On the Design of Multi-type Networks via the Cross-Entropy Method

Sho Nariai and Dirk P. Kroese
Department of Mathematics, University of Queensland, Brisbane 4072, Australia
Email: sho@maths.uq.edu.au, kroese@maths.uq.edu.au

***Abstract*— We apply the cross-entropy method to a network design problem with multi-type links and nodes, in which the network's reliability is to be maximized subject to a fixed budget. Numerical experiments illustrate the simplicity and effectiveness of the method.**

***Index Terms*— Network design, multi-type network, network reliability, cross-entropy method, constrained stochastic optimization.**

I. Introduction

Network design problems can often be formulated in terms of complicated optimization problems, possibly involving discrete and/or continuous variables, multiple constraints, and noise. Standard generic stochastic algorithms such as genetic/evolutionary algorithms [1], [2] and simulated annealing [3], [4] may provide a viable way to solve many of these problems, but are frequently found to be not flexible enough [5]. The purpose of this paper is to introduce the cross-entropy (CE) method [6] as a powerful and flexible new way to solve a wide range of design problems.

Most papers on optimal design deal with the selection of a single type of link between each node pair, where the objective is to either maximize the reliability subject to a cost constraint, or minimize the cost subject to a network reliability constraint, see for example [7]–[9] and references therein. In this paper we focus on a multi-type design problem. Specifically, the problem is to design a network by selecting or purchasing predefined nodes and links of various types, subject to a fixed budget, so that the system reliability is maximized. The model considered is a generalization of both [10] and [11]. In the former model, every node and link has to be purchased; however, various link and node types can be selected. In the latter model, the network design involves only single-type links and no nodes.

The CE method is an adaptive Monte Carlo technique for both optimization and rare event simulation, which has attracted considerable interest around the world. It is based on a simple iterative procedure where each iteration contains two phases: (a) generate a random data sample (trajectories, vectors, etc.) according to a specified mechanism, (b) update the parameters of the random mechanism based on the data to produce a "better" sample in the next iteration. This last step involves cross-entropy minimization. The strengths of the method are (1) its simplicity (CE programs are easy to write and require little "tuning"), (2) generality (it can handle without much alteration integer, discrete and continuous and noisy problems) and (3) accuracy. The CE method has been applied to a great range of problems in operations research, including buffer allocation [12], the stochastic shortest path problem [13], the vehicle routing problem [14], queueing models of telecommunication systems [15], network reliability [16], and network planning [11]. An introductory treatment of the CE method can be found in the tutorial [17]. For a more comprehensive treatment and further references, we refer to the recent monograph [6]. The CE homepage can be found at `http://www.cemethod.org`.

This research was supported by the Australian Research Council, grant number DP0558957.

It is out of the scope of this paper to discuss the advantages and disadvantages of the multitude of stochastic search algorithms that could be applied to the network design problem. We simply wish to show that the CE method performs very well with respect to all the above criteria (1)–(3). Simulated annealing is probably closest to CE with respect to (1) and (2), although simulated annealing is a local search algorithm (whose performance depends critically on a proper choice for the cooling scheme), while the CE method is a global optimization method.

The rest of the paper is organized as follows. In Section II we formulate the problem in mathematical terms. In Section III we discuss how we tackle the design problem using the CE method. Section IV presents numerical experiments on three test cases, including some comparisons with simulated annealing (SA) and genetic algorithms (GA).

Notation

A	sampling matrix
$c_i(x)$	cost of component i of type x ($C(\boldsymbol{x})$ total cost)
$C_{\max}$	total budget
L	decay parameter for SA
$m,\ n$	number of [links, nodes]
N	sample size for CE
$p_i(x)$	reliability of component i of type x
r	network reliability (r^* optimal)
T	initial temperature for SA
$\boldsymbol{x}$	network topology ($\boldsymbol{x}^*$ optimal, $\boldsymbol{X}$ random)
$\boldsymbol{y}$	system state ($\boldsymbol{Y}$ random)
α	smoothing parameter for CE
$\varphi_{\boldsymbol{x}}$	structure function of topology $\boldsymbol{x}$
ρ	rarity parameter for CE

0-7803-9439-9/05/$20.00 ©2005 IEEE

II. Problem Description

Consider an undirected graph with n nodes and m links. The graph represents a communication network "on sale". That is, each node and link in the graph can be bought for a certain price. Without loss of generality, we may label the nodes $1, \ldots, n$ and the links $n+1, \ldots, n+m$. Suppose that, for each component (node or link) in the network, there are three different types to select from: type 3 is the most reliable and expensive, type 2 is the second most reliable and expensive, and type 1 is the least reliable and cheapest. In addition, type 0 is assigned to a component that we decide not to purchase. The objective is to determine which types to assign to all components, in order to maximize the reliability — defined as the probability that certain specified *terminal nodes* in the network are connected by operational links through operational nodes — subject to a total budget $C_{\max}$.

