The cross-entropy method for power system combinatorial optimization problems

Damien Ernst *, Mevludin Glavic †, Guy-Bart Stan ‡, Shie Mannor § and Louis Wehenkel ¶
* Supélec, France, Email: damien.ernst@supelec.fr
† University of Liège, Belgium, Email: glavic@montefiore.ulg.ac.be
‡ University of Cambridge, United Kingdom, Email: gvs22@eng.cam.ac.uk
§ McGill University, Canada, Email: shie.mannor@mcgill.ca
¶ University of Liège, Belgium, Email: l.wehenkel@ulg.ac.be

***Abstract*— We present an application of a cross-entropy based combinatorial optimization method for solving some unit commitment problems. We report simulation results and analyze, under several perspectives (accuracy, computing times, ability to solve efficiently large-scale problems), the performances of the approach.**

Keywords: cross-entropy method, combinatorial optimization, power systems.

I. Introduction

Many problems faced by power system engineers can be formalized as combinatorial optimization problems. As examples, we mention transmission network expansion planning problems where one needs to evaluate the best possible combination of single investments [16], the unit commitment problem [11], the optimal placement of phasor measurements units for state estimation [12] or the design of special protection schemes [13].

We can write a pure combinatorial optimization problem using the generic form:

$$u^* = \arg\max_{u \in \mathcal{U}} S(u), \tag{1}$$

where $S(\cdot)$ denotes the performance function and $\mathcal{U}$ the discrete search space of the optimization problem. Typically, $\mathcal{U}$ is of large dimension − that is, an element $u = (u[1], u[2], \ldots, u[n])$ of the search space is described by a large number of components. Solving a combinatorial problem through simple enumeration has a complexity which grows exponentially with the dimension of the search space, making such an approach hardly applicable when dealing with large dimensional $\mathcal{U}$.

To curb this exponential growth problem, two types of strategies are commonly adopted. The first one exploits the particular structure of the performance function $S(u)$ to design 'customized algorithms'. We cite for example the integer programming problems [7] which are particular instances of (1) where the performance function is given by $S(u) = c_1 u[1] + c_2 u[2] + \ldots + c_n u[n]$ and for which various 'customized techniques', as for example those based on cutting planes [2] or branch-and-bound [20], have been developed.

Randomized algorithms such as genetic algorithms [9], simulated annealing [1], ant colony optimization [5], tabu search [8] or nested partitioning [19] constitute the second family of strategies. These are iterative algorithms for which the solution given at the next iteration is a 'stochastic refinement' of the current solution. They have been shown to be able to efficiently identify good solutions of numerous real-world problems and have gained an immense popularity in the scientific community. In particular, references [3], [11], [12], [13], [14], [16], [21], are only a small sample of the success met by these methods when applied to power system combinatorial problems.

Around 1997, the so-called Cross-Entropy (CE) method was proposed by R.Y. Rubinstein for solving rare-event simulation problems [17] and was afterwards extended to solving combinatorial problems. Several randomized optimization algorithms based on the CE method have been proposed in the literature and have been shown to lead to good performances on numerous optimization problems, often outperforming other randomized algorithms [18].

While applications of these CE-based techniques to various fields of engineering have already been reported (see, e.g. [4], [10]), they have, to our best knowledge, not been applied yet to power systems.

In this paper, we introduce the cross-entropy method to the power system community by illustrating its application on problems for which one has to compute short-term production plans to generate electricity at minimal cost. These problems are commonly referred to as unit commitment problems. Based on the simulation results, we also will analyze, under several perspectives (accuracy, computing times, ability to solve efficiently large-scale problems), the performances of the CE-based optimization algorithm.

The paper is organized as follows. Section II focuses on the CE method and describes a practically implementable CE-based combinatorial algorithm. Section III discusses the results obtained by running this algorithm on unit commitment problems. Finally, Section IV concludes and the Appendix provides a detailed description of the benchmark problem used.

II. The CE method

We start this section by describing the CE method in the rare-event framework. Afterwards, we explain how to adapt

algorithms developed in this framework for solving combinatorial optimization problems. Finally, we describe a practically implementable and well-performing CE-based combinatorial algorithm for problems whose search spaces are n-dimensional binary spaces, i.e. $\mathcal{U} = \{0,1\}^n$.

The material of this section is largely borrowed from [17] to which we refer the reader for a complement of information.

