The Cross-Entropy Method for Optimization

Zdravko I. Botev, Department of Computer Science and Operations Research, Université de Montréal, Montréal Québec H3C 3J7, Canada.
botev@iro.umontreal.ca

Dirk P. Kroese, School of Mathematics and Physics, The University of Queensland, Brisbane 4072, Australia.
kroese@maths.uq.edu.au.

Reuven Y. Rubinstein, Faculty of Industrial Engineering and Management, Technion, Haifa, Israel.
ierrr01@ie.technion.ac.il

Pierre L'Ecuyer, Department of Computer Science and Operations Research, Université de Montréal, Montréal Québec H3C 3J7, Canada.
lecuyer@iro.umontreal.ca

Abstract

The cross-entropy method is a versatile heuristic tool for solving difficult estimation and optimization problems, based on Kullback–Leibler (or cross-entropy) minimization. As an optimization method it unifies many existing population-based optimization heuristics. In this chapter we show how the CE method can be applied to a diverse range of combinatorial, continuous, and noisy optimization problems.

1 Introduction

The *cross-entropy (CE) method* was proposed by Rubinstein (1997) as an adaptive importance sampling procedure for the estimation of rare-event probabilities, that uses the *cross-entropy* or *Kullback–Leibler divergence* as a measure of closeness between two sampling distributions. Subsequent work by Rubinstein (1999; 2001) has shown that many optimization problems can be translated into a rare-event estimation problem. As a result, adaptive importance sampling methods such as the CE method can be utilized as randomized algorithms for optimization. The gist of the idea is that the probability of locating an optimal or near optimal solution using naive random search is a rare-event probability. The cross-entropy method can be used to gradually change the sampling distribution of the random search so that the rare-event is more likely to occur. For this purpose, using the CE distance, the method estimates a sequence of sampling distributions that converges to a distribution with probability mass concentrated in a region of near-optimal solutions.

To date, the CE method has been successfully applied to mixed integer nonlinear programming (Kothari and Kroese 2009); continuous optimal control problems (Sani 2009, Sani and Kroese 2008); continuous multi-extremal optimization (Kroese et al. 2006); multidimensional independent component analysis (Szabó et al. 2006); optimal policy search (Busoniu et al. 2010); clustering (Botev and Kroese 2004, Kroese et al. 2007b, Boubezoula et al. 2008);

signal detection (Liu et al. 2004); DNA sequence alignment (Keith and Kroese 2002, Pihur et al. 2007); noisy optimization problems such as optimal buffer allocation (Alon et al. 2005); resource allocation in stochastic systems (Cohen et al. 2007); network reliability optimization (Kroese et al. 2007a); vehicle routing optimization with stochastic demands (Chepuri and Homem-de-Mello 2005); power system combinatorial optimization problems (Ernst et al. 2007); and neural and reinforcement learning (Lörincza et al. 2008, Menache et al. 2005, Unveren and Acan 2007, Wu and Fyfe 2008).

A tutorial on the CE method is given in de Boer et al. (2005). A comprehensive treatment can be found in Rubinstein and Kroese (2004); see also Rubinstein and Kroese (2007; Chapter 8). The CE method homepage is `www.cemethod.org`.

2 From Estimation to Optimization

The CE method can be applied to two types of problems:

1. **Estimation**: Estimate $\ell = \mathbb{E}[H(\mathbf{X})]$, where $\mathbf{X}$ is a random object taking values in some set $\mathscr{X}$ and H is a function on $\mathscr{X}$. An important special case is the estimation of a probability $\ell = \mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$, where S is another function on $\mathscr{X}$.

2. **Optimization**: Optimize (that is, maximize or minimize) a given objective function $S(\mathbf{x})$ over all $\mathbf{x} \in \mathscr{X}$. S can be either a known or a *noisy* function. In the latter case the objective function needs to be estimated, e.g., via simulation.

In this section we review the CE method as an adaptive importance sampling method for rare-event probability estimation, and in the next section we show how the CE estimation algorithm leads naturally to an optimization heuristic.

Cross-Entropy for Rare-event Probability Estimation

Consider the estimation of the probability

$$\ell = \mathbb{P}(S(\mathbf{X}) \geqslant \gamma) = \mathbb{E}[\mathrm{I}_{\{S(\mathbf{X}) \geqslant \gamma\}}] = \int \mathrm{I}_{\{S(\mathbf{x}) \geqslant \gamma\}} \, f(\mathbf{x}; \mathbf{u}) \, \mathrm{d}\mathbf{x} \,, \tag{1}$$

where S is a real-valued function, γ is a threshold or level parameter, and the random variable $\mathbf{X}$ has probability density function (pdf) $f(\cdot; \mathbf{u})$, which is parameterized by a finite-dimensional real vector $\mathbf{u}$. We write $\mathbf{X} \sim f(\cdot; \mathbf{u})$. If $\mathbf{X}$ is a discrete variable, simply replace the integral in (1) by a sum. We are interested in the case where ℓ is a *rare-event probability*; that is, a very small probability, say, less than 10^{-4}. Let g be another pdf such that $g(\mathbf{x}) = 0 \Rightarrow H(\mathbf{x}) \, f(\mathbf{x}; \mathbf{u}) = 0$ for every $\mathbf{x}$. Using the pdf g we can represent ℓ as

$$\ell = \int \frac{f(\mathbf{x}; \mathbf{u}) \, \mathrm{I}_{\{S(\mathbf{x}) \geqslant \gamma\}}}{g(\mathbf{x})} \, g(\mathbf{x}) \, \mathrm{d}\mathbf{x} = \mathbb{E}\left[\frac{f(\mathbf{X}; \mathbf{u}) \, \mathrm{I}_{\{S(\mathbf{X}) \geqslant \gamma\}}}{g(\mathbf{X})} \right], \quad \mathbf{X} \sim g \,. \tag{2}$$

Consequently, if $\mathbf{X}_1, \ldots, \mathbf{X}_N$ are independent random vectors with pdf g, written as $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim_{\text{iid}} g$, then

$$\widehat{\ell} = \frac{1}{N} \sum_{k=1}^{N} \mathrm{I}_{\{S(\mathbf{X}_k) \geqslant \gamma\}} \, \frac{f(\mathbf{X}_k; \mathbf{u})}{g(\mathbf{X}_k)} \tag{3}$$

is an unbiased estimator of ℓ: a so-called *importance sampling estimator*. It is well known (see, for example, (Rubinstein and Kroese 2007; Page 132)) that the optimal importance sampling pdf (that is, the pdf g^* for which the variance of $\widehat{\ell}$ is minimal) is the density of $\mathbf{X}$ conditional on the event $S(\mathbf{X}) \geqslant \gamma$; that is,

$$g^*(\mathbf{x}) = \frac{f(\mathbf{x}; \mathbf{u}) \, \mathrm{I}_{\{S(\mathbf{x}) \geqslant \gamma\}}}{\ell} \, . \tag{4}$$

The idea of the CE method is to choose the importance sampling pdf g from within the parametric class of pdfs $\{f(\cdot; \mathbf{v}), \ \mathbf{v} \in \mathcal{V}\}$ such that the Kullback–Leibler divergence between the optimal importance sampling pdf g^* and g is minimal. The Kullback–Leibler divergence between g^* and g is given by

$$\mathcal{D}(g^*, g) = \int g^*(\mathbf{x}) \ln \frac{g^*(\mathbf{x})}{g(\mathbf{x})} \, \mathrm{d}\mathbf{x} = \mathbb{E}\left[\ln \frac{g^*(\mathbf{X})}{g(\mathbf{X})} \right], \quad \mathbf{X} \sim g^* \, . \tag{5}$$

The CE minimization procedure then reduces to finding an optimal reference parameter vector, $\mathbf{v}^*$ say, by cross-entropy minimization:

$$\begin{aligned} \mathbf{v}^* &= \operatorname*{argmin}_{\mathbf{v}} \mathcal{D}(g^*, f(\cdot; \mathbf{v})) \\ &= \operatorname*{argmax}_{\mathbf{v}} \mathbb{E}_{\mathbf{u}} \mathrm{I}_{\{S(\mathbf{X}) \geqslant \gamma\}} \ln f(\mathbf{X}; \mathbf{v}) \\ &= \operatorname*{argmax}_{\mathbf{v}} \mathbb{E}_{\mathbf{w}} \mathrm{I}_{\{S(\mathbf{X}) \geqslant \gamma\}} \ln f(\mathbf{X}; \mathbf{v}) \frac{f(\mathbf{X}; \mathbf{u})}{f(\mathbf{X}; \mathbf{w})} \, , \end{aligned} \tag{6}$$

where $\mathbf{w}$ is any reference parameter and the subscript in the expectation operator indicates the density of $\mathbf{X}$. This $\mathbf{v}^*$ can be estimated via the stochastic counterpart of (6):

$$\widehat{\mathbf{v}} = \operatorname*{argmax}_{\mathbf{v}} \frac{1}{N} \sum_{k=1}^{N} \mathrm{I}_{\{S(\mathbf{X}) \geqslant \gamma\}} \frac{f(\mathbf{X}_k; \mathbf{u})}{f(\mathbf{X}_k; \mathbf{w})} \ln f(\mathbf{X}_k; \mathbf{v}) \, , \tag{7}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim_{\text{iid}} f(\cdot; \mathbf{w})$. The optimal parameter $\widehat{\mathbf{v}}$ in (7) can often be obtained in explicit form, in particular when the class of sampling distributions forms an *exponential family* (Rubinstein and Kroese 2007; Pages 319–320). Indeed, analytical updating formulas can be found whenever explicit expressions for the *maximal likelihood estimators* of the parameters can be found (de Boer et al. 2005; Page 36).

A complication in solving (7) is that for a rare-event probability ℓ most or all of the indicators $\mathrm{I}_{\{S(\mathbf{X}) \geqslant \gamma\}}$ in (7) are zero, and the maximization problem becomes useless. In that case a *multi-level* CE procedure is used, where a sequence of reference parameters $\{\widehat{\mathbf{v}}_t\}$ and levels $\{\widehat{\gamma}_t\}$ is constructed with the

goal that the former converges to $\mathbf{v}^*$ and the latter to γ. At each iteration t we simulate N independent random variables $\mathbf{X}_1, \ldots, \mathbf{X}_N$ from the currently estimated importance sampling density $f(\cdot; \widehat{\mathbf{v}}_{t-1})$ and let $\widehat{\gamma}_t$ be the $(1-\varrho)$-quantile of the performance values $S(\mathbf{X}_1), \ldots, S(\mathbf{X}_N)$, where ϱ is a user specified parameter called the *rarity parameter*. We then update the value of $\widehat{\mathbf{v}}_{t-1}$ to $\widehat{\mathbf{v}}_t$, where $\widehat{\mathbf{v}}_t$ is calculated using likelihood maximization (or equivalently cross entropy minimization) based on the $N^{\mathrm{e}} = \lceil \varrho N \rceil$ random variables for which $S(\mathbf{X}_i) \geqslant \widehat{\gamma}_t$.

This leads to the following algorithm (Rubinstein and Kroese 2007; Page 238).