To identify which types are assigned to which nodes and links, we introduce a *topology vector* $\boldsymbol{x} = (x_1, \ldots, x_{n+m})$, where x_i represents the type of component i, $i = 1, \ldots, n+m$. As an example, consider the network in Figure 1, where five nodes and six links are for sale.

Assume that $\boldsymbol{x} = (1, 3, 1, 0, 3, 2, 0, 2, 1, 3, 0)$. This means that for nodes 2 and 5 the most reliable and expensive type is purchased, the least expensive and unreliable type is purchased for nodes 1 and 3, and node 4 is not bought at all. The same notation applies for the links. Note that because node 4 is not bought it makes no sense to buy links 7 and 11. The set of all topology vectors is denoted by $\mathcal{X}$.

For each component i, let $p_i(x)$ be the reliability of the assigned node or link type $x \in \{0, \ldots, 3\}$, defined as the probability that component of type x is operational. By definition $p_i(0) = 0$, $i = 1, \ldots, n+m$. Similarly, let $c_i(x)$ be the cost of component i of type x. Note that the reliabilities and costs may depend on i. The total cost of a given network topology $\boldsymbol{x}$ is thus

$$C(\boldsymbol{x}) = \sum_{i=1}^{n+m} c_i(x_i). \tag{1}$$

To identify operational components in the network, we define for each component i its *state* by

$$y_i = \begin{cases} 1 & \text{if } x_i \text{ is bought and operational,} \\ 0 & \text{otherwise.} \end{cases}$$

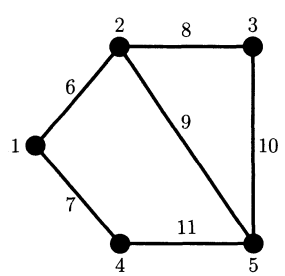

Fig. 1. Network with 5 nodes and 6 links.

The corresponding vector $\boldsymbol{y}$ is called the state vector.

For each vector $\boldsymbol{x}$, let $\varphi_{\boldsymbol{x}}$ be the *structure function* of the system. This function assigns the state of the system determined by $\boldsymbol{y}$. That is,

$$\varphi_{\boldsymbol{x}}(\boldsymbol{y}) = \begin{cases} 1 & \text{if the system is operational,} \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

Now consider random states, where each component i of type x in the network is operational with probability $p_i(x)$, independent of the other components. Let Y_i be the random state of component i, $i = 1, \ldots, n+m$ and let $\boldsymbol{Y}$ be the corresponding random state vector.

The reliability of the network defined by network topology $\boldsymbol{x}$ is given by

$$r(\boldsymbol{x}) = \mathbb{P}(\varphi_{\boldsymbol{x}}(\boldsymbol{Y}) = 1) = \sum_{\boldsymbol{y}} \varphi_{\boldsymbol{x}}(\boldsymbol{y})\, \mathbb{P}(\boldsymbol{Y} = \boldsymbol{y}) \,. \tag{3}$$

The network design problem is thus translated into the following constrained optimization problem:

$$\text{Maximize } r(\boldsymbol{x}), \quad \text{with } \boldsymbol{x} \in \mathcal{X}, \tag{4}$$

subject to

$$C(\boldsymbol{x}) \leq C_{\max}. \tag{5}$$

Let $\boldsymbol{x}^*$ be an optimal solution and $r^* := r(\boldsymbol{x}^*)$ denote the optimal network reliability.

III. Cross-Entropy Method

In this section, we look at how the CE method can be used to solve the constrained optimization problem (4), (5). The CE method in this context involves two steps:

1) Generate a random sample of topology vectors $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N$ according to a specified random mechanism, and
2) Update the parameters of this random mechanism in order to generate a better sample in the next iteration. This last step involves cross-entropy minimization.