A. The CE method for rare event simulation

Let X be a random variable taking its value in some discrete space $\mathcal{X}$ with a probability mass function (pmf) $f(\cdot)$, $S'(\cdot)$ be a real-value function defined on $\mathcal{X}$ and γ be a real number. In the rare-event simulation context, one needs to estimate the probability of occurrence l of an event $\{S'(X) \geq \gamma\}$, i.e. to estimate the expression $E_{X\sim f(\cdot)}\left[I_{\{S'(X)\geq\gamma\}}\right]$.[1]

In rare-event simulation problems, this probability is extremely low, say less than 10^{-6}, and estimating l with enough accuracy by relying on a Crude Monte-Carlo (CMC) estimator

$$\hat{l} = \frac{1}{N}\sum_{j=1}^{N} I_{\{S'(X_j)\geq\gamma\}} \tag{2}$$

requires to draw a considerably large sample X_1, X_2, ..., X_N from $f(\cdot)$. For example, for estimating $l = 10^{-6}$, with a relative error $\kappa = 0.01$, a sample size of $N \simeq \frac{1}{\kappa^2 l} = 10^{10}$ is required, which shows that it is generally computationally meaningless to estimate small probabilities via CMC.

An alternative to CMC is based on importance sampling. With such an approach, a random sample X_1, X_2, ..., X_N is drawn from an importance sampling pmf $g(\cdot)$ and the probability of occurrence of the event is estimated via the following unbiased estimator

$$\hat{l} = \frac{1}{N}\sum_{j=1}^{N} I_{\{S'(X_j)\geq\gamma\}}\frac{f(X_j)}{g(X_j)} \quad . \tag{3}$$

The best way to estimate l is to adopt the 'ideal' importance sampling pmf

$$g^*(X) = \frac{I_{\{S'(X)\geq\gamma\}}f(X)}{l} \quad . \tag{4}$$

Since l is constant using this 'ideal' importance sampling (4) would lead to an estimator (3) having a zero variance. Consequently, we would need to produce only a one element sample to determine l.

The obvious difficulty is that $g^*(\cdot)$ depends on the unknown parameter l.

The main idea of the CE method for rare event simulation is to find inside an a priori given set $\mathcal{G}$ of pmfs defined on $\mathcal{X}$, the element $g(\cdot)$ such that its distance with the 'ideal' sampling distribution is minimal.

A particularly convenient measure of distance between two pmfs $a(\cdot)$ and $b(\cdot)$ on $\mathcal{X}$ is the Kullback-Leibler distance, which is also termed the cross-entropy between $a(\cdot)$ and $b(\cdot)$. The Kullback-Leibler distance, which is not a "distance " in the formal sense since it is for example not symmetric, is defined as follows:

$$\mathcal{D}(a,g) = E_{X\sim a(\cdot)}\left[\ln\frac{a(X)}{b(X)}\right] \tag{5}$$

The CE method reduces the problem of finding an appropriate importance sampling pmf to the following optimization problem:

$$\arg\min_{g\in\mathcal{G}}\mathcal{D}(g^*,g) \quad . \tag{6}$$

One can show through simple mathematical derivations that solving (6) is equivalent to solve:

$$\arg\max_{g\in\mathcal{G}} E_{X\sim f(\cdot)}\left[I_{\{S'(X)\geq\gamma\}}\ln g(X)\right] \tag{7}$$

which does not depend explicitly on l anymore.

If l is not too small, CE-based algorithms for rare-event simulations estimate a good solution of (7) by solving its stochastic counterpart:

$$\arg\max_{g\in\mathcal{G}}\sum_{j=1}^{M} I_{\{S'(X_j)\geq\gamma\}}\ln g(X_j) \tag{8}$$

where the sample X_1, X_2, ..., X_M is drawn according to $f(\cdot)$. When l is too small, say $l < 10^{-6}$, which is often the case in rare-event simulation, the value of M one has to adopt for having a 'good' stochastic counterpart may be prohibitively high and some specific iterative techniques need to be adopted to solve (7). The use of these techniques is often equivalent to solving a sequence of rare event problems using the same pmf $f(\cdot)$ and function S' but with increasing values of γ converging to the value of γ related to the original problem.

Under some specific assumptions on $\mathcal{X}$, $f(\cdot)$ and $\mathcal{G}$, it is possible to solve analytically the optimization problem (8). This property is often exploited in the CE context.

