) : utilisé dans la sélection automatique de la capacité des files d'attente, voir le paramètre `queueCapacity`. Valeur par défaut : `2`.

— `verbose` (B) : si `true`, le programme imprimera les résultats à chaque évaluation. Nous suggérons de mettre à `false` lorsque utilisé avec les logiciels d'optimisation.

Exemple d'un fichier de paramètres du logiciel LD :

```
<lossDelayParams
   currentPeriod="0"              maxIteration="400"
   absErrorTolerance="1e-4"       queueAbandon="true"
   queueCapacity="-1"             delta="2"
   verbose="false"/>
```

Listing II.4 – Exemple d'un fichier de paramètres du logiciel LD