Algorithm 2.1 (CE Algorithm for Rare-Event Estimation) *Given the sample size N and the parameter ϱ, execute the following steps.*

1. *Define $\widehat{\mathbf{v}}_0 = \mathbf{u}$. Let $N^{\mathrm{e}} = \lceil \varrho N \rceil$. Set $t = 1$ (iteration counter).*

2. *Generate $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim_{\mathrm{iid}} f(\cdot; \widehat{\mathbf{v}}_{t-1})$. Calculate $S_i = S(\mathbf{X}_i)$ for all i, and order these from smallest to largest: $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$. Let $\widehat{\gamma}_t$ be the sample $(1-\varrho)$-quantile of performances; that is, $\widehat{\gamma}_t = S_{(N-N^{\mathrm{e}}+1)}$. If $\widehat{\gamma}_t > \gamma$, reset $\widehat{\gamma}_t$ to γ.*

3. *Use the* **same** *sample $\mathbf{X}_1, \ldots, \mathbf{X}_N$ to solve the stochastic program (7), with $\mathbf{w} = \widehat{\mathbf{v}}_{t-1}$. Denote the solution by $\widehat{\mathbf{v}}_t$.*

4. *If $\widehat{\gamma}_t < \gamma$, set $t = t + 1$ and reiterate from Step 2; otherwise, proceed with Step 5.*

5. *Let $T = t$ be the final iteration counter. Generate $\mathbf{X}_1, \ldots, \mathbf{X}_{N_1} \sim_{\mathrm{iid}} f(\cdot; \widehat{\mathbf{v}}_T)$ and estimate ℓ via importance sampling, as in (3) with $\mathbf{u} = \widehat{\mathbf{v}}_T$.*

We now show how this estimation algorithm leads naturally to a simple optimization heuristic.

Cross-Entropy Method for Optimization

To see how Algorithm 2.1 can be used for optimization purposes, suppose that the goal is to find the maximum of $S(\mathbf{x})$ over a given set $\mathscr{X}$. Assume, for simplicity, that there is only one maximizer $\mathbf{x}^*$. Denote the maximum by γ^*, so that

$$S(\mathbf{x}^*) = \gamma^* = \max_{\mathbf{x} \in \mathscr{X}} S(\mathbf{x}) \,. \tag{8}$$

We can now associate with the above optimization problem the estimation of the probability $\ell = \mathbb{P}(S(\mathbf{X}) \geqslant \gamma)$, where $\mathbf{X}$ has some probability density $f(\mathbf{x}; \mathbf{u})$ on $\mathscr{X}$ (for example corresponding to the uniform distribution on $\mathscr{X}$) and γ is close to the unknown γ^*. Typically, ℓ is a rare-event probability, and the multi-level CE approach of Algorithm 2.1 can be used to find an importance sampling distribution that concentrates all its mass in a neighborhood of the point $\mathbf{x}^*$. Sampling from such a distribution thus produces optimal or near-optimal states. Note that, in contrast to the rare-event simulation setting,

the final level $\gamma = \gamma^*$ is generally not known in advance, but the CE method for optimization produces a sequence of levels $\{\widehat{\gamma}_t\}$ and reference parameters $\{\widehat{\mathbf{v}}_t\}$ such that ideally the former tends to the optimal γ^* and the latter to the optimal reference vector $\mathbf{v}^*$ corresponding to the point mass at $\mathbf{x}^*$; see, e.g., (Rubinstein and Kroese 2007; Page 251).

Algorithm 2.2 (CE Algorithm for Optimization)

1. *Choose an initial parameter vector* $\widehat{\mathbf{v}}_0$. *Let* $N^{\mathrm{e}} = \lceil \varrho N \rceil$. *Set* $t = 1$ *(level counter).*

2. *Generate* $\mathbf{X}_1, \ldots, \mathbf{X}_N \sim_{\mathrm{iid}} f(\cdot; \widehat{\mathbf{v}}_{t-1})$. *Calculate the performances* $S(\mathbf{X}_i)$ *for all* i, *and order them from smallest to largest:* $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$. *Let* $\widehat{\gamma}_t$ *be the sample* $(1-\varrho)$*-quantile of performances; that is,* $\widehat{\gamma}_t = S_{(N-N^{\mathrm{e}}+1)}$.

3. *Use the* **same** *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *and solve the stochastic program*

$$\max_{\mathbf{v}} \frac{1}{N} \sum_{k=1}^{N} I_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} \ln f(\mathbf{X}_k; \mathbf{v}) \; . \tag{9}$$

   Denote the solution by $\widehat{\mathbf{v}}_t$.

4. *If some stopping criterion is met, stop; otherwise, set* $t = t+1$, *and return to Step 2.*

Note that the estimation Step 5 of Algorithm 2.1 is missing in Algorithm 2.2, because in the optimization setting we are not interested in estimating ℓ per se. For the same reason the likelihood ratio term $f(\mathbf{X}_k; \mathbf{u})/f(\mathbf{X}_k; \widehat{\mathbf{v}}_{t-1})$ in (7) is missing in (9).

To run the algorithm one needs to propose a class of parametric sampling densities $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$, the initial vector $\widehat{\mathbf{v}}_0$, the sample size N, the rarity parameter ϱ, and a stopping criterion. Of these the most challenging is the selection of an appropriate class of parametric sampling densities $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$. Typically, there is not a unique parametric family and the selection is guided by the following competing objectives. First, the class $\{f(\cdot; \mathbf{v}), \mathbf{v} \in \mathcal{V}\}$ has to be flexible enough to include a reasonable parametric approximation to the optimal importance sampling density (4) for the estimation of the associated rare-event probability ℓ. Second, each density $f(\cdot; \mathbf{v})$ has to be simple enough to allow fast random variable generation and closed-form solutions to the (weighted) maximum likelihood estimation program (9). In many cases these two competing objectives are reconciled by using a standard statistical model for $f(\cdot; \mathbf{v})$, such as the multivariate Bernoulli or Gaussian densities with independent components of the vector $\mathbf{X} \sim f(\cdot; \mathbf{v})$. However, in some cases it is beneficial to consider the more complicated Bernoulli or Gaussian mixture models, in which the estimation program (9) is carried out using the EM algorithm (see Section 3.2). In the examples that follow we primarily use the multivariate Bernoulli and Gaussian densities for combinatorial and continuous optimization problems, respectively. Note, however, that special parameterizations may be needed for problems such

as the traveling salesman problem, where the optimization is carried out over a set of possible permutations; see Rubinstein and Kroese (2004) for more details.

In applying Algorithm 2.2, the following *smoothed updating* has proven useful. Let $\widetilde{\mathbf{v}}_t$ be the solution to (9) and $0 \leqslant \alpha \leqslant 1$ be a *smoothing parameter*, then for the next iteration we take the parameter vector

$$\widehat{\mathbf{v}}_t = \alpha\, \widetilde{\mathbf{v}}_t + (1-\alpha)\, \widehat{\mathbf{v}}_{t-1} \; . \tag{10}$$

Other modifications can be found in Kroese et al. (2006), Rubinstein and Kroese (2004), and Rubinstein and Kroese (2007). The effect that smoothing has on convergence is discussed in detail in Costa et al. (2007). In particular, it is shown that for discrete binary optimization problems with non-constant smoothing $\widehat{\mathbf{v}}_t = \alpha_t\, \widetilde{\mathbf{v}}_t + (1-\alpha_t)\, \widehat{\mathbf{v}}_{t-1},\ \alpha_t \in [0,1]$ the optimal solution is generated with probability 1 if the smoothing sequence $\{\alpha_t\}$ satisfies:

$$\sum_{t=1}^{\infty} \prod_{k=1}^{t} (1-\alpha_k)^n = \infty \; ,$$

where n is the length of the binary vector $\mathbf{x}$. Other convergence results can be found in Rubinstein and Kroese (2004) and Margolin (2005). Finally, significant speed up can be achieved by using a *parallel* implementation of CE (Evans et al. 2007).

3 Applications to Combinatorial Optimization

When the state space $\mathscr{X}$ is finite, the optimization problem (8) is often referred to as a *discrete* or *combinatorial optimization* problem. For example, $\mathscr{X}$ could be the space of combinatorial objects such as binary vectors, trees, paths through graphs, permutations, etc. To apply the CE method, one needs to first specify a convenient parameterized random mechanism to generate objects $\mathbf{X}$ in $\mathscr{X}$. An important example is where $\mathbf{X} = (X_1, \ldots, X_n)$ has independent components such that $X_i = j$ with probability $v_{i,j},\ i = 1, \ldots, n,\ j = 1, \ldots, m$. In that case, the solution of the program in (9) at the t-th iteration is

$$\widehat{v}_{t,i,j} = \frac{\sum_{k=1}^{N} \mathrm{I}_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} \mathrm{I}_{\{X_{k,i}=j\}}}{\sum_{k=1}^{N} \mathrm{I}_{\{S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}}} \; , \;\; i = 1, \ldots, n, \;\; j = 1, \ldots, m \; , \tag{11}$$

where $\mathbf{X}_1, \ldots, \mathbf{X}_N$ are independent copies of $\mathbf{X} \sim \{\widehat{v}_{t-1,i,j}\}$ and $X_{k,i}$ is the i-th element of $\mathbf{X}_k$ (see de Boer et al. (2005; Page 56)). We call the set of vectors satisfying $S(\mathbf{X}_k) \geqslant \widehat{\gamma}_t$ the *elite sample* at iteration t. Thus, the updated probability $\widehat{v}_{t,i,j}$ is simply the number of elite samples for which the i-th component is equal to j, divided by the total number of elite samples.

A possible stopping rule for combinatorial optimization problems is to stop when the overall best objective value does not change over a number of iterations. Alternatively, one could stop when the sampling distribution has "degenerated" enough. For example, when in (11) the $\{\widehat{v}_{t,i,j}\}$ differ less than some small $\varepsilon > 0$ from the $\{\widehat{v}_{t-1,i,j}\}$.

3.1 Knapsack Packing Problem

A well-known NP-complete combinatorial optimization problem is the 0-1 knapsack problem (Kellerer et al. 2004) defined as:

$$\begin{aligned} &\max_{\mathbf{x}} \sum_{j=1}^{n} p_j x_j, \quad x_j \in \{0,1\}, \;\; j = 1, \ldots, n \;, \\ &\text{subject to :} \sum_{j=1}^{n} w_{i,j} x_j \leqslant c_i, \quad i = 1, \ldots, m \;. \end{aligned} \tag{12}$$

Here $\{p_j\}$ and $\{w_{i,j}\}$ are positive weights and $\{c_i\}$ are positive cost parameters. In order to define a single objective function, we adopt the penalty function approach and define

$$S(\mathbf{x}) \stackrel{\text{def}}{=} \beta \sum_{i=1}^{m} \mathrm{I}_{\{\sum_j w_{i,j} x_j > c_i\}} + \sum_{j=1}^{n} p_j \, x_j \;,$$

where $\beta = -\sum_{j=1}^{n} p_j$. In this way, $S(\mathbf{x}) \leqslant 0$ if one of the inequality constraints is not satisfied and $S(\mathbf{x}) = \sum_{j=1}^{n} p_j \, x_j$ is all of the constraints are satisfied. As a particular example, consider the *Sento1.dat* knapsack problem given in the appendix of Senju and Toyoda (1968). The problem has 30 constraints and 60 variables. Since the solution vector $\mathbf{x}$ is binary, a simple choice for the sampling density in Step 2 of Algorithm 2.2 is the multivariate Bernoulli density $f(\mathbf{x}; \mathbf{v}) = \prod_{j=1}^{n} v_j^{x_j} (1 - v_j)^{1 - x_j}$. We apply Algorithm 2.2 to this particular problem with $N = 10^3$ and $N^{\mathrm{e}} = 20$, $\widehat{\mathbf{v}}_0 = (1/2, \ldots, 1/2)$. Note that we do not use any smoothing for $\widehat{\mathbf{v}}_t$ (that is, $\alpha = 1$ in (10)) and the solution $\widehat{\mathbf{v}}_t$ of (9) at each iteration is given by

$$\widehat{v}_{t,j} = \frac{\sum_{k=1}^{N} \mathrm{I}_{\{\widehat{S}(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} X_{k,j}}{\sum_{k=1}^{N} \mathrm{I}_{\{\widehat{S}(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}}}, \quad j = 1, \ldots, n \;, \tag{13}$$

where $X_{k,j}$ is the j-th component of the k-th random binary vector $\mathbf{X}$. In Step 4 of Algorithm 2.2 we stop the algorithm if $d_t = \max_{1 \leqslant j \leqslant n} \{\min\{\widehat{v}_{t,j}, 1 - \widehat{v}_{t,j}\}\} \leqslant 0.01$. Table 1 shows the typical evolution of the CE combinatorial optimization algorithm on the *Sento1.dat* problem. For each iteration t we recorded the threshold $\widehat{\gamma}_t$, the largest value of $S(\mathbf{X}_k)$ in the current population, and the value of the stopping criterion d_t.