For example, let us suppose that $\mathcal{X} = \{0,1\}^n$ and let us denote by $Ber_n(\cdot,p)$ the n-dimensional Bernoulli pmf

$$Ber_n(X,p) = \Pi_{i=1}^{n} p[i]^{X[i]}(1-p[i])^{1-X[i]} \tag{9}$$

where p is a vector of parameters belonging to the n-cube (i.e., $[0,1]^n$), and often referred to as the vector of probabilities associated with the Bernoulli distribution, and where $X[i]$ is the ith component of the random variable X.

Then, one can show that if $f(\cdot)$ is a n-dimensional Bernoulli pmf and $\mathcal{G}$ is the set of all n-dimensional Bernoulli pmfs, the solution $Ber_n(\cdot,p^*)$ of (8) can be computed analytically:

$$p^*[i] = \frac{\sum_{j=1}^{M} I_{\{S'(X_j)\geq\gamma\}}X_j[i]}{\sum_{j=1}^{M} I_{\{S'(X_j)\geq\gamma\}}} \quad . \tag{10}$$

[1] The function $I_{\{logical\_expression\}}$ is defined by $I_{\{logical\_expression\}} = 1$ if $logical\_expression = true$ and 0 otherwise. The expression $E_{X\sim f(\cdot)}\left[I_{\{S'(X)\geq\gamma\}}\right]$ can be written equivalently as $\sum_{X\in\mathcal{X}} I_{\{S'(X)\geq\gamma\}}f(X)$.

Input: A performance function $S : \mathcal{U} \rightarrow \mathbb{R}$ where $\mathcal{U} = \{0,1\}^n$ and two parameters: C and ϱ.
Output: An element $u_{output} \in \mathcal{U}$.
Algorithm:

Step 1. Set t equal to 1 and the components $p_t[i]$ of the n-dimensional vector p_t equal to 0.5.
Set $nbElite$ equal to the largest integer inferior or equal to $\varrho \times C \times n$. If $nbElite < 1$ then set $nbElite$ to 1.

Step 2. Set U_t equal to an empty set and r_t to an empty vector.

Step 3. Draw independently $C \times n$ elements according to the Bernouilli pmf $Ber_n(\cdot, p_t)$ and set them in U_t.

Step 4. For every element $u \in U_t$, compute $S(u)$ and add this value at the end of the vector r_t.

Step 5. Order the vector r_t in decreasing order and set $\hat{\gamma}_t = r_t[nbElite]$.

Step 6. If *stopping conditions* are met, then return $u_{output} = \underset{u \in U_1 \cup U_2 \cup \ldots \cup U_t}{\arg\max} S(u)$ and stop. Otherwise go to **Step 7.**

Step 7. Set $p_{t+1}[i] = \frac{\sum_{u \in U_t} I_{\{S(u) \geq \hat{\gamma}_t\}} u[i]}{nbElite}$ for $i = 1, 2, \ldots, n$ and $t \leftarrow t + 1$. Go to **Step 2.**

Fig. 1. A cross-entropy based combinatorial optimization algorithm for search spaces composed of vectors of binary variables.

B. From rare event simulation to combinatorial optimization

The main ideas of the CE algorithm for solving the combinatorial optimization problem described by Eqn (1) are based on the following two observations. First, the event $\{S(u) \geq \gamma = S(u^*)\}$, where u is a random variable taking its values in $\mathcal{U}$ with a pmf $f_u(\cdot)$, tends to be a rare event. For example, if $f_u(\cdot)$ is a uniform pmf possessing a unique maximum, the probability of this event is $\frac{1}{\#\mathcal{U}}$. Second, when solving this particular type of rare-event problem by using a CE method, one generally obtains, as byproduct, a pmf which is close to the 'ideal' one, and, therefore, likely to generate samples u for which the value of $S(u)$ is close to maximal. Based on these two observations, several practically implementable randomized algorithms have been proposed.

