# On a Convergence of the Cross-Entropy Method

Leonid Margolin

Faculty of Industrial Engineering and Management, Technion, Haifa, Israel
and
Schema Ltd, Herzlia, Israel

E-mail: *leonid_m@schema.com*

**Abstract**

The cross-entropy method is a relatively new method for combinatorial optimization. The idea of the method came from the simulation field and then was successfully applied to different combinatorial optimization problems. In this paper we prove asymptotical convergence of some modifications of the cross-entropy method.

**Keywords:** combinatorial optimization, convergence, cross-entropy.

## 1 Introduction

The *cross-entropy (CE)* method was developed by Rubinstein [9] for solving both continuous multi-extremal and various discrete combinatorial optimization problems. The method was successfully examined on a wide range of problems (for example, the travelling salesman problem [9], the maximal cut problem [12], the bipartition problem [9], the buffer allocation problem [1], image matching [2], image segmentation [2], e.t.c.). This method is an adaptive stochastic procedure based on the importance sampling and Kullback-Leibler cross-entropy. While most of the stochastic algorithms for combinatorial optimization are based on local search (they employ local neighborhood structures), the cross-entropy method is a global random search procedure.

The CE method arises from the rare-event estimation framework [8] and uses similar techniques. The main idea of the CE method can be stated as follows.

First, we randomize our deterministic problem (by defining a probability distribution on the set of all the possible solutions). Next, we define the following event: "the optimal solution appears within sample of certain size (which is very small in comparison with the number of all the feasible solutions)". For the vast majority of the problems this event

is rare, and its estimate is not trivial. For this purpose we make adaptive changes of the probability distribution according to the Kullback-Leibler cross-entropy approach. This procedure significantly increases the chances of the optimal solution (or an almost optimal) to appear within our sample.

In this paper we prove asymptotical convergence for some modifications of the method. The idea and technique of the proof are similar to the ones from the recent paper of Gutjahr [5], where the convergence of two ant algorithms is proved. More precisely, in this paper we define two cross-entropy algorithms, which are very close to the ant algorithms considered by Gutjahr. Due to this similarity, Gutjahr's proof can be readily modified and adopted to our case. For an alternative convergence prove of CE see [6].

The rest of the work is organized as follows. In the second section the general cross-entropy method is described. In the third section two modifications of the CE method are introduced and their convergence is proved.

# 2 The General Cross-Entropy Method

As mentioned above, the initial version of the CE method was presented in [9]. Then, in order to speed up the method and/or improve convergence, several additional modifications were introduced (see [7], [10], [11], [13]): for instance, the conservative (smoothed) modification, the CE method with lower bound on probabilities, the CE method with varying sample size, the CE method with memory, etc. In this section, which is based on [7], we describe the general CE method. For easier understanding, we illustrate the terms and technique with an emphasis to the well-known traveling salesman problem (TSP).

We consider the following combinatorial optimization problem:

$$\min_{\boldsymbol{x}\in\mathcal{X}} L(\boldsymbol{x}), \tag{2.1}$$

where $\mathcal{X} \subset R^n$ is a finite set, $L(\cdot)$ is a real function from $\mathcal{X}$ to $R$.

Assume that $\boldsymbol{x}^* = (\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_n^*)$ is the unique optimal solution of (2.1).

Additionally, assume that each feasible solution of the problem can be defined by the matrix $\boldsymbol{X}$ with components $X_I$, where $I$ belongs to some set of indices $\mathcal{I}$ (that is, $|\mathcal{I}|$ numbers define some feasible solution). Note that $\mathcal{I}$ can be one-dimensional, two-dimensional, etc.

The general cross-entropy method consists of three main steps:

1. Choosing a probability family. Initialization of the distribution parameters and the method parameters. Generating solutions according to the chosen distribution.