The global maximum value for this problem is 6704. Figure 1 shows the evolution of the probability vector $\widehat{\mathbf{v}}_t$, which characterizes the multivariate Bernoulli distribution $f(\cdot; \widehat{\mathbf{v}}_t)$. Note that $\widehat{\mathbf{v}}_t$ converges to a binary vector corresponding to the optimal solution.

Table 1: Typical evolution of the CE method on the knapsack problem. The last column shows the stopping value $d_t = \max_{1 \leqslant j \leqslant n}\{\min\{\widehat{v}_{t,j}, 1 - \widehat{v}_{t,j}\}\}$ for each t.

t	$\widehat{\gamma}_t$	best score	d_t	t	$\widehat{\gamma}_t$	best score	d_t
1	-192450	-45420	0.495	9	4410	5920	0.46
2	-119180	1056	0.455	10	5063.5	6246	0.46
3	-48958	1877	0.395	11	5561	6639	0.452
4	-8953	3331	0.34	12	5994.5	6704	0.47
5	1186	3479	0.415	13	6465.5	6704	0.46
6	2039.5	4093	0.455	14	6626	6704	0.355
7	2836.5	4902	0.435	15	6677	6704	0.346
8	3791	5634	0.485	16	6704	6704	0

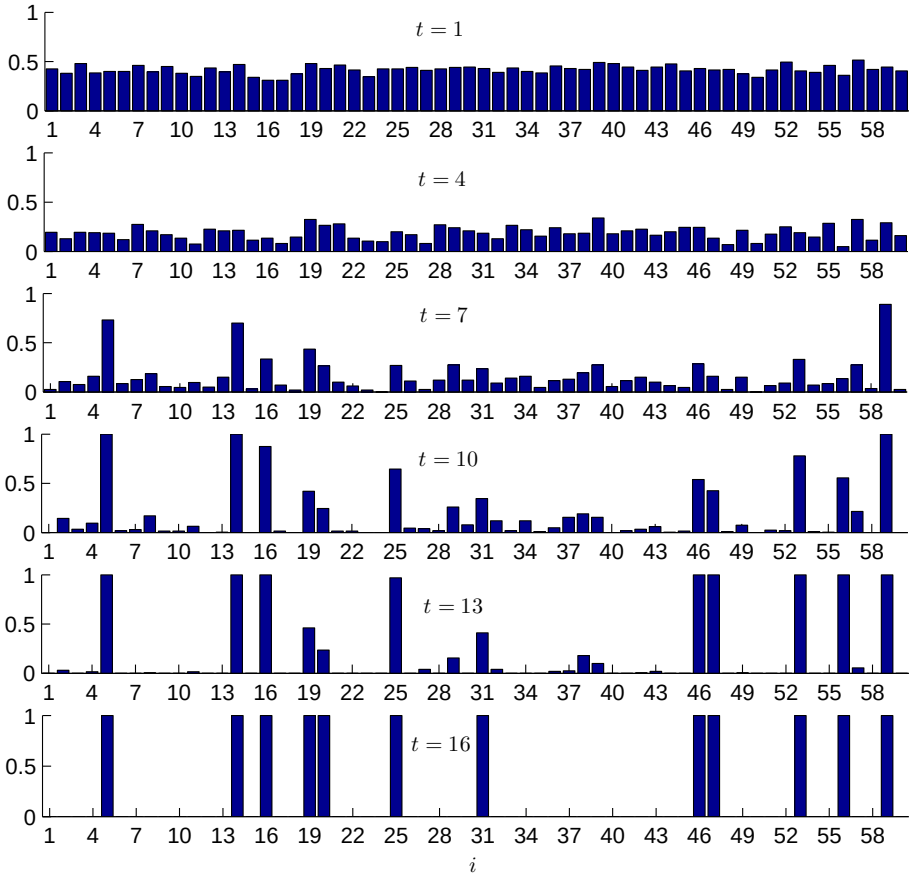

Figure 1: The evolution of the probability vector $\widehat{\mathbf{v}}_t$.

3.2 Boolean Satisfiability Problem

Let $\mathbf{x} = (x_1, \ldots, x_n)$ be a binary vector and suppose $c_1(\mathbf{x}), \ldots, c_m(\mathbf{x})$ are m functions, called *clause functions*, such that $c_i : \{0,1\}^n \to \{0,1\}$. For given clauses, the objective is to find

$$\mathbf{x}^* = \underset{\mathbf{x}\in\{0,1\}^n}{\operatorname{argmax}} \prod_{i=1}^{m} c_i(\mathbf{x}) \; . \tag{14}$$

That is, to find a vector $\mathbf{x}^*$ satisfying all clauses simultaneously. This Boolean satisfiability (SAT) problem (Gu et al. 1997) is of importance in computer science and operations research, because any NP-complete problem, such as the traveling salesman problem, can be translated into a SAT problem in polynomial time.

We now show how one can apply Algorithm 2.2 to the SAT problem. First, note that the function $\prod_{i=1}^{m} c_i(\mathbf{x})$ in (14) takes on only the values 0 or 1, and is thus not suitable for the level approach in the CE method. A more suitable and equivalent formulation is:

$$\mathbf{x}^* = \underset{\mathbf{x}\in\{0,1\}^n}{\operatorname{argmax}} S(\mathbf{x}) \stackrel{\text{def}}{=} \underset{\mathbf{x}\in\{0,1\}^n}{\operatorname{argmax}} \sum_{i=1}^{m} c_i(\mathbf{x}) \; , \tag{15}$$

where the function $S : \{0,1\}^n \to \{1, \ldots, m\}$. In this way, the function S can be used to measure the distance from our goal by indicating the number of satisfied clauses (as opposed to simply indicating whether all clauses are satisfied or not). As a specific example we consider the SAT instance `uf75-01` from Hoos and Stützle (2000).

To illustrate the role of the sampling distribution, we first use the same method of generation as in the knapsack problem; that is, at each iteration t the j-th component of $\mathbf{X}$ is generated according to a $\mathsf{Ber}(\widehat{v}_{t,j})$ distribution, independently of all other components. The updating formula is thus (13). We run the CE algorithm with a sample size of $N = 10^4$ and a rarity parameter of $\varrho = 0.1$, giving $N^{\mathrm{e}} = 10^3$ elite samples per iteration. We take $\widehat{\mathbf{v}}_0 = (0.5, \ldots, 0.5)$ and do not use smoothing ($\alpha = 1$). The upper panel of Figure 2 shows the evolution of the method over the first 30 iterations. The figure shows the scores of the best and worst performing elite samples at each iteration t, together with the value $S(\mathbf{x}^*) = 325$. The optimization gets close to the maximal level 325 (indicated by a horizontal line), but stops at 323. The reason why the CE algorithm does not reach the maximum is that the sampling distribution is too simple.

Next, instead of the basic multivariate Bernoulli model for $f(\cdot; \mathbf{v})$, we consider the Bernoulli mixture model (see Section 4.2 for a discussion of parametric mixture models)

$$f(\mathbf{x}; \mathbf{v}) = \sum_{k=1}^{K} w_k \prod_{j=1}^{n} p_{k,j}^{x_j} \, (1 - p_{k,j})^{1-x_j} \; ,$$

where K is the number of mixture components, $\mathbf{w} = (w_1, \ldots, w_K)$ are the weights ($w_k \geqslant 0, \; \sum_k w_k = 1$) associated with each component, each $\mathbf{p}_k =$

$(p_{k,1}, \ldots, p_{k,n})$ is a vector of probabilities, and $\mathbf{v} = (\mathbf{w}, \mathbf{p}_1, \ldots, \mathbf{p}_K)$ collects all the unknown parameters (we assume that K is given). The greater flexibility in the parametric model comes at a cost — the maximum likelihood program (9) in Step 3 of Algorithm 2.2 no longer has a simple closed form solution. Nevertheless, an approximate solution of (9) can be obtained using the EM algorithm of Dempster et al. (1977), where the initial starting point for the EM routine is chosen at random from the elite vectors. The lower panel of Figure 2 shows the evolution of this CE algorithm using $K = 6$ components for the Bernoulli mixture density and the same algorithmic parameters ($N = 10^4, \varrho = 0.1, \alpha = 1, \widehat{\mathbf{v}}_0 = (0.5, \ldots, 0.5)$). With this setup the algorithm quickly reaches the desired level of 325, justifying the extra computational cost of fitting a parametric mixture model via the EM algorithm.

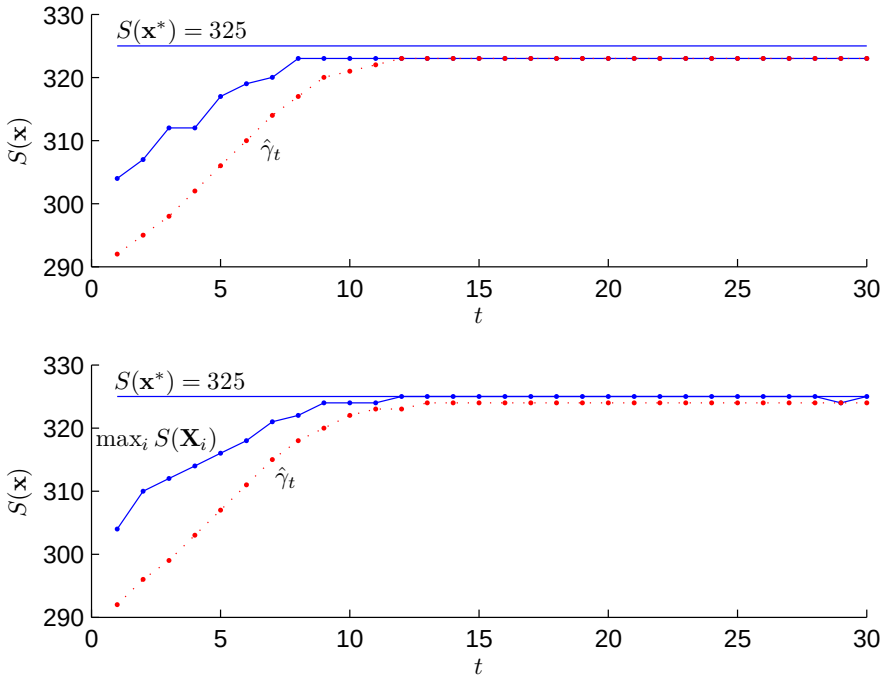

Figure 2: Evolution of the CE algorithm on the `uf75-01` SAT instance. The upper panel corresponds to the case where $f(\mathbf{x}; \mathbf{v})$ is a multivariate Bernoulli density and the lower panel corresponds to the case where $f(\mathbf{x}; \mathbf{v})$ is Bernoulli mixture model with $K = 6$ components. Both panels show the best score, $\max_k S(\mathbf{X}_k)$, and $\widehat{\gamma}_t$ for each iteration t.