These algorithms often exploit the following iterative scheme. Since the value of $S(u^*)$ is unknown, they start by drawing a random sample U_1 according to a pmf $g_1(\cdot)$ given as input of the algorithm (and often chosen as the uniform pmf) and from this U_1 compute $\gamma_1 = \max_{u \in U_1} S(u)$. Afterwards, they solve the rare-event simulation problem $E_{u \sim g_1(\cdot)} \left[I_{\{S(u) \geq \gamma_1\}} \right]$ from which they deduce by solving expression (7), where the set $\mathcal{G}$ is an input of the algorithm, a pmf $g_2(\cdot)$. At the second iteration, they use $g_2(\cdot)$ to draw a new sample U_2 from which they deduce $\gamma_2 = \max_{u \in U_2} S(u)$. Then, they solve the rare-event simulation problem defined by $g_2(\cdot)$, γ_2 and U_2. From the solution of this rare event problem, they deduce a pmf $g_3(\cdot)$. By proceeding like this, they compute a sequence of pmfs $g_1(\cdot)$, $g_2(\cdot)$, $g_3(\cdot)$, $\ldots$. By assuming, among others, that U_t is large enough and g_t not too far from the 'ideal' sampling distribution corresponding to the rare-event problem defined at iteration t, these pmfs become more likely to generate samples U_t having elements corresponding to high-values of $S(\cdot)$ when t increases.

C. A practically implementable algorithm

Figure 1 gives the tabular version of a CE-based optimization algorithm for combinatorial problems whose search spaces are of the type $\mathcal{U} = \{0,1\}^n$. This algorithm is based on the iterative scheme described at the end of previous subsection and is particularized to the case where the set $\mathcal{G}$ is the set of all n-dimensional Bernoulli pmfs defined on $\mathcal{U}$. The algorithm solves the optimization problem (7) by relying on its stochastic counterpart (8) whose solution is computed analytically by exploiting (10).

There is however one notable difference between the iterative scheme adopted by the algorithm of Fig. 1 and the one described in previous subsection. Indeed, the rare-event problem the algorithm of Fig. 1 solves at iteration t does not have a value of γ equal to $\gamma_t = \max_{u \in U_t} S(u)$. Instead, this value is equal to $\hat{\gamma}_t$ with $\hat{\gamma}_t$ defined in such a way that only a small fraction ϱ of the elements $u \in U_t$ lead to a value $S(u)$ larger or equal to $\hat{\gamma}_t$. These elements having a value of $S(u) \geq \hat{\gamma}_t$ are often referred to in the CE literature as the 'elite elements' or 'elite samples'. The main reason for using $\hat{\gamma}_t$ rather than γ_t to define the rare-event problem solved at iteration t is to ensure that this problem does not correspond to a too small value of l. Indeed, if l is too small, one would be required to draw a prohibitively large sample to have a stochastic counterpart (8) whose solution is accurate enough.

We note also that the algorithm uses at iteration t the sample U_t to define the stochastic counterpart (8) of (7).

The algorithm has two parameters C and ϱ. The parameter C determines the size of the samples U_t in a way that $\#U_t = C \times n$. The rationale behind adopting samples U_t whose cardinality is growing with n is that usually, the larger the search space is, the larger the samples U_t one has to draw for the algorithm to be able to output an element u_{output} corresponding to a high-value of the function $S(\cdot)$. We suggest a default value for this parameter $C = 10$. As explained before, the parameter ϱ determines the number of 'elite samples' and we adopt for this parameter a default value of $\varrho = 0.1$.

When the stopping conditions are met, whatever they are, the algorithm returns the element $u_{output} = \underset{u \in U_1 \cup U_2 \cup \ldots \cup U_t}{\arg\max} S(u)$ which represents the best elements among those which have been evaluated throughout the course the algorithm.

We found out when carrying our simulations that the sequence of Bernoulli pmfs $Ber_n(\cdot, p_1)$, $Ber_n(\cdot, p_2)$, $\ldots$ gen-

erated by this algorithm was often converging in a relatively small number of iterations to a degenerate pmf, that is a pmf that assigns all the probability, i.e. probability 1, to a single element of $\mathcal{U}$. When convergence to a degenerate pmf occurs, the algorithm can be stopped since it will only produce identical degenerate pmfs afterwards.

III. Simulation results

In this section, we discuss some simulation results obtained by running the CE-based optimization algorithm described on Fig. 1 on an academic unit commitment problem.

We start by giving a description of the main characteristics of this problem. Then, we comment on a typical single run of this algorithm. And finally, we study the performances of the algorithm for various sizes of this problem. All the results have been generated by using the algorithm with its default values: $\varrho = 0.1$ and $C = 10$.