By iterating the above two steps, the CE aims to locate an optimal sampling distribution, in this case the sampling distribution which assigns probability mass 1 to $\boldsymbol{x}^*$, corresponding to the optimal network topology.

The mechanism that is used to generate a random topology vector is determined by a $(n+m) \times 4$ stochastic matrix A; where A is updated in each iteration of the algorithm. Denote the i-th row by $\boldsymbol{a}_i = (a_{i0}, \ldots, a_{i3})$, $i = 1, \ldots, n+m$. The generation of $\boldsymbol{X} = (X_1, \ldots, X_{n+m})$ is as follows: First, generate a uniform permutation $(\pi_1, \ldots, \pi_{n+m})$ of $(1, \ldots, n+m)$. This can be done, for example, by independently drawing $n+m$ samples from the uniform distribution on $[0, 1]$ and letting $\pi_1, \pi_2, \ldots$ denote indices of ordered observations. Second, for a given permutation $(\pi_1, \ldots, \pi_{n+m})$, if there is enough money left to purchase type 3 for component π_1, we draw X_{π_1} from $\{0, 1, 2, 3\}$ according to $\boldsymbol{a}_{\pi_1}$. If this is not the case, but there is enough money to by type 2, then draw X_{π_1} from $\{0, 1, 2\}$ according to $\boldsymbol{a}_{\pi_1}$ truncated to this set. Otherwise, if there is only enough money to assign type 1,

draw X_{π_1} from $\{0,1\}$, with probabilities $a_{\pi_1 0}/(a_{\pi_1 0}+a_{\pi_1 1})$ and $a_{\pi_1 1}/(a_{\pi_1 0}+a_{\pi_1 1})$, respectively. Finally, if there is no money to purchase component π_1, set $X_{\pi_1}=0$. We repeat the above procedure for $X_{\pi_2}, X_{\pi_3}, \ldots$, until all components are assigned one of the four types. The algorithm for generating random topology vectors is thus summarized as follows:

Algorithm 1 (Generation Algorithm):

1) Generate a uniform random permutation $\boldsymbol{\pi}=(\pi_1,\ldots,\pi_{n+m})$. Set $k=1$ and $B_0=0$.
2) If $c_{\pi_k}(3)+B_{k-1}\le C_{\max}$, then draw X_{π_k} from $\boldsymbol{a}_{\pi_k}$.
3) Otherwise, if $c_{\pi_k}(2)+B_{k-1}\le C_{\max}$, draw X_{π_k} from $\boldsymbol{a}_{\pi_k}$ truncated to $\{0,1,2\}$.
4) Otherwise, if $c_{\pi_k}(1)+B_{k-1}\le C_{\max}$, draw X_{π_k} from $\boldsymbol{a}_{\pi_k}$ truncated to $\{0,1\}$.
5) Otherwise set $X_{\pi_k}=0$.
6) Terminate if $k=m+n$; otherwise let $B_k=B_{k-1}+c_{\pi_k}(X_{\pi_k})$, set $k=k+1$, and reiterate from step 2.

The idea is now to generate a sequence of sampling matrices with the aim of reaching a "degenerate" matrix A^* — consisting only of zeros and ones — that corresponds to the optimal sampling distribution. This is done via a two-stage procedure.

In the first stage, one determines the $(1-\rho)$-quantile γ_t of the performance (i.e., network reliability) under the previous sampling distribution. The parameter ρ is typically chosen between 0.01 and 0.1. An estimator $\widehat{\gamma}_t$ of γ_t can be obtained by first generating a random sample $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_N$ using the above generation algorithm, computing the performances $r(\boldsymbol{X}_i), i=1,\ldots,N$, and setting $\widehat{\gamma}_t = r_{(\lceil (1-\rho)N \rceil)}$, where $r_{(1)}\le\ldots\le r_{(N)}$ are the order statistics of the performances.