3.3 Network Planning Problem

An important integer optimization problem is the *Network Planning Problem* (NPP). Here a network is represented as an undirected graph with edges (links) that may fail with a given probability. The objective of the NPP is to optimally purchase a collection of edges, subject to a fixed budget, so as to maximize the *network reliability* — defined as the probability that all nodes in a given set of

nodes are connected by functioning edges. Each edge comes with a pre-specified cost and reliability (probability that it works). The NPP has applications in engineering (Wang et al. 2009), telecommunications, transportation, energy supply systems (de Silva et al. 2010, Kothari and Kroese 2009), computer and social networking (Hintsanen et al. 2010). The difficulty in solving this problem derives from the following aspects of the optimization.

1. Typically the computation of the reliability of large networks is a difficult #P-complete computational problem, which either requires Monte Carlo simulation or sharp bounds on the reliability (Rubino 1998, Won and Karray 2010). In the Monte Carlo case the NPP becomes a noisy optimization problem, that is, the objective function values are estimated from simulation.

2. For highly reliable networks, the network reliability is a rare-event probability. In such cases Crude Monte Carlo simulation is impractical and one has to resort to sophisticated rare-event probability estimation methods (Rubino and Tuffin 2009).

3. Even if the reliability could somehow be computed easily, thus dispensing with the problems above, the NPP remains an NP-hard 0–1 knapsack optimization problem with a nonlinear objective function. In response to this, various optimization heuristics have been developed, such as simulated annealing (Cancela and Urquhart 1995) and genetic algorithms (Dengiz et al. 1997, Reichelt et al. 2007).

4. Finally, the problem is constrained and for most optimization heuristics like the simulated annealing and genetic algorithms, the penalty function approach is inefficient.

Here we apply the CE method to tackle all of the above problems simultaneously. We now explain the implementation and mathematical details, largely following Kroese et al. (2007a).

Let $\mathcal{G}(\mathscr{V}, \mathscr{E}, \mathscr{K})$ be an undirected graph, where $\mathscr{V} = \{1, \ldots, \dot{v}\}$ is the set of $\dot{v}$ nodes (vertices), $\mathscr{E} = \{1, \ldots, n\}$ is the set of n edges, and $\mathscr{K} \subseteq \mathscr{V}$ with $|\mathscr{K}| \geqslant 2$ is the set of *terminal nodes*. Sometimes we may refer to an edge by specifying the pair of nodes it connects, and write

$$\mathscr{E} = \{(\dot{v}_i, \dot{v}_j), \ldots, (\dot{v}_k, \dot{v}_l)\}, \quad \dot{v}_i, \dot{v}_j, \dot{v}_k, \dot{v}_l \in \mathscr{V} .$$

For example, in Figure 3 the edge set is given by

$$\mathscr{E} = \{(1,2), (1,3), (1,6), (2,3), (2,4), (3,4), (3,6), (4,5), (5,6)\} ,$$

and the set of terminal nodes is $\{2, 5\}$. For each $e = 1, \ldots, n$ let $B_e \sim \mathsf{Ber}(p_e)$ be a Bernoulli random variable with success probability p_e indicating whether the e-th link is operational. In other words, the event $\{B_e = 1\}$ corresponds to edge e being operational and the event $\{B_e = 0\}$ corresponds to the edge being nonoperational. It follows that each of the 2^n states of the network can be represented by a binary vector $\mathbf{B} = (B_1, \ldots, B_n)$. For example, the state of

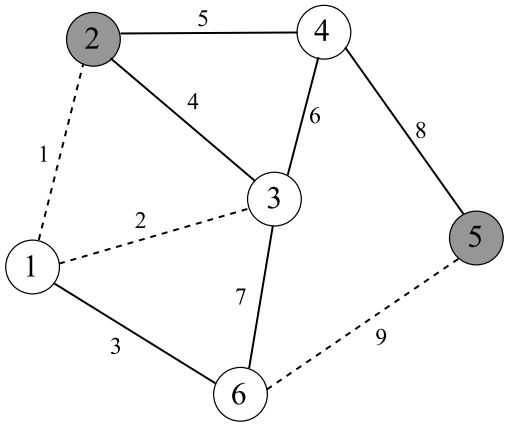

Figure 3: A reliability network with 6 nodes and 9 edges or links. The network works if the two terminal nodes (filled circles) are connected by functioning links. Failed links are indicated by dashed lines.

the network on Figure 3 can be represented as $\mathbf{B} = (0, 0, 1, 1, 1, 1, 1, 1, 0)$. Let $\mathscr{E}[\mathbf{B}]$ denote the set of operational edges corresponding to $\mathbf{B}$. For example, the set of operational edges on Figure 3 is

$$\mathscr{E}[\mathbf{B}] = \{(1, 6), (2, 3), (2, 4), (3, 4), (3, 6), (4, 5)\} \equiv \{3, 4, 5, 6, 7, 8\} .$$

The size of the set of edges $\mathscr{E}[\mathbf{B}]$ is $|\mathscr{E}[\mathbf{B}]| = B_1 + \cdots + B_n$. The *reliability* of the network is the probability that all of the nodes in $\mathscr{K}$ are connected by the set $\mathscr{E}[\mathbf{B}]$ of operational edges. For a given set of edges $\mathscr{E}$ we define the *structure function* Φ as

$$\Phi[\mathscr{E}] = \begin{cases} 1 & \text{if } \mathscr{E} \text{ contains a path connecting the nodes in } \mathscr{K} \\ 0 & \text{if } \mathscr{E} \text{ does not contain a path connecting the nodes in } \mathscr{K} . \end{cases}$$

We can write the reliability of the network as

$$r = \sum_{\boldsymbol{b}} \Phi(\mathscr{E}[\boldsymbol{b}]) \prod_{e=1}^{n} p_e^{b_e} (1 - p_e)^{1 - b_e} = \mathbb{E}\Phi(\mathscr{E}[\mathbf{B}]) ,$$

where $B_e \sim \mathsf{Ber}(p_e)$, independently for all $e = 1, \ldots, n$. Denote by $\mathbf{x} = (x_1, \ldots, x_n)$ the *purchase vector* with $(e = 1, \ldots, n)$

$$x_e = \begin{cases} 1 & \text{if link } e \text{ is purchased} \\ 0 & \text{if link } e \text{ is not purchased.} \end{cases}$$

Let us denote by $\mathbf{c} = (c_1, \ldots, c_n)$ the vector of link costs. Define

$$\mathbf{B}(\mathbf{x}) = (B_1 x_1, \ldots, B_n x_n)$$
$$S(\mathbf{x}) = \mathbb{E}\Phi(\mathscr{E}[\mathbf{B}(\mathbf{x})]) ,$$

where $S(\mathbf{x})$ is the reliability for a given purchase vector $\mathbf{x}$. Then, the NPP can be written as:

$$\begin{aligned} &\max_{\mathbf{x}} S(\mathbf{x}) \\ \text{subject to:} \quad &\sum_{e} x_e c_e \leqslant C_{\max} \ , \end{aligned} \tag{16}$$

where $C_{\max}$ is the available budget. We next address the problem of estimating $S(\mathbf{x})$ for highly reliable networks using Monte Carlo simulation.

3.3.1 Permutation Monte Carlo and Merge Process

In principle, any of the sophisticated rare-event probability estimation methods presented in Rubino and Tuffin (2009) can be used to estimate $S(\mathbf{x})$. Here we employ the merge process (MP) of Elperin et al. (1991). Briefly, Elperin et al. (1991) view the static network as a snapshot of a dynamical network, in which the edges evolve over time — starting failed and eventually becoming operational (or, alternatively, starting operational and eventually failing). They exploit the fact that one can compute exactly the reliability of the network, given the order in which the links of the network become operational, and suggests a conditional Monte Carlo method as a variance reduction tool: generate random permutations (indicating the order in which the links become operational) and then evaluate the unreliability for each permutation.

Suppose $\mathscr{E}_{\mathbf{x}} = \mathscr{E}[\mathbf{x}] = \{e_1, \ldots, e_m\} \subseteq \mathscr{E}$ is the set of purchased links. The corresponding link reliabilities are $p_{e_1}, \ldots, p_{e_m}$. Define $\lambda_e = -\ln(1 - p_e)$.

Algorithm 3.1 (Merge Process for Graph $\mathcal{G}(\mathscr{V}, \mathscr{E}_{\mathbf{x}}, \mathscr{K})$)

1. *Generate $E_i \sim \mathsf{Exp}(\lambda_{e_i})$ independently for all purchased links $e_i \in \mathscr{E}_{\mathbf{x}}$. Sort the sequence $E_1, \ldots, E_m$ to obtain the permutation $(\pi(1), \ldots, \pi(m))$ such that*

$$E_{\pi(1)} < E_{\pi(2)} < \cdots < E_{\pi(m)} \ .$$

Let $\mathscr{D}$ be a dynamic ordered list of nonoperational links. Initially, set

$$\mathscr{D} = (e_{\pi(1)}, e_{\pi(2)}, \ldots, e_{\pi(m)}) \ .$$

Compute

$$\lambda_1^* = \sum_{e \in \mathscr{D}} \lambda_e$$

and set the counter $b = 1$.

2. *Let e^* be the first link in the list $\mathscr{D}$. Remove link e^* from $\mathscr{D}$ and add it to the set $\mathscr{U}$ (which need not be ordered). Let $\mathcal{G}_b \equiv \mathcal{G}(\mathscr{V}, \mathscr{U}, \mathscr{K})$ be the network in which all edges in $\mathscr{U}$ are working and the rest of the edges (all edges in $\mathscr{D}$) are failed.*

3. *If the network $\mathcal{G}_b$ is operational go to Step 6; otherwise, continue to Step 4.*

4. *Identify the* connected component *of which link e^* is a part. Find all links in $\mathscr{D}$ in this connected component. These are called* redundant links, *because they connect nodes that are already connected. Denote the set of redundant links by $\mathscr{R}$.*

5. *Remove the links in $\mathscr{R}$ from $\mathscr{D}$. Compute*

$$\lambda_{b+1}^* = \lambda_b^* - \lambda_{e^*} - \sum_{e \in \mathscr{R}} \lambda_e = \sum_{e \in \mathscr{D}} \lambda_e \; .$$

   Increment $b = b + 1$ and repeat from Step 2.

6. *Let A be the $b \times b$ square matrix:*

$$A = \begin{pmatrix} -\lambda_1^* & \lambda_1^* & 0 & \dots & 0 \\ 0 & -\lambda_2^* & \lambda_2^* & \dots & 0 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & -\lambda_{b-1}^* & \lambda_{b-1}^* \\ 0 & \dots & 0 & 0 & -\lambda_b^* \end{pmatrix} ,$$

   and let $A^* = (A_{i,j}^*) = \mathrm{e}^A = \sum_{k=0}^{\infty} \frac{A^k}{k!}$ be the matrix exponential of A. Output the unbiased estimator of the unreliability of the network

$$Z = \sum_{j=1}^{b} A_{1,j}^* \; .$$

For the NPP we use the following unbiased estimator of the reliability $S(\mathbf{x})$:

$$\widehat{S}(\mathbf{x}) = 1 - \frac{1}{M} \sum_{k=1}^{M} Z_k, \tag{17}$$

where $Z_1, \dots, Z_M$ are independent realizations from Algorithm 3.1. Note that if $\mathscr{E}_{\mathbf{x}}$ does not contain enough operational links to connect the nodes in $\mathscr{K}$, then the network reliability is trivially equal to 0.