2. Updating the distribution parameters, based on the Kullback-Leibler cross-entropy.

3. Checking the stopping rule. Updating the method parameters in the case the stopping rule fails.

At the first step we should choose a "suitable" representation. Finding such "suitable" representation, that is well fitted to the structure of the problem, is of great importance. However, there is no general rule to do so, and our decision is mainly based on intuition.

For example, consider the TSP with $n$ cities on a fully connected graph. As mentioned above, each feasible solution (trajectory) is assumed to be described by the matrix $\boldsymbol{X}$ with components $X_I$, where $I$ belongs to some set of indices $\mathcal{I}$. We would like to present two different representations for this problem.

For instance, we can define trajectories using a matrix $\boldsymbol{X}$, where

$$X_{(r,s)} = \begin{cases} 1, & \text{the transition from } r \text{ to } s \text{ occurred,} \\ 0, & \text{otherwise.} \end{cases}$$

Obviously, $X_{(r,r)} = 0$. In this case $\mathcal{I} = \{(r,s) \in \{1,\dots,n\} \times \{1,\dots,n\} : r \neq s\}$.

With this representation we can generate feasible trajectories randomly, using the following algorithm.

**Algorithm 2.1 Solution Generation for the TSP (for the First Representation):**

*1. Define the probability matrix* $\boldsymbol{P}$*:*

$$\boldsymbol{P} = \begin{bmatrix} 0 & p_{(1,2)} & \dots & p_{(1,n)} \\ p_{(2,1)} & 0 & \dots & p_{(2,n)} \\ \vdots & \vdots & \vdots & \vdots \\ p_{(n,1)} & p_{(n,2)} & \dots & 0 \end{bmatrix} \tag{2.2}$$

*where the probability* $p_{(r,s)}$ *corresponds to the transition from the node* $r$ *to the node* $s$*. Assume that for* $r \neq s$*:* $p_{(r,s)} \neq 0$*.*

*2. Set* $counter = 1$*,* $\boldsymbol{U} = \{1\}$*,* $r = 1$*.*

*3. Generate a next node* $S$ *from the following distribution:*

$$Prob\{S = s\} = \begin{cases} p_{(r,1)}, & s = 1, \\ p_{(r,2)}, & s = 2, \\ p_{(r,3)}, & s = 3, \\ \vdots & \\ p_{(r,n)}, & s = n. \end{cases}$$

*Assume, that* $S$ *received value* $s^*$*. Define*

$$X_{(r,s)} = \begin{cases} 1, & s = s^*, \\ 0, & s \neq s^*. \end{cases}$$

*4. Set* $\boldsymbol{U} = \boldsymbol{U} \cup \{s^*\}$*,* $counter = counter + 1$*,* $r = s^*$*.*

*5. If* $(counter < n - 1)$ *update the* $r$*-th row of the matrix:*

$sum = 0$
*for* $s = 0$ *to* $n$
    *if* $(s \neq r)$ *and* $(s \in \boldsymbol{U})$
        $sum = sum + p_{(r,s)}$
        $p_{(r,s)} = 0$
    *end if*
*end for*
*for* $s = 0$ *to* $n$
    *if* $(s \notin \boldsymbol{U})$
        $p_{(r,s)} = p_{(r,s)}/(1 - sum)$
    *end if*
*end for*

*Go to Step 3.*

*6. Set* $\{s^*\} = \boldsymbol{V} \setminus \boldsymbol{U}$ *(at this step* $\boldsymbol{U}$ *contains exactly* $n-1$ *nodes),* $X_{(r,s^*)} = 1$.

**Remark 2.1** *To accelerate solution generation we can use so-called alias method (see [14]), which is a very effective method for generation random numbers from an arbitrary n-point distribution. More detailed information on using the alias method for the CE algorithms can be found in [7].*

An alternative representation can be stated as follows. Any trajectory can be defined by a matrix $\boldsymbol{X}$, where

$$X_{(r,s)} = \begin{cases} 1, & \text{the node } r \text{ is arranged to the place } s, \\ 0, & \text{otherwise.} \end{cases}$$

In this case $\mathcal{I} = \{(r,s) \in \{1,\dots,n\} \times \{1,\dots,n\}\}$.