In the second stage, the reference matrix A is updated. More precisely, A is chosen such that the sampling distribution is as close as possible to the theoretically optimal sampling distribution for estimating $\mathbb{P}(r(\boldsymbol{X})\ge\gamma_t)$. In the CE method, the CE distance (also called Kullback-Leibler distance) is used [6] as a measure of proximity between the distributions. The result is that the updating rules are often of a simple form. In particular, the estimated optimal sampling parameters correspond to the *maximum likelihood estimates* of the parameters of the distribution, using only the *elite samples*: those samples for which $r(\boldsymbol{X}_i)\ge\widehat{\gamma}_t$. Denote the set of elite samples at iteration t by $\mathcal{E}_t$, and let $\hat{A}_t=(\hat{a}_{t,jx})$ be the estimated optimal sampling matrix at iteration t. In our case, the updating formula for the sampling parameters is given by

$$\hat{a}_{t,jx}=\frac{\sum_{\boldsymbol{X}_i\in\mathcal{E}_t} I_{\{X_{ij}=x\}}}{|\mathcal{E}_t|}, \tag{6}$$

where X_{ij} is the j-th coordinate of $\boldsymbol{X}_i$. That is, we simply count how many times in the elite sample type x is assigned to component j.

The main CE algorithm for optimizing (4), (5) using the above generation algorithm is summarized as follows.

Algorithm 2 (Main CE Algorithm):

1) Initialize $\widehat{A}_0$. Set $t=1$ (iteration counter).
2) Generate a random sample $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_N$ using Algorithm 1 with $A=\widehat{A}_{t-1}$. Compute the $(1-\rho)$-sample quantile of performances $\widehat{\gamma}_t$ and identify the set $\mathcal{E}_t$ of elite samples.
3) Update $\widehat{A}_t$, using (6).
4) If some stopping criterion is met, then stop; otherwise set $t=t+1$ and reiterate from step 2.

Remark 1 (Smoothed Updating): Instead of updating directly using (6), one may choose to use a *smoothed updating* procedure

$$\widehat{A}_t=\alpha\widetilde{A}_t+(1-\alpha)\widehat{A}_{t-1} \tag{7}$$

where $\widetilde{A}_t$ is obtained via (6) and α is a smoothing parameter. Note that for $\alpha=1$ the original updating procedure is attained. Typically the parameter is set between $0.7\le\alpha<1$. Smoothed updating can help prevent the algorithm from converging too fast to a sub-optimal solution.

IV. Numerical Experiments

To illustrate the performance of the CE algorithm, we present three test cases. In each test case, we also compare the results with those obtained by SA (see the appendix for the implementation details). To make a fair comparison between the algorithms, we run them for the same number of function evaluations.

In each test case the cost and reliability of a node is only dependent on the type, but not on the node itself, i.e., $c_i(x)=c_j(x)$ and $p_i(x)=p_j(x)$, for all $i,j\in\{1,\ldots,n\}$. The same holds for the reliabilities of the links. The cost of the links is the product of the *unit cost* per length, which is the same for all links (depending on the type), and the length of the link (the distance between the corresponding nodes).

Case 1: Network with 5 Nodes and 6 Links

The first test case is concerned with the all-terminal network based on Figure 1. We assume that any two adjacent nodes are some distance away from each other and that each link type has a fixed cost per unit distance. The cost and reliability of component and the distance matrix are given in Tables I and II respectively.

The total budget is equal to $C_{\max}=13000$. For each algorithm we terminate when the number of function evaluations reaches 16000. We used the following CE parameters: $N=800$, $\rho=0.1$, $\alpha=0.7$ and SA parameters: $T=$

TABLE I
Reliability and Cost of Components (Case 1)

	Node		Link	
Type	Reliability	Cost	Reliability	Cost per Unit Length
0	0	0	0	0
1	0.99171	1400	0.9907	8
2	0.99232	1900	0.9927	12
3	0.99658	2550	0.9941	20

TABLE II

DISTANCES BETWEEN NODES (CASE 1)

	1	2	3	4	5
1	-	62	-	34	-
2		-	57	-	25
3			-	-	19
4				-	42
5					-

TABLE III

COMPARISON OF CE AND SA FOR TEST CASE 1, BASED ON 20 INDEPENDENT TRIALS

Algorithm	Success	r_{best}	r_{mean}	r_{worst}
CE	20	0.9737	0.9737	0.9737
SA	9	0.9737	0.9723	0.9696

2, $\beta = 0.9$, $L = 20$. Initially, we set $\widehat{a}_{0,jx} = 0.25$ for $j = 1, \ldots, n+m$ and $x \in \{0, 1, 2, 3\}$.