3.3.2 Sampling with a Budget Constraint

While it is possible to incorporate the budget constraint via a penalty function added to (17), here we take a more direct approach, in which we generate a sample of n dependent Bernoulli random variables such that the constraint $\sum_e c_e X_e \leqslant C_{\max}$ is satisfied. In other words, in Step 2 of Algorithm 2.2 we sample random purchase vectors $\mathbf{X}_1, \dots, \mathbf{X}_N$ from a parametric density $f(\cdot; \mathbf{v})$, $\mathbf{v} = (v_1, \dots, v_n)$ that takes into account the budget constraint in (16) and is implicitly defined via the following algorithm (Kroese et al. 2007a).

Algorithm 3.2 (Purchase vector generation) *Given the vector* $\mathbf{v}$*, set* $i = 1$ *and execute the following steps.*

1. *Generate* $U_1, \ldots, U_n \overset{iid}{\sim} \mathsf{U}(0,1)$ *and let* $\boldsymbol{\pi}$ *be the permutation which satisfies* $U_{\pi(1)} < U_{\pi(2)} < \cdots < U_{\pi(n)}$.

2. *If* $c_{\pi(i)} + \sum_{e=1}^{i-1} c_{\pi(e)} X_{\pi(e)} \leqslant C_{\max}$*, generate* $U \sim \mathsf{U}(0,1)$ *and set* $X_{\pi(i)} = \mathrm{I}_{\{U < v_{\pi(i)}\}}$*; otherwise, set* $X_{\pi(i)} = 0$.

3. *If* $i < n$*, increment* $i = i + 1$ *and repeat from Step 2; otherwise, deliver the random purchase vector* $\mathbf{X} = (X_1, \ldots, X_n)$.

We can interpret $\mathbf{v}$ as a vector of purchase probabilities. The aim of the CE optimization Algorithm 2.2 is then to construct a sequence of probabilities $\{\widehat{\mathbf{v}}_t, t = 0, 1, 2, 3\}$ that converges to a degenerate (binary) vector $\mathbf{x}^*$ corresponding to the optimal purchase vector. The updating of $\widehat{\mathbf{v}}_t$ at each iteration is given by (9) with S replaced by the estimated $\widehat{S}$. The solution of (9) is:

$$\widetilde{v}_{t,j} = \frac{\sum_{k=1}^{N} \mathrm{I}_{\{\widehat{S}(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}} X_{k,j}}{\sum_{k=1}^{N} \mathrm{I}_{\{\widehat{S}(\mathbf{X}_k) \geqslant \widehat{\gamma}_t\}}}, \quad j = 1, \ldots, n\,. \tag{18}$$

For clarity we now restate the CE optimization algorithm as applied to the NPP.

Algorithm 3.3 (CE Optimization for NPP)

1. *Let* $\widehat{\mathbf{v}}_0 = (1/2, \ldots, 1/2)$*. Let* $N^{\mathrm{e}} = \lceil \varrho N \rceil$*, where* ϱ *is the user-specified rarity parameter. Set* $t = 1$ *(level counter).*

2. *Generate independent purchase vectors* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *using Algorithm 3.2 with* $\mathbf{v} = \widehat{\mathbf{v}}_t$*. For each purchase vector estimate the reliability* $\widehat{S}(\mathbf{X}_i)$ *using the merge process estimator* (17) *and rank the reliabilities from smallest to largest:* $S_{(1)} \leqslant \ldots \leqslant S_{(N)}$*. Let* $\widehat{\gamma}_t$ *be the sample* $(1 - \varrho)$*-quantile of reliabilities; that is,* $\widehat{\gamma}_t = S_{(N - N^{\mathrm{e}} + 1)}$.

3. *Use the* **same** *sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *to estimate the purchase probabilities via* (18).

4. *Smooth the purchase probabilities in Step 3 to obtain* $\widehat{\mathbf{v}}_t$ *with*

$$\widehat{v}_{t,j} = \alpha\, \widetilde{v}_{t,j} + (1 - \alpha)\, \widehat{v}_{t-1,j}, \quad j = 1, \ldots, n\,.$$

5. *If*

$$\max_{1 \leqslant j \leqslant n} \{\min\{\widehat{v}_{t,j}, 1 - \widehat{v}_{t,j}\}\} \leqslant \varepsilon$$

for some small ε*, say* $\varepsilon = 10^{-2}$*, stop; otherwise, set* $t = t + 1$ *and repeat from Step 2.*

We now illustrate the method on the complete graph K_{10} given in Figure 4.

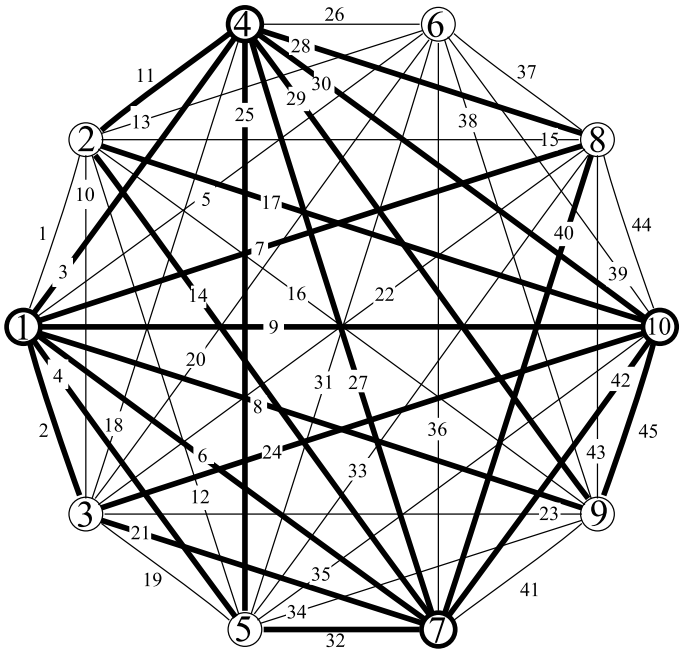

Figure 4: A complete graph with 10 nodes and 45 edges.

The costs and reliabilities for the graph were generated in the (quite arbitrary) pseudo-random fashion as follows.

Algorithm 3.4 (Generation of Costs and Reliabilities for a K_{10} graph)
Set $i = 0$ and $j = 1$
while $j \leqslant 45$ **do**
$i \leftarrow i + 1$
$b = 1 - (7654321 \mod (987 + i))/10^5$
if $b > 0.99$ and $b \neq p_k$ for all k **then**
$p_j = b$
$c_j = 20/\exp(8\sqrt{1-b})$
$j \leftarrow j + 1$
end if
end while

We let $\mathscr{K} = \{1, 4, 7, 10\}$ and $C_{\max} = 250$. We apply Algorithm 3.3 with $N = 100$, $N^{\mathrm{e}} = 10$ ($\varrho = 0.1$), $\alpha = 1/2$, and $\varepsilon = 10^{-2}$. For the merge process estimator (17) we used $M = 100$. The estimated overall purchase vector corresponds to the following edges:

$$\mathscr{E}[\mathbf{x}^*] = \{2, 3, 4, 6, 7, 8, 9, 11, 14, 17, 21, 24, 25, 27, 28, 29, 30, 32, 40, 42, 45\}$$

with a total cost of 241.2007. These edges are drawn thicker on Figure 4. Using $M = 10^5$ samples in (17) we obtained the estimate 4.80×10^{-16} (with an estimated relative error of 1.7%) of the corresponding unreliability of the purchased network.

4 Continuous Optimization

When the state space is continuous, in particular when $\mathscr{X} = \mathbb{R}^n$, the optimization problem is often referred to as a *continuous optimization* problem. The CE sampling distribution on $\mathbb{R}^n$ can be quite arbitrary and does not need to be related to the function that is being optimized. The generation of a random vector $\mathbf{X} = (X_1, \ldots, X_n) \in \mathbb{R}^n$ in Step 2 of Algorithm 2.2 is most easily performed by drawing the n coordinates independently from some 2-parameter distribution. In most applications a normal (Gaussian) distribution is employed for each component. Thus, the sampling density $f(\cdot; \mathbf{v})$ of $\mathbf{X}$ is characterized by a vector of means $\boldsymbol{\mu}$ and a vector of variances $\boldsymbol{\sigma}^2$ (and we may write $\mathbf{v} = (\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$). The choice of the normal distribution is motivated by the availability of fast normal random number generators on modern statistical software and the fact that the maximum likelihood maximization (or cross-entropy minimization) in (9) yields a very simple solution — at each iteration of the CE algorithm the parameter vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the vectors of sample means and sample variance of the elements of the set of N^{e} best performing vectors (that is, the elite set); see, for example, Kroese et al. (2006). In summary, the CE method for continuous optimization with a Gaussian sampling density is as follows.

Algorithm 4.1 (CE for Continuous Optimization: Normal Updating)

1. ***Initialize:*** *Choose* $\widehat{\boldsymbol{\mu}}_0$ *and* $\widehat{\boldsymbol{\sigma}}_0^2$. *Set* $t = 1$.

2. ***Draw:*** *Generate a random sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *from the* $\mathsf{N}(\widehat{\boldsymbol{\mu}}_{t-1}, \widehat{\boldsymbol{\sigma}}_{t-1}^2)$ *distribution.*

3. ***Select:*** *Let* $\mathcal{I}$ *be the indices of the* N^{e} *best performing (=elite) samples.* ***Update:*** *For all* $j = 1, \ldots, n$ *let*

$$\widetilde{\mu}_{t,j} = \sum_{k \in \mathcal{I}} X_{k,j} / N^{\mathrm{e}} \tag{19}$$

$$\widetilde{\sigma}_{t,j}^2 = \sum_{k \in \mathcal{I}} (X_{k,j} - \widetilde{\mu}_{t,j})^2 / N^{\mathrm{e}}. \tag{20}$$

4. ***Smooth:***

$$\widehat{\boldsymbol{\mu}}_t = \alpha \widetilde{\boldsymbol{\mu}}_t + (1-\alpha) \widehat{\boldsymbol{\mu}}_{t-1}, \quad \widehat{\boldsymbol{\sigma}}_t = \alpha \widetilde{\boldsymbol{\sigma}}_t + (1-\alpha) \widehat{\boldsymbol{\sigma}}_{t-1} \,. \tag{21}$$

5. *If* $\max_j \{\widehat{\sigma}_{t,j}\} < \varepsilon$ **stop** *and return* $\boldsymbol{\mu}_t$ *(or the overall best solution generated by the algorithm) as the approximate solution to the optimization. Otherwise, increase* t *by* 1 *and return to Step 2.*

For *constrained* continuous optimization problems, where the samples are restricted to a subset $\mathscr{X} \subset \mathbb{R}^n$, it is often possible to replace the normal sampling with sampling from a truncated normal distribution while retaining the updating formulas (19)–(20). An alternative is to use a beta distribution.