A. The benchmark unit commitment problem

The unit commitment problem is the problem of scheduling at minimum cost the production of electricity under various constraints. Many formulations of this problem have been proposed (e.g., security constraint unit commitment problem [6], profit-based unit commitment [15]). We adopt here a simple formulation of this problem which, among others, has no explicit constraints and assumes that the power produced by a generator can only be either zero or maximal. We give, for the sake of reproducibility of the results, a detailed description of this benchmark unit commitment problem in the Appendix.

The problem has been stated in a way that 'its size' depends on a parameter n that represents both the number of generators and the dimensionality of the combinatorial search space. These generators are indexed by a number i ($i = 1, 2, \ldots, n$) and produce electrical power at a cost which grows linearly with i. By assuming that n is a multiple of 10, one can show that it is optimal to schedule only generators 1, 2, ..., $\frac{8\times n}{10}$ (see the Appendix for more details).

The ith component of an element $u = (u[1], u[2], \ldots, u[n]) \in \mathcal{U}$ is binary and determines whether the generator i is scheduled for production or not. It is equal to 1 if the machine is scheduled and zero otherwise.

B. An illustration of a typical single run of the algorithm

Figure 2 describes a typical single run of CE-based combinatorial algorithm with its default values on the benchmark unit commitment problem defined by $n = 20$. This leads to a combinatorial optimization problem having a rather small search space ($\#\mathcal{U} = 1,048,576$) and, therefore, for which we could identify the optimal solution by relying on exhaustive search. However, we found out that for illustrative purposes it was more convenient to consider here a small value of n.

The first column of the table gives the iteration number t, the next twenty columns the components of p_t and the last one the largest value that $S(\cdot)$ takes on U_t. As one can see, the pmfs $Ber_n(\cdot, p_t)$ have generated samples whose best elements are leading to increasing values of $S(\cdot)$ when t starts increasing. After a few iterations, the best element of U_t always leads to a value of $S(\cdot)$ equal to -92210.5, which is actually the maximum value $S(\cdot)$ can achieve over $\mathcal{U}$. At iteration $t = 6$, the corresponding pmf ($Ber_n(\cdot, p_6)$) is a degenerate pmf which assigns a probability 1 to the optimal element u^* and the algorithm is stopped.

C. Performances of the CE-based combinatorial algorithm

Stochastic optimization algorithms, such as the CE-based combinatorial algorithm used in this paper, have essentially an anytime behaviour − that is, they can return the best answer possible even if they are not allowed to run to completion and may improve on the answer if they are allowed to run longer. When studying performance of such algorithms, one is usually interested in establishing the relation that exists between the quality of their answer and the time they are allowed to run as well as in establishing how this relation evolves with the size of the problem.

To illustrate the performances of the CE-based combinatorial algorithm on the benchmark unit commitment problem, we have chosen to run simulations to determine for several sizes n of the optimization problem (i) the relation that represents the distance between the solution the CE-based algorithm would provide if stopped after t iterations ($u_t = \arg\max_{u \in U_1 \cup U_2 \cup \ldots \cup U_t} S(u)$) and the optimal solution (u^*) (ii) the CPU time associated with an iteration of the algorithm, which for a given size n of the optimization problem is essentially constant whatever the iteration t.

The results of these experiments are reported on Fig. 3. We used as distance measure between u_t and u^* the expression $(S(u^*) - S(u_t))/(S(u^*))$ rather than $(S(u^*) - S(u_t))$ to make this measure of suboptimality somehow independant of the size n of the optimization problem. An analysis of Fig. 3 shows that for every value of n, the suboptimality of the solution seems to decrease to 0 when t increases. It also shows that to reach a given degree of suboptimality, one needs to carry out more iterations when n increases. Finally, one can see that the CPU time per iteration grows more that linearly with the size of the optimization problem n. Actually, one can show that the CPU time per iteration grows quadratically with n. This quadratic growth of the CPU times with n is a consequence of the fact that $\#U_t$ increases linearly with n and that the time needed to generate each element of U_t and evaluate its performance $S(\cdot)$ also grows linearly with n.