In this case it is possible to obtain the optimal network topologies via an exhaustive search. The three optimal solutions $\boldsymbol{x}^*$ are listed below:

(3,3,3,2,1,1,1,1,1,2,1)
(2,3,3,3,1,1,1,1,1,2,1)
(1,3,3,3,2,1,1,1,1,2,1)

The optimal network reliability is equal to $r^* = 0.97371$ and the optimal cost is $C(\boldsymbol{x}^*) = 12988$.

Figure 2 shows the performance of the CE method. Each cell represents the probability of assigning a certain type to a corresponding component. For example, the first column in each picture represents the probability of assigning each type to component 1. As the figure shows, the probability matrix $\widehat{A}_t$ quickly converges to an optimal solution out of 4^{11} possible solutions.

Table III shows the results for CE and SA on the first test case over 20 independent replications. Here Success denotes the number of times the algorithm obtained the optimal topology; r_{best} is the best network reliability obtained; r_{worst} is the worst reliability; and r_{mean} is the average reliability of the best solutions throughout the numerical experiments.

The CE performed exceptionally, finding an optimal topology in all 20 replications. SA performed less effectively, obtaining an optimal solutions on 9 occasions. A possible reason is that the function values for different topologies sometimes lie very close together; e.g., less than $\Delta r = 10^{-4}$ apart. This causes a problem with SA since when Δr is very small, the algorithm almost always takes a worse solution as $e^{\Delta r/T}$ is very close to 1.

Both algorithms take around the same CPU time (9 seconds) to output a solution, using a Matlab implementation on a 3.0GHz PC.

Case 2: 6 Nodes and 7 Links

The second case concerns a network design problem with 6 nodes and 7 links given in Figure 3. The setting is the same as [10] where a genetic algorithm (GA) was used to tackle this problem, except that we also allow type 0 to be assigned to a component, which increases the search space.

One advantage of using CE over GA is that it does *not* require the penalty function to solve this problem, as in [10]. At each iteration samples are generated in such way that no infeasible networks are generated. However, there is nothing preventing CE from using penalty functions. In [10], the authors introduce the following penalty function:

$$z(\boldsymbol{x}) = \begin{cases} r(\boldsymbol{x}) & \text{if } C(\boldsymbol{x}) \le C_{\max}, \\ r(\boldsymbol{x}) - \sqrt{\frac{C(\boldsymbol{x}) - C_{\max}}{C_{\max}}} & \text{otherwise,} \end{cases}$$

to penalize infeasible solutions, as the reproduction procedure does not ensure that all new solutions are feasible. They argue that it is important to take infeasible solutions into account as a good solution is often reproduced from feasible and infeasible solutions.

The total budget is $C_{\max} = 14505$. The total number of function evaluations is set to 15000. We take the following CE parameters: $N = 750$, $\rho = 0.1$, $\alpha = 0.7$, and SA parameters: $T = 2$, $\beta = 0.99$, $L = 20$. The optimal network is $\boldsymbol{x}^* = (2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 1, 2, 3)$ and the corresponding reliability and cost are 0.4576 and 14504 respectively. The cost

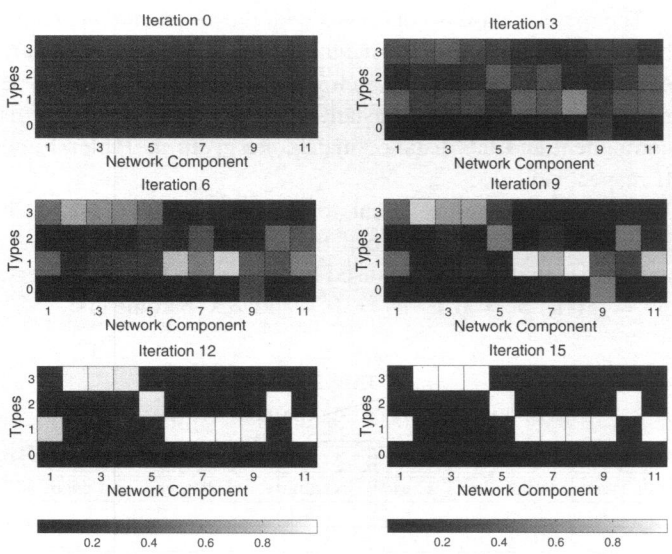

Fig. 2. Convergence of the CE method for test case 1.