Smoothing, as in Step 4, is often crucial to prevent premature shrinking of the sampling distribution. Another approach is to *inject* extra variance into the sampling distribution, for example by increasing the components of $\boldsymbol{\sigma}^2$, once the distribution has degenerated; see the examples below and Botev and Kroese (2004).

4.1 Optimal Control

Optimal control problems have been studied in many areas of science, engineering, and finance. Yet the numerical solution of such problems remains challenging. A number of different techniques have been used, including nonlinear and dynamic programming (Bertsekas 2007), ant colony optimization (Borzabadi and Mehne 2009), and genetic algorithms (Wuerl et al. 2003). For a comparison among these methods see Borzabadi and Heidari (2010). In this section we show how to apply the CE Algorithm 4.1 to optimal control problems over a fixed interval of time. As a particular example we consider the following nonlinear optimal control problem:

$$
\begin{aligned}
& \min_{u \in \mathscr{U}} J\{u\} = \min_{u \in \mathscr{U}} \int_0^1 \left(2\, z_1(\tau) - \frac{u(\tau)}{2} \right) \mathrm{d}\tau \\
\text{subject to:} \quad & \frac{\mathrm{d}z_1}{\mathrm{d}\tau} = u - z_1 + z_2\, \mathrm{e}^{-z_2^2/10} \\
& \frac{\mathrm{d}z_2}{\mathrm{d}\tau} = u - \pi^{5/2}\, z_1 \cos\left(\frac{\pi}{2}\, u \right) \\
& z_1(0) = 0, \quad z_2(0) = 1, \quad |u(\tau)| \leqslant 1, \quad \tau \in [0,1]
\end{aligned}
$$

where the function u belongs to the set $\mathscr{U}$ of all piecewise continuous functions on $[0,1]$. To obtain an approximate numerical solution we translate the infinite-dimensional *functional* optimization problem into a finite-dimensional *parametric* optimization problem. A simple way to achieve this is to divide the interval $[0,1]$ into $n-1$ subintervals $[\tau_1, \tau_2], \ldots, [\tau_{n-1}, \tau_n]$ and approximate u via a spline that interpolates the points $\{(\tau_i, x_i)\}$, where $\mathbf{x} = (x_1, \ldots, x_n)$ is a vector of *control points*. For a given vector $\mathbf{x}$ of control points we can write the approximation as the linear combination:

$$
u(\cdot) \approx u_{\mathbf{x}}(\cdot) = \sum_{k=1}^{n} x_k \phi_k(\cdot) \,,
$$

where $\{\phi_k \in \mathscr{U}\}$ are interpolating polynomials. For example, one of the simplest choices gives the *nearest neighbor* interpolation ($\tau_{n+1} = \tau_n$ and $\tau_0 = \tau_1$):

$$
\phi_k(\tau) = \mathrm{I}\left\{ \frac{\tau_k + \tau_{k-1}}{2} < \tau < \frac{\tau_{k+1} + \tau_k}{2} \right\} \,. \tag{22}
$$

The objective now is to find the optimal $u_{\mathbf{x}}(\cdot)$ that solves the finite-dimensional control problem:

$$
\begin{aligned}
& \min_{\mathbf{x}} S(\mathbf{x}) = \min_{\mathbf{x}} \int_0^1 \left(2\, z_1(\tau) - \frac{u_{\mathbf{x}}(\tau)}{2} \right) \mathrm{d}\tau \\
\text{subject to:} \quad & \frac{\mathrm{d}z_1}{\mathrm{d}\tau} = u_{\mathbf{x}} - z_1 + z_2\, \mathrm{e}^{-z_2^2/10} \\
& \frac{\mathrm{d}z_2}{\mathrm{d}\tau} = u_{\mathbf{x}} - \pi^{5/2}\, z_1 \cos\left(\frac{\pi}{2}\, u_{\mathbf{x}} \right) \\
& z_1(0) = 0, \quad z_2(0) = 1, \quad |u_{\mathbf{x}}(\tau)| \leqslant 1, \quad \tau \in [0,1].
\end{aligned}
\tag{23}
$$

To find the optimal vector of control points $\mathbf{x} = (x_1, \ldots, x_n)$ we apply Algorithm 4.1 in the following way.

1. In Step 2 we sample each component X_i of $\mathbf{X}$ from a $\mathsf{N}(\widehat{\mu}_{t,j}, \widehat{\sigma}^2_{t,j})$ distribution, which is truncated to the interval $[-1, 1]$. This ensures that the approximation $u_{\mathbf{x}}(\cdot)$ is constrained in the interval $[-1, 1]$.

2. We divide the time interval $[0, 1]$ into $n - 1$ equal subintervals and use (22) for the approximating curve. This gives an optimization problem of n dimensions.

3. For a given $u_{\mathbf{x}}$ we solve the nonlinear ODE system in (23) using the Runge–Kutta formulae of Dormand and Prince (1980).

4. The sample size is $N = 100$, the stopping parameter $\varepsilon = 0.01$, and $\widehat{\boldsymbol{\mu}}_0 = \mathbf{0}$ and $\widehat{\boldsymbol{\sigma}}_0 = (10, \ldots, 10)$, and $N^{\mathrm{e}} = 20$. Since this is a minimization problem and $N^{\mathrm{e}} = 20$, the elite samples are the first 20 control vectors that give the lowest score $S(\mathbf{x})$. We applied smoothing only to the parameter $\boldsymbol{\sigma}$ with $\alpha = 0.5$.

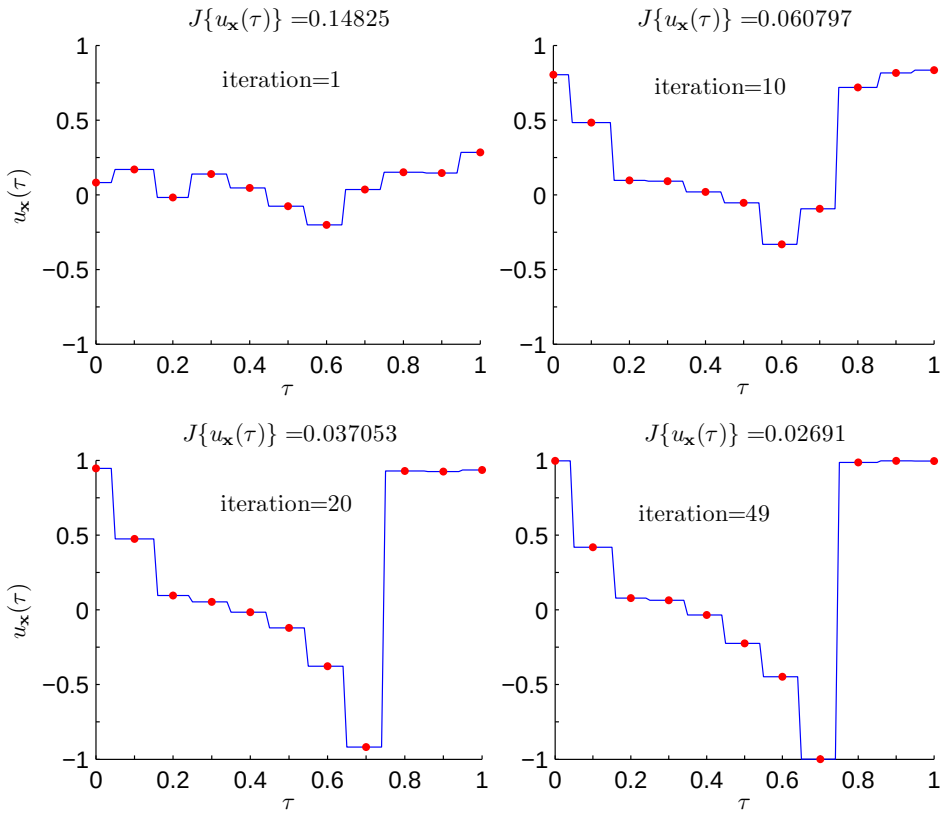

Figure 5: The evolution of the CE optimization method on the nonlinear optimal control problem with $n = 11$ control points. The graphs show the control $u_{\boldsymbol{\mu}_t}(\cdot)$ for $t = 1, 10, 20$, and the final $t = 49$. The control points $\boldsymbol{\mu}_t$ are shown as dots on the curve $u_{\boldsymbol{\mu}_t}(\cdot)$.

Figure 5 shows the typical evolution of the CE optimization Algorithm 4.1 with $n = 11$ control points. The figure shows the approximation $u_{\mathbf{x}}(\cdot)$ and the control points $\mathbf{x} = \boldsymbol{\mu}_t$ (where $\boldsymbol{\mu}_t$ is the mean vector given in (19)) for $t = 1, 10, 20$, and the final iteration $t = 49$. For each graph we have recorded the corresponding value of J. In particular, for this simulation the smallest value found is $J\{u_{\mathbf{x}}(\cdot)\} = 0.02691$. Observe the qualitative behavior of the optimal control: u starts near 1 and gradually decreases to $u = -1$ until at about $\tau = 0.7$ the control abruptly switches to $u = 1$ and remains so till the final time $\tau = 1$.

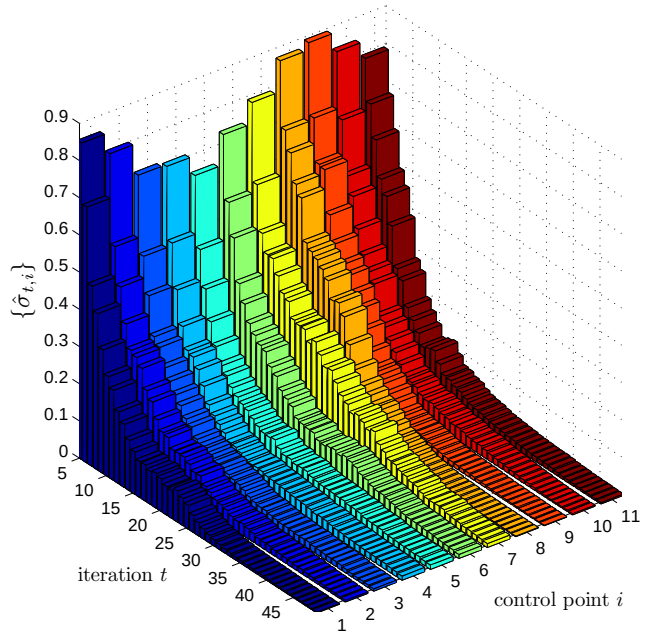

Figure 6: Evolution of the vector $\widehat{\boldsymbol{\sigma}}_t$ for iterations $t = 5, \ldots, 49$.

Figure 6 shows the evolution of the standard deviation vector $\widehat{\boldsymbol{\sigma}}_t$ associated with the sampling distribution $\mathsf{N}(\widehat{\boldsymbol{\mu}}_t, \widehat{\boldsymbol{\sigma}}_t^2)$ of each random control vector $\mathbf{X}$. Notice that all of the components of $\widehat{\boldsymbol{\sigma}}_t$ shrink to zero as the iterations progress, and that since $\widehat{\sigma}_{t,7}$ and $\widehat{\sigma}_{t,8}$ are the last to decay, the control points $\mathbf{x}_7$ and $\mathbf{x}_8$ are the last to converge.