We ran the CE-based optimization algorithm one thousand times for different values of n (25, 50, 100, 200 and 400) and observed that it converged every time to degenerate pfms with a reasonable number of iterations. Figure 4 reports for different values of n (i) the average number of iterations to convergence, (ii) the average suboptimality of the solution outputted by the algorithm[2] and (iii) the probability that the algorithm identifies an optimum. The main two observations to be drawn from this table are the following. First, we observe that the number of

[2]This average suboptimality is here equivalent to the asymptotic suboptimality since we only stop the algorithm when it has converged.

t	$p_t[1]$	$p_t[2]$	$p_t[3]$	$p_t[4]$	$p_t[5]$	$p_t[6]$	$p_t[7]$	$p_t[8]$	$p_t[9]$	$p_t[10]$	$p_t[11]$	$p_t[12]$	$p_t[13]$	$p_t[14]$	$p_t[15]$	$p_t[16]$	$p_t[17]$	$p_t[18]$	$p_t[19]$	$p_t[20]$	$\max_{u \in U_t} S(u)$
1	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.5	-100316.0
2	0.75	0.75	0.8	0.7	0.9	0.8	0.8	0.6	0.8	0.65	0.65	0.6	0.95	0.5	0.35	0.55	0.5	0.6	0.5	0.45	-95526.3
3	0.9	1	1	1	0.9	0.95	1	0.8	0.9	0.9	0.75	0.8	1	0.3	0.45	0.75	0.4	0.65	0.25	0.5	-92578.9
4	1	1	1	1	1	1	1	0.92	0.96	0.96	0.89	1	1	0.42	0.64	0.67	0.32	0.60	0.07	0.46	-92210.5
5	1	1	1	1	1	1	1	1	1	1	1	1	1	0.54	0.90	0.80	0.06	0.58	0.06	0.03	-92210.5
6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	-92210.5

Fig. 2. A typical run of the cross-entropy based algorithm for combinatorial optimization.

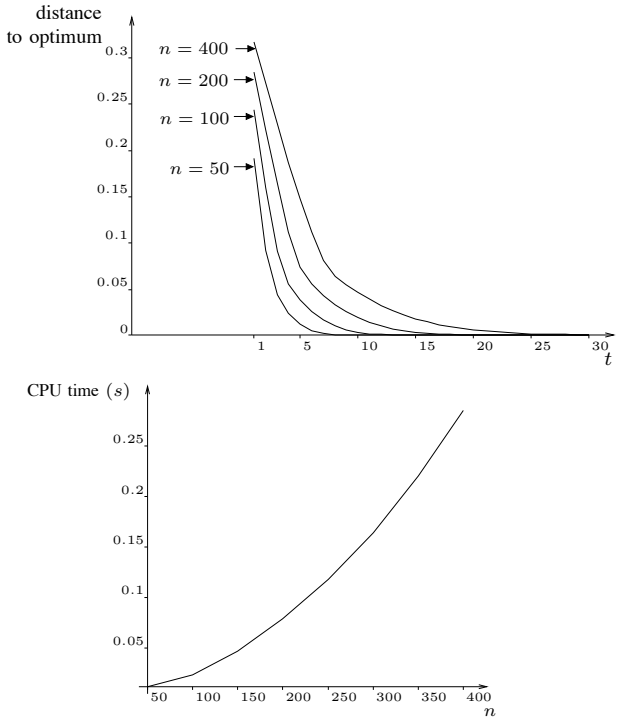

Fig. 3. Some insights into the computational complexity of the CE-based optimization algorithm for solving the unit commitment problems. The figure on the top-row plots for different values of n the expected distance between the element $u_t = \arg\max_{u \in U_1 \cup U_2 \cup \ldots \cup U_t} S(u)$ and the optimum u^* as a function of the number of iterations. The distance between u^* and u_t is defined as $(S(u^*) - S(u_t))/(S(u^*))$. The figure on the bottom-row represents the average CPU times (in seconds on an AMD 2800+/1.6 GHz processor) per iteration of the CE-based optimization algorithm as a function of the size n of the unit commitment problem. The results have been averaged over 1000 runs of the algorithm.

iterations to convergence increases slightly less than linearly with n. Moreover, the number of elements of $\mathcal{U}$ for which the performance $S(\cdot)$ must be evaluated is equal to $C \times n$ per iteration, where C is a constant. As a result, the number of evaluations of the performance function $S(\cdot)$ required before convergence grows less than quadratically with n. Knowing that $\#\mathcal{U}$ grows exponentially with n, we have therefore an exponential decrease with n of the percentage of elements $u \in \mathcal{U}$ whose performances are assessed throughout the course of the algorithm. This percentage is equal on average to 5.97×10^{-3} if $n = 25$ and drops to 5.26×10^{-10}, 1.39×10^{-24}, 3.30×10^{-54} and 6.26×10^{-114} when n is equal to 50, 100, 200 and 400, respectively.