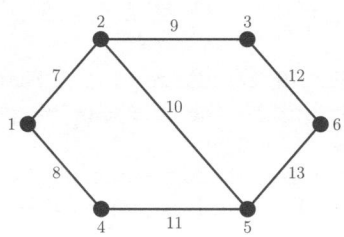

Fig. 3. Network with 6 nodes and 7 links.

and reliability of each component and the distance matrix are given in Tables IV and V respectively.

Table VI shows the results for CE and SA on the second test case over 10 independent replications. We also list the results for the GA algorithm from [10], including the coefficient of variation ϵ (standard deviation divided by mean) which is used [10] to report the variability of the results (based on 10 replications).

Although the search space for this test case ($4^{13} = 67108864$) is around 40 times larger than the case that GA was applied to ($3^{13} = 1594323$), the CE achieved better results than the GA and SA, obtaining an optimal solution in all replications. The worst value obtained by SA was 0.4446.

Case 3: 6 Nodes and 15 Links

Test case 3 deals with a 3-terminal network based on 6-node fully connected graph given in Figure 4. We take the same cost and reliability structure as given in Table I, with the distance matrix given in Table VII.

This problem is a lot harder than the previous two cases, not only because the search space is much larger (4^{21}), but also because there are many near-optimal solutions.

The total budget is $C_{\max} = 22000$ and we take the following CE and SA parameters: $N = 3000$, $\rho = 0.1$, $\alpha = 0.7$, $T = 2$, $\beta = 0.99$, $L = 50$. The maximum number of function evaluations is set to 60000. The optimal topology is equal to

$$\boldsymbol{x}^* = (3,3,3,3,3,3,2,3,2,2,3,2,0,0,2,2,2,3,0,2,3)$$

and the corresponding network reliability and cost are 0.989775 and 21924.

TABLE IV

RELIABILITY AND COST OF COMPONENTS FOR TEST CASE 2

	Node		Link	
Type	Reliability	Cost	Reliability	Cost per Unit Length
1	0.85	1500	0.75	8
2	0.90	1750	0.85	12
3	0.95	2500	0.95	20

TABLE V

DISTANCES BETWEEN NODES (CASE 2)

	1	2	3	4	5	6
1	-	46	-	64	-	-
2		-	39	-	92	-
3			-	-	-	69
4				-	47	-
5					-	35
6						-

TABLE VI

COMPARISON OF CE AND SA FOR TEST CASE 2, BASED ON 10 INDEPENDENT TRIALS

Algorithm	Success	r_{best}	r_{mean}	ϵ
CE	10	0.4576	0.4576	0.0000
SA	2	0.4576	0.4504	0.0137
GA	n/a	0.4576	0.4563	0.0090

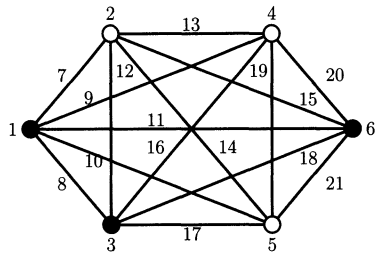

Fig. 4. 6-node fully connected graph with 3 terminal nodes, denoted by black nodes.

TABLE VII

DISTANCES BETWEEN NODES FOR TEST CASE 3

	1	2	3	4	5	6
1	-	62	16	55	56	44
2		-	47	39	51	49
3			-	25	50	25
4				-	39	33
5					-	20
6						-

In 20 trials CE found the optimal solution in every case. However, SA did not find a single optimal solution. Two examples of solutions found by SA are

$$(3,3,3,2,1,3,3,3,3,2,3,2,1,1,2,3,1,3,2,2,2)$$

$$(3,3,3,1,3,3,2,1,2,3,1,3,2,1,2,2,2,2,0,3,1).$$

These have reliabilities very close to the optimal reliability (less than 10^{-6} away).

APPENDIX

Simulated annealing is a well-know generic search and optimization algorithm. There are many different variants, but in its basic form (see e.g. [5]) the algorithm, when applied to our network design problem, involves the following steps:

1) Randomly generate a neighboring topology $\boldsymbol{X}_{\text{new}}$ from a current topology $\boldsymbol{X}_{\text{curr}}$. If $r(\boldsymbol{X}_{\text{new}}) > r(\boldsymbol{X}_{\text{curr}})$, accept $\boldsymbol{X}_{\text{new}}$ as the new topology; otherwise accept $\boldsymbol{X}_{\text{new}}$ only with probability $e^{\Delta r/T}$ where $\Delta r = r(\boldsymbol{X}_{\text{new}}) - r(\boldsymbol{X}_{\text{curr}})$ and T is the temperature.
2) Reduce the temperature using a pre-specified cooling scheme.