It is desirable to investigate the dependence of the solution on the number of control points. Note that while increasing the number of control points $\mathbf{x}$ improves the approximation of u, it also makes the optimization problem more difficult, because of the increased dimensions of the optimization problem. Figure 7 shows the evolution of the CE optimization Algorithm 4.1 with $n = 21$ control points, where the interval $[0, 1]$ is again subdivided into equal segments. The algorithmic setup and the values of all parameters are kept the same. In this case the optimization effort was larger and the algorithm terminated in 88 iterations. Observe that as a result of using more control points we were able to achieve a lower minimum for J, namely, $J\{u_{\mathbf{x}}(\cdot)\} = 0.018395$.

We were not able to obtain a lower value for J using more than 21 points due to the increased complexity of the optimization. In addition, smooth cubic splines did not yield better solutions than the simple nearest neighbor interpolation due to the discontinuous nature of the optimal control.

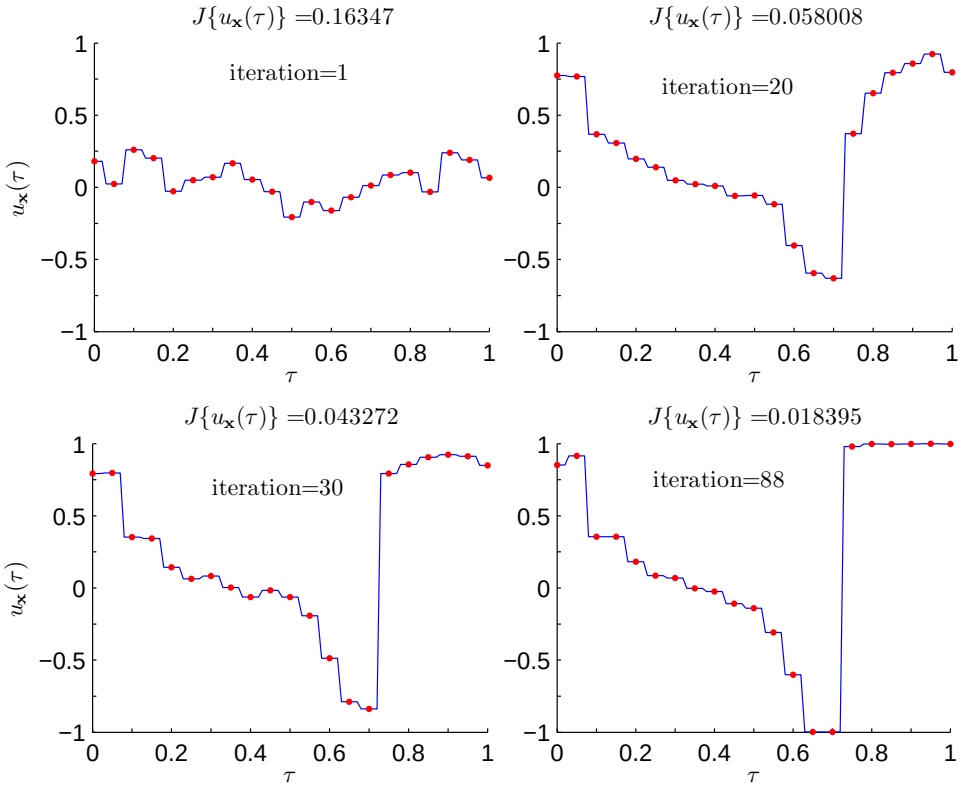

Figure 7: The evolution of the CE optimization method on the nonlinear optimal control problem with $n = 21$ control points. The optimization terminated in 88 iterations and achieved the minimum of $J\{u_{\mathbf{x}}(\cdot)\} = 0.018395$.

For more examples of using the CE method for optimal control problems see Sani and Kroese (2008), Sani (2010). In addition to a number of simple examples, Sani and Kroese (2008) apply the CE method to the optimization of a large scale model for the spread of HIV/AIDS.

4.2 Maximum Likelihood Optimization

Mixture models are widely used for clustering and pattern recognition (McLachlan and Peel 2000). A finite mixture pdf is a density of the form

$$f(\mathbf{y}) = \sum_{k=1}^{c} w_k \, f_k(\mathbf{y}),$$

where $\{w_k\}$ are probabilities summing to 1, $\{f_k\}$ are pdfs, and c is the number of mixture components. The pdfs $\{f_k\}$ may or may not belong to a single parametric family. The fitting of mixture models is best carried out using the principle of maximum likelihood and requires the maximization of a likelihood

function with many local maxima and saddle points. Typically, the EM algorithm of Dempster et al. (1977) is the most efficient method for fitting of mixture models when $\{f_k\}$ belong to the exponential family of parametric pdfs. However, the EM algorithm is either inapplicable or inefficient when the densities $\{f_k\}$ are not part of an exponential family or are not continuous with respect to some of its model parameters. For example, consider a mixture consisting of one d-dimensional Gaussian and $c-1$ uniform densities on d-dimensional rectangles:

$$f(\mathbf{y};\boldsymbol{\theta}) = \frac{w_1 \, \mathrm{e}^{-\frac{1}{2}\sum_{i=1}^{d}(y_i-\nu_{1,i})^2/\varsigma_{1,i}^2}}{(2\pi)^{\frac{d}{2}}\prod_i^d \varsigma_{1,i}} + \sum_{k=2}^{c} w_k \prod_{i=1}^{d} \frac{\mathrm{I}\left\{|y_i-\nu_{k,i}| < \frac{1}{2}\varsigma_{k,i}\right\}}{\varsigma_{k,i}} \,. \quad (24)$$

Here, $(\nu_{k,1},\ldots,\nu_{k,d})$ and $(\varsigma_{k,1},\ldots,\varsigma_{k,d})$ are the location and scale parameters of the k-th component, and

$$\boldsymbol{\theta} = ((\nu_{1,1},\ldots,\nu_{c,d}),(\varsigma_{1,1},\ldots,\varsigma_{c,d}),(w_1,\ldots,w_c))$$

summarizes all model parameters. The log-likelihood function given the data $\mathbf{y}_1,\ldots,\mathbf{y}_n$ is $S(\boldsymbol{\theta}) = \sum_{j=1}^{n} \ln f(\mathbf{y}_j;\boldsymbol{\theta})$, which is not continuous with respect to all parameters, because the support of each uniform component depends on the scale parameters $\{\varsigma_{k,i},\, k \geqslant 2\}$. We now show how to apply the CE method to solve the optimization program $\widehat{\boldsymbol{\theta}} = \mathrm{argmax}_{\boldsymbol{\theta}\in\Theta} S(\boldsymbol{\theta})$, where Θ is the set for which all $\varsigma_{k,i} > \varsigma_{\mathrm{low}} > 0$ for some threshold ς_{low} and $\{w_k\}$ are probabilities summing to one. Note that ς_{low} must be strictly positive to ensure that $\max_{\boldsymbol{\theta}\in\Theta} S(\boldsymbol{\theta}) < \infty$.

In Step 2 of Algorithm 4.1 the population $\mathbf{X}_1,\ldots,\mathbf{X}_N$ corresponds to randomly generated proposals for the parameter $\boldsymbol{\theta} = \mathbf{X}$. Thus, the CE parameter vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are of length $(2d+1)c$, with the first $d\,c$ elements associated with the location parameters of the mixture, the next $d\,c$ elements associated with the scale parameters, and the last c elements associated with the mixture weights. In particular, we sample the locations $\nu_{1,1},\ldots,\nu_{c,d}$ from the normal distributions $\mathsf{N}(\mu_1,\sigma_1^2),\ldots,\mathsf{N}(\mu_{c\,d},\sigma_{c\,d}^2)$. To account for the constraints on the scale parameters of the mixture we sample $\varsigma_{1,1},\ldots,\varsigma_{c,d}$ from the normal distributions $\mathsf{N}(\mu_{c\,d+1},\sigma_{c\,d+1}^2),\ldots,\mathsf{N}(\mu_{2c\,d},\sigma_{2c\,d}^2)$ truncated to the set $[\varsigma_{\mathrm{low}},\infty)$. To generate the mixture weights $\{w_k\}$ we need to sample positive random variables $\omega_1,\ldots,\omega_c$ such that $\sum_k \omega_k = 1$. For this purpose we generate all weights from truncated normal distributions in a random order (specified by a random permutation), as in the following algorithm.

Algorithm 4.2 (Generation of Mixture Weights)

Require: The last c elements of the CE parameter vectors $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$.

Generate a random permutation $\boldsymbol{\pi}$ of $1,\ldots,c$.

for $k = 1,\ldots,(c-1)$ **do**

Generate $w_{\boldsymbol{\pi}(k)}$ from the normal distribution $\mathsf{N}(\mu_{2dc+\boldsymbol{\pi}(k)},\sigma_{2dc+\boldsymbol{\pi}(k)}^2)$ truncated to the interval $[0,\ 1-(w_{\boldsymbol{\pi}(1)}+\cdots+w_{\boldsymbol{\pi}(k-1)})]$.

end for

Set $w_{\boldsymbol{\pi}(c)} = 1-(w_{\boldsymbol{\pi}(1)}+\cdots+w_{\boldsymbol{\pi}(c-1)})$.

return Random mixture weights $w_1,\ldots,w_c$.

As a particular example, consider fitting the mixture model (24) to the $n = 200$ random points depicted on Figure 8. The data are pseudo-random variables generated from (24) with $c = 3$, weights $(w_1, w_2, w_3) = (1/2, 1/4, 1/4)$, location parameters

$$\begin{pmatrix} \nu_{1,1} \\ \nu_{1,2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \nu_{2,1} \\ \nu_{2,2} \end{pmatrix} = \begin{pmatrix} -2 \\ -2 \end{pmatrix}, \begin{pmatrix} \nu_{3,1} \\ \nu_{3,2} \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \end{pmatrix},$$

and scale parameters

$$\begin{pmatrix} \varsigma_{1,1} \\ \varsigma_{1,2} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \begin{pmatrix} \varsigma_{2,1} \\ \varsigma_{2,2} \end{pmatrix} = \begin{pmatrix} \varsigma_{3,1} \\ \varsigma_{3,2} \end{pmatrix} = \begin{pmatrix} 4 \\ 4 \end{pmatrix}.$$

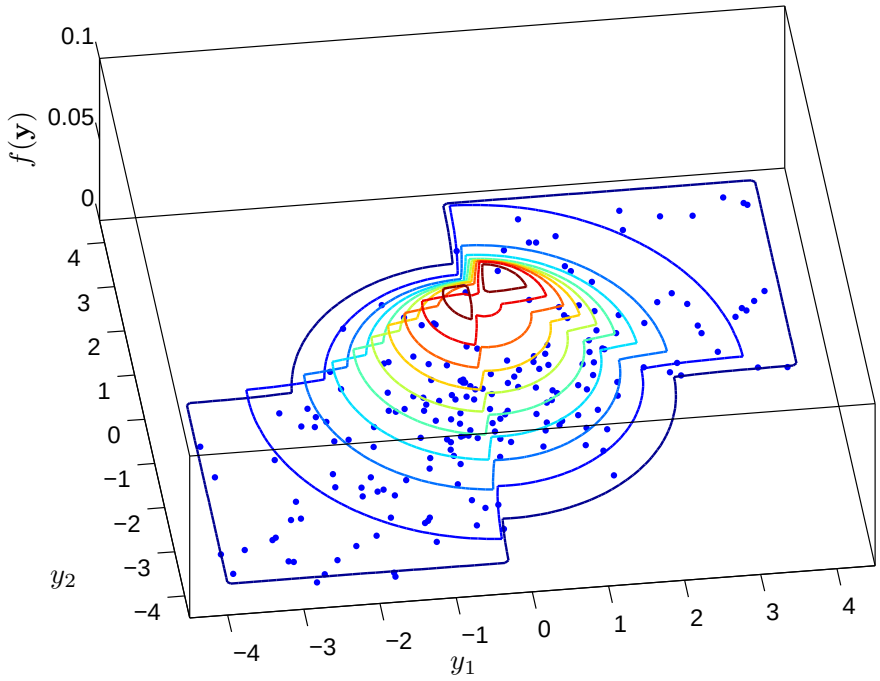

Figure 8: A sample of size 200 from a mixture of two uniform and a Gaussian distributions. Superimposed is the contour plot of the fitted mixture density $f(\mathbf{y}; \widehat{\boldsymbol{\theta}})$.