Second, and perhaps surprisingly, the accuracy of the algorithm in asymptotic conditions is the best for the largest values of n. In particular, we found out that over 1000 runs, the algorithm was only once unable to identify the optimum when n was equal to 200 or 400.

n	average number of iter. for $Ber_n(\cdot, p_t)$ to degenerate	average suboptimality of the algorithm	probability identifying the optimum u^*
25	8.02	7.833×10^{-5}	0.98
50	11.86	2.222×10^{-5}	0.974
100	17.63	1.678×10^{-6}	0.99
200	26.57	3.792×10^{-8}	0.999
400	40.45	9.455×10^{-9}	0.999

Fig. 4. Influence of the size n of the combinatorial optimization problem on (i) the average number of iterations after which $Ber_n(\cdot, p_t)$ degenerates, (ii) the average suboptimality of the algorithm measured as $(S(u^*) - \max_{u \in U_1 \cup U_2 \cup \ldots \cup U_{t_{end}}} S(u))/(S(u^*))$ (iii) the probability the algorithm identifies u^* over one run, that is the probability that $u^* \in U_1 \cup U_2 \cup \ldots \cup U_{t_{end}}$ where t_{end} refers to the first iteration to which corresponds a degenerate pmf. These results have been generated by running for every value of n the CE-based combinatorial algorithm one thousand times.

IV. Conclusions

In this paper, we reported some results obtained by running a combinatorial algorithm based on the CE method on an academic class of unit commitment problems. We found out that the algorithm, with its suggested default parameters, has good performances for this class of problems.

The experiments suggest that CE-based tools can certainly be useful for solving power system combinatorial problems, even if it is not clear whether these methods would perform better that other stochastic optimization algorithms (e.g., genetic algorithms, nested partitioning, ant colony optimization). In this respect, it would certainly be interesting for the power system community to define a library of benchmarks for power system combinatorial problems that researchers could use to assess the performances of their approaches, something we found out was missing.

We underline that the CE-based concepts could be exploited to solve other power system related problems than combinatorial optimization ones. For example, by assuming we have a probabilistic description of the operating conditions of a power system and of the possible contingencies, we could exploit these concepts in the rare-event framework to estimate the (small) probability that the integrity of the system may be lost and also to identify the pairs "operating condition-contingency" that lead to this loss of integrity. The CE method

could also be applied to the resolution of continuous and mixed-variable power system optimization problems.

APPENDIX

We describe in this appendix the unit commitment problem used in our simulations.

This unit commitment problem has n generators indexed by 1, 2, ..., n. Generator i can be turned on or off. If turned on, it produces a power equal to $P[i]$ MW at a cost $C[i]$ \$ per Megawatt hour (MWh). We associate to every generator i a so-called *utility variable* $u[i]$, which is equal to 1 if the generator is on and 0 if it is off. We want to determine the value of the vector $u \in \{0,1\}^n$ that minimizes

$$costProduction(u) = \sum_{i=1}^{n} C[i]P[i]u[i] + \begin{cases} 0 & \text{if } \sum_{i=1}^{n} P[i]u[i] \geq P_l \\ (P_l - \sum_{i=1}^{n} P[i]u[i]) \times penalty & \text{otherwise} \end{cases}. \quad (11)$$

Expression (11) represents the sum of the production costs over one hour to which a penalty term is added if the scheduled generators cannot cover the load P_l.

The symbols n, $P[i]$, $C[i]$, P_l, $penalty$ are parameters of the optimization problem. They have been chosen as follows in our simulations:

1) the number of generators is equal to a multiple of 10.
2) the powers $P[i]$ are equal to 100 MW whatever i.
3) $C[i] = 30 + \frac{(i-1)}{n-1} \times 70$, that is the cost of production per MWh grows linearly with the index number of a generator. It is equal to 30\$/MWh for generator 1 and 100\$/MWh for generator n.
4) the value of P_l is equal to $\frac{8 \times n}{10} \times 100$, which means that, since n is a multiple of 10, eighty percents of the generators need to be turned on to cover the load.
5) the $penalty$ factor is equal to 110\$/MWh. The penalty to be paid for not covering a certain amount of the load P_l is therefore larger than the cost of producing this amount of load with the generators whose production costs are the highest.