There are many possible ways to generate a neighboring topology $\boldsymbol{X}_{\text{new}}$. The simplest one is to select, given the current topology $\boldsymbol{X}_{\text{curr}}$, randomly and uniformly a component $X_{\text{curr},i}$, $i = 1,\ldots,n+m$, and set this component to any of 0,1,2,3, with equal probability. If the new topology does not satisfy the constraint (5), we repeat the above process until a feasible topology is obtained. We repeat this process L times where L is a positive integer before the temperature decrement.

The temperature decrement at each iteration is very important in obtaining the optimal network topology. For example, if the temperature decreases very quickly, the algorithm may be trapped in sub-optimal region. On the other hand, if the temperature decreases very slowly, it will slow down the convergence and as a result will increase a computational time.

There are number of methods to decrease the temperature. A simple method is to use geometric decay:

$$T = \beta \cdot T, \tag{8}$$

where the decay parameter β is a constant close but not equal to 1. For other decrement methods see [18], [19].

ACKNOWLEDGMENTS

We like to thank Dr Kin-Ping Hui of the Australian Defence Science and Technology Organisation for his assistance with the numerical experiments, and Dr Michael Bulmer of the University of Queensland for his valuable comments on simulated annealing.

REFERENCES

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

[2] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, 1996.

[3] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science, Number 4598, 13 May 1983*, vol. 220, 4598, pp. 671–680, 1983.

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

[5] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley, 2003.

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

[7] R. H. Jan, F. J. Hwang, and S. T. Chen, "Topological optimization of a communication network subject to reliability constraint," *IEEE Transactions on Reliability*, vol. 42, pp. 63 – 70, 1993.

[8] B. Dengiz, F. Altiparmak, and A. E. Smith, "Efficient optimization of all-terminal reliable networks, using an evolutionary approach," *IEEE Transactions on Reliability*, vol. 46, no. 1, pp. 18 – 26, March 1997.

[9] D. L. Deeter and A. E. Smith, "Heuristic optimization of network design considering all-terminal reliability," in *Proceedings of the Reliability & Maintainability Symposium*, 1997, pp. 194 – 199.

[10] F. Altiparmak, B. Dengiz, and A. E. Smith, "Reliability optimization of computer communication networks using genetic algorithms," in *IEEE International Conference on Systems, Man., and Cybernetics*, vol. 5, October 1998, pp. 4676 – 4681.

[11] S. Nariai, K.-P. Hui, and D. P. Kroese, "Designing an optimal network using the cross-entropy method," in *IDEAL*, ser. Lecture Notes in Computer Science, M. Gallagher, J. M. Hogan, and F. Maire, Eds., vol. 3578. Springer, 2005, pp. 228–233.

[12] G. Alon, D. P. Kroese, T. Raviv, and R. Y. Rubinstein, "Application of the buffer allocation problem in simulation-based environment," *Annals of Operations Research*, vol. 134, no. 1, pp. 137 – 151, 2005.

[13] R. Y. Rubinstein, "Combinatorial optimisation, cross-entropy, ants and rare events," in *Stochastic Optimization: Algorithms and Applications*, S. Uryasev and P. M. Pardalos, Eds., Kluwer, 2001, pp. 304–358.

[14] K. Chepuri and T. Homem de Mello, "Solving the vehicle routing problem with stochastic demands using the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 153 – 181, 2005.

[15] P. T. de Boer, D. P. Kroese, and R. Y. Rubinstein, "A fast cross-entropy method for estimating buffer overflows in queueing networks," *Management Science*, vol. 50, no. 7, pp. 883 – 895, 2004.

[16] K.-P. Hui, N. Bean, M. Kraetzl, and D. P. Kroese, "The cross-entropy method for network reliability estimation," *Annals of Operations Research*, vol. 134, no. 1, pp. 101–118, 2005.

[17] P. T. de Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, "A tutorial on the cross-entropy method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19 – 67, 2005.

[18] P. J. M. van Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*. D. Reidel Publishing Company, 1987.

[19] M. Lundy and A. Mees, "Convergence of an annealing algorithm," *Math. Prog.*, pp. 111 – 124, 1986.