We applied Algorithm 4.1 with $N = 100$, $N^{\mathrm{e}} = 10$, $\alpha = 0.5$, $\varepsilon = 0.01$, and $\widehat{\boldsymbol{\mu}}_0 = (1, \ldots, 1)$ and $\widehat{\boldsymbol{\sigma}}_0 = (10, \ldots, 10)$. The algorithm terminated after $t = 89$ iterations with the overall maximum log-likelihood of $S(\widehat{\boldsymbol{\theta}}) = -707.7593$. Figure 8 shows the estimated mixture model. The model appears to fit the data well. In addition, the true value of the parameters $\boldsymbol{\theta}$ gives $S(\boldsymbol{\theta}) = -710.7937 < S(\widehat{\boldsymbol{\theta}})$, which suggests that $\widehat{\boldsymbol{\theta}}$ is the global maximizer. We found that for fitting mixture models the CE algorithm is not sensitive to the initial conditions and works satisfactorily for $0.2 \leq \alpha \leq 0.8$.

5 Summary

We have described the CE method for combinatorial and continuous optimization. In summary, any CE algorithm for optimization involves the following two main iterative phases:

1. **Generate** a random sample of objects in the search space $\mathscr{X}$ (trajectories, vectors, etc.) according to a specified probability distribution.
2. **Update** or refit the parameters of that distribution based on the N^{e} best performing samples (the so-called elite samples) using CE minimization.

An important part of the algorithm involves the selection of a suitable probability density that allows for simple random variable generation and simple updating formulae (arising from the CE minimization) for the parameters of the density. For most optimization problems standard parametric models such as the multivariate Bernoulli and Gaussian densities (for discrete and continuous problems, respectively) are adequate. Finally, the CE method is a robust tool for noisy continuous or discrete optimization problems.

References

G. Alon, D. P. Kroese, T. Raviv, and R. Y. Rubinstein. Application of the cross-entropy method to the buffer allocation problem in a simulation-based environment. *Annals of Operations Research*, 134(1):137–151, 2005.

D. P. Bertsekas. *Dynamic Programming and Optimal Control*, volume I. Athena Scientific, third edition, 2007.

A. H. Borzabadi and M. Heidari. Comparison of some evolutionary algorithms for approximate solutions of optimal control problems. *Australian Journal of Basic and Applied Sciences*, 4(8):3366–3382, 2010.

A. H. Borzabadi and H. H. Mehne. Ant colony optimization for optimal control problems. *Journal of Information and Computing Science*, 4(4):259–264, 2009.

Z. I. Botev and D. P. Kroese. Global likelihood optimization via the cross-entropy method with an application to mixture models. In *Proceedings of the 36th Winter Simulation Conference*, pages 529–535, Washington, D.C., 2004.

A. Boubezoula, S. Paris, and M. Ouladsinea. Application of the cross entropy method to the GLVQ algorithm. *Pattern Recognition*, 41(10):3173–3178, 2008.

L. Busoniu, R. Babuska, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming using Function Approximators*. Taylor & Francis Group, New York, 2010.

H. Cancela and M. E. Urquhart. Simulated annealing for communication network reliability improvements. In *Proceedings of the XXI Latin American Conference On Informatics*, pages 1413–1424, 1995.

K. Chepuri and T. Homem-de-Mello. Solving the vehicle routing problem with stochastic demands using the cross entropy method. *Annals of Operations Research*, 134(1):153–181, 2005.

I. Cohen, B. Golany, and A. Shtub. Resource allocation in stochastic, finite-capacity, multi-project systems through the cross entropy methodology. *Journal of Scheduling*, 10(1):181–193, 2007.

A. Costa, J. Owen, and D. P. Kroese. Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters*, 35 (5):573–580, 2007.

P. T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134(1):19–67, 2005.

A. M. Leite de Silva, R. A. G. Fernandez, and C. Singh. Generating capacity reliability evaluation based on Monte Carlo simulation and cross-entropy methods. *IEEE Transactions on Power Systems*, 25(1):129–137, 2010.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

B. Dengiz, F. Altiparmak, and A. E. Smith. Local search genetic algorithm for optimal design of reliable networks. *IEEE Transactions on Evolutionary Computation*, 1(3):179–188, 1997.

J. R. Dormand and P. J. Prince. A family of embedded Runge-Kutta formulae. *J. Comp. Appl. Math.*, 6:19–26, 1980.

T. Elperin, I. B. Gertsbakh, and M. Lomonosov. Estimation of network reliability using graph evolution models. *IEEE Transactions on Reliability*, 40 (5):572–581, 1991.

D. Ernst, M. Glavic, G.-B. Stan, S. Mannor, L. Wehenkel, and Gif sur Yvette Supelec. The cross-entropy method for power system combinatorial optimization problems. In *Power Tech, 2007 IEEE Lausanne*, pages 1290 – 1295, Lausanne, 2007.

G. E. Evans, J. M. Keith, and D. P. Kroese. Parallel cross-entropy optimization. In *Proceedings of the 2007 Winter Simulation Conference*, pages 2196–2202, Washington, D.C., 2007.

J. Gu, P. W. Purdom, J. Franco, and B. W. Wah. Algorithms for the satisfiability (SAT) problem: A survey. In *Satisfiability Problem: Theory and Applications*, pages 19–152. American Mathematical Society, Providence, RI, 1997.

P. Hintsanen, H. Toivonen, and P. Sevon. Fast discovery of reliable subnetworks. In *2010 International Conference on Advances in Social Network Analysis and Mining*, pages 104 – 111, 2010.

H. H. Hoos and T. Stützle. SATLIB: An online resource for research on SAT. In: SAT 2000, I. P. Gent, H. v. Maaren, T. Walsh, editors, pages 283–292. `www.satlib.org`, IOS Press, 2000.

J. Keith and D. P. Kroese. Sequence alignment by rare event simulation. In *Proceedings of the 2002 Winter Simulation Conference*, pages 320–327, San Diego, 2002.

H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer Verlag, first edition, 2004.

R. P. Kothari and D. P. Kroese. Optimal generation expansion planning via the cross-entropy method. In M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, editors, *Proceedings of the 2009 Winter Simulation Conference.*, pages 1482–1491, 2009.

D. P. Kroese, S. Porotsky, and R. Y. Rubinstein. The cross-entropy method for continuous multi-extremal optimization. *Methodology and Computing in Applied Probability*, 8(3):383–407, 2006.

D. P. Kroese, K.-P. Hui, and S. Nariai. Network reliability optimization via the cross-entropy method. *IEEE Transactions on Reliability*, 56(2):275–287, 2007a.

D. P. Kroese, R. Y. Rubinstein, and T. Taimre. Application of the cross-entropy method to clustering and vector quantization. *Journal of Global Optimization*, 37:137–157, 2007b.

Z. Liu, A. Doucet, and S. S. Singh. The cross-entropy method for blind multiuser detection. In *IEEE International Symposium on Information Theory*, Chicago, 2004. Piscataway.

A. Lörincza, Z. Palotaia, and G. Szirtesb. Spike-based cross-entropy method for reconstruction. *Neurocomputing*, 71(16–18):3635–3639, 2008.

L. Margolin. On the convergence of the cross-entropy method. *Annals of Operations Research*, 134(1):201–214, 2005.

G. J. McLachlan and D. Peel. *Finite Mixture Models*. John Wiley & Sons, New York, 2000.

I. Menache, S. Mannor, and N. Basis function adaptation in temporal difference reinforcement learning. *Annals of Operations Research*, 134(1):215–238, 2005.

V. Pihur, S. Datta, and S. Datta. Weighted rank aggregation of cluster validation measures: a Monte Carlo cross-entropy approach. *Bioinformatics*, 23 (13):1607–1615, 2007.

D. Reichelt, F. Rothlauf, and P. Bmilkowsky. Designing reliable communication networks with a genetic algorithm using a repair heuristic. In *Evolutionary Computation in Combinatorial Optimization*, pages 177–186. Springer-Verlag, Heidelberg, 2007.

G. Rubino. Network reliability evaluation. In J. Walrand, K. Bagchi, and G. W. Zobrist, editors, *Network Performance Modeling and Simulation*, chapter 11. Blackwell Scientific Publications, Amsterdam, 1998.

G. Rubino and B. Tuffin. *Rare Event Simulation*. John Wiley & Sons, New York, 2009.

R. Y. Rubinstein. Combinatorial optimization, cross-entropy, ants and rare events. In S. Uryasev and P. M. Pardalos, editors, *Stochastic Optimization: Algorithms and Applications*, pages 304–358, Dordrecht, 2001. Kluwer.

R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

R. Y. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190, 1999.

R. Y. Rubinstein and D. P. Kroese. *Simulation and the Monte Carlo Method*. John Wiley & Sons, New York, second edition, 2007.

R. Y. Rubinstein and D. P. Kroese. *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte Carlo Simulation and Machine Learning*. Springer-Verlag, New York, 2004.

A. Sani. *The Spread of HIV/AIDS in Mobile Populations: Modelling, Analysis, Simulation*. LAP LAMBERT Academic Publishing, Berlin, 2010.

A. Sani. *Stochastic Modelling and Intervention of the Spread of HIV/AIDS*. PhD thesis, The University of Queensland, Brisbane, 2009.

A. Sani and D. P. Kroese. Controlling the number of HIV infectives in a mobile population. *Mathematical Biosciences*, 213(2):103–112, 2008.

S. Senju and Y. Toyoda. An approach to linear programming with 0-1 variables. *Management Science*, 15(4):B196–B207, 1968.

Z. Szabó, B. Póczos, and A. Lörincz. Cross-entropy optimization for independent process analysis. In *Independent Component Analysis and Blind Signal Separation*, volume 3889, pages 909–916. Springer-Verlag, Heidelberg, 2006.

A. Unveren and A. Acan. Multi-objective optimization with cross entropy method: Stochastic learning with clustered pareto fronts. In *IEEE Congress on Evolutionary Computation (CEC 2007)*, pages 3065–3071, 2007.

G.-B. Wang, H.-Z. Huang, Y. Liu, X. Zhang, and Z. Wang. Uncertainty estimation of reliability redundancy in complex systems based on the Cross-Entropy method. *Journal of Mechanical Science and Technology*, 23:2612–2623, 2009.

J.-M. Won and F. Karray. Cumulative update of all-terminal reliability for faster feasibility decision. *IEEE Transactions on Reliability*, 59(3):551–562, 2010.

Y. Wu and C. Fyfe. Topology perserving mappings using cross entropy adaptation. In *7th WSEAS international conference on artificial intelligence, knowledge engineering and data bases*, Cambridge, 2008.

A. Wuerl, T. Crain, and E. Braden. Genetic algorithm and calculus of variations-based trajectory optimization technique. *Journal of spacecraft and rockets*, 40(6):882–888, 2003.