With such a choice of parameters, the optimal solution consists of dispatching the machines 1, 2, ..., $\frac{8 \times n}{10}$ and the minimum cost is equal to: $\sum_{i=1}^{\frac{8 \times n}{10}} [(30 + \frac{(i-1)}{n-1} \times 70) \times 100]$. While the unit commitment problem has been formulated here as a minimization problem, we treat it in our simulation result section as a maximization problem by taking a performance function $S(u)$ equal to $-costFunction(u)$ everywhere on $\mathcal{U}$.

REFERENCES

[1] E.H.L. Aarts and J.H.M. Korst. Simulated Annealing and Boltzmann Machines. *John Wiley & Sons*, 1989.

[2] S. Ceria, C. Cordier, H. Marchand, and L.A. Wolsey. Cutting planes for integer programs with general integer variables. *Mathematical programming*, 81(2):201–214, April 1998.

[3] C.S. Chang, L. Tian, and F.S. Wen. A new approach to fault section estimation in power systems using Ant system. *Electric Power Systems Research*, 49(1):63–70, 1999.

[4] P.T. de Boer. *Analysis and Efficient Simulation of Queueing Models of Telecommunication Systems*. PhD thesis, University of Twente, 2000.

[5] M. Dorigo, V. Maniezzo, and A. Colorni. The ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics - Part B*, 26(1):26–41, 1996.

[6] Y. Fu, M. Shadidehpour, and Z. Li. Security-constrained unit commitment with AC constraints. *IEEE Transactions on Power Systems*, 20(3):1538–1550, August 2005.

[7] R.S. Garfinkel and G.L. Nemhauser. *Integer Programming*. John Wiley & Sons, New-York, 1972.

[8] F. Glover. Tabu search - part ii. *ORSA Journal on Computing*, 2:4–32, 1990.

[9] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[10] B.E. Helvik and O. Wittner. Using the cross-entropy method to guide/govern mobile agent's path finding in networks. In *3rd International Workshop on Mobile Agents for Telecommunication Applications − MATA'01*, pages 255–268, 2001.

[11] S.A. Kazarlis, A.G. Bakirtzis, and V. Petridis. A genetic algorithm solution to the unit commitment problem. *IEEE Transactions on Power Systems*, 11(1):83–92, February 1996.

[12] F.J. Marin, F. Garcia-Lagos, G. Joya, and F. Sandoval. Genetic algorithms for optimal placement of phasor measurement units in electrical networks. *Electronics Letters*, 39(19):1403–1405, September 2003.

[13] C. Moors, D. Lefebvre, and T. Van Cutsem. Combinatorial optimization approaches to the design of load shedding schemes against voltage instability. In *Proceedings of the 32nd North American Power Symposium (NAPS)*, pages 1014–1021, 2000.

[14] K. Nara, A. Shiose, M. Kitagawa, and T. Ishihara. Implementation of genetic algorithm for distribution systems loss minimum re-configuration. *IEEE Transactions on Power Systems*, 7(3):1044–1051, 1992.

[15] C.W. Richter and G.B. Sheble. A profit based unit commitment GA for the competitive environement. *IEEE Transactions on Power Systems*, 15(2):715–721, May 2000.

[16] R. Romero, R.A. Gallego, and A. Monticelli. Transmission system expansion planning by simulated annealing. *IEEE Transactions on Power Systems*, 11(1):364–369, February 1996.

[17] Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operations Research*, 99:89–112, 1997.

[18] R.Y. Rubinstein and D.P. Kroese. *The Cross-Entropy Method. A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*. Information Science and Statistics. Springer, 2004.

[19] L. Shi and S. Olafsson. Nested partitioning for global optimization. *Operations Research*, 48(3):390–407, 2000.

[20] J.A. Tomlin. An improved branch-and-bound method for integer programming. *Operations Research*, 19(4):1070–1075, 1971.

[21] H. Yoshida, K. Kawata, Y. Fukuyama, S. Takayama, and Y. Nakanishi. A particle swarm optimization for reactive power and voltage control considering voltage security assessment. *IEEE Transactions on Power Systems*, 15(4):1232–1239, November 2000.