Below we present the trajectory generation algorithm corresponding to this representation.

**Algorithm 2.2 Solution Generation for the TSP (for the Second Representation) :**

*1. Define the probability matrix* $\boldsymbol{P}$*:*

$$\boldsymbol{P} = \begin{bmatrix} p_{(1,1)} & p_{(1,2)} & \cdots & p_{(1,n)} \\ p_{(2,1)} & p_{(2,2)} & \cdots & p_{(2,n)} \\ \vdots & \vdots & \vdots & \vdots \\ p_{(n,1)} & p_{(n,2)} & \cdots & p_{(n,n)} \end{bmatrix} \tag{2.3}$$

*where the probability* $p_{(r,s)}$ *corresponds to the arrangement of the node* $r$ *to the place* $s$*. Assume, that all* $p_{(r,s)} \neq 0$.

*2. Set* $\boldsymbol{U} = \emptyset,\ r = 1.$

*3. Generate a value of the random variable* $S$ *from the following distribution:*

$$Prob\{S = s\} = \begin{cases} p_{(r,1)}, & s = 1, \\ p_{(r,2)}, & s = 2, \\ p_{(r,3)}, & s = 3, \\ \vdots & \\ p_{(r,n)}, & s = n. \end{cases}$$

*Assume that* $S$ *received value* $s^*$*. Define*

$$X_{(r,s)} = \begin{cases} 1, & s = s^*, \\ 0, & s \neq s^*. \end{cases}$$

*4. Set* $\boldsymbol{U} = \boldsymbol{U} \cup \{s^*\},\ r = r + 1.$

*5. If* $(r < n)$ *update the* $r$*-th row of the matrix:*

$sum = 0$
*for* $s = 0$ *to* $n$
    *if* $(s \in \boldsymbol{U})$
        $sum = sum + p_{(r,s)}$
        $p_{(r,r)} = 0$
    *end if*
*end for*
*for* $s = 0$ *to* $n$
    *if* $(s \notin \boldsymbol{U})$
        $p_{(r,s)} = p_{(r,s)}/(1 - sum)$
    *end if*
*end for*

*Go to Step 3.*

*6. Set* $\{s^*\} = \boldsymbol{V} \setminus \boldsymbol{U}$ *(at this step* $\boldsymbol{U}$ *contains exactly* $n - 1$ *nodes),* $X_{(n,s^*)} = 1.$

Usually for a specific problem more than one representation can be found. Note that the choice of representation is of great importance for the method convergence.

To proceed with the general CE method, recall that we generate solutions using the random matrix $\boldsymbol{X}$ with components $X_I$, where $I$ belongs to some set of indices $\mathcal{I}$. For each $I \in \mathcal{I}$ assume that $X_I$ can take values from the following set: $\{a_1(I), \ldots, a_{R(I)}(I)\}$.

Let $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{N_t}$ denote solutions participating in the $t$-th iteration (generally, the number of solutions to be considered $(N_t)$ can vary from iteration to iteration). Assume that $\boldsymbol{X}_1, \ldots, \boldsymbol{X}_{K_t}$ are the best solutions obtained in the previous iterations, while $\boldsymbol{X}_{K_t+1}, \ldots, \boldsymbol{X}_{N_t}$ are the solutions generated during the iteration $t$. We will call $K_t$ the "memory size" of the method at the iteration $t$.

Let $\Lambda_t$ be a set of all the solutions obtained during the iterations $1, \ldots, t$ (including $t$).

Let $p_{t,I}(k)$ be a current probability (after the first $t$ iterations) corresponding to the event $\{X_I = a_k(I)\}$, where $I \in \mathcal{I}$, $k \in \{1, \ldots, R(I)\}$.

Let $\phi(L(\boldsymbol{X}), \Lambda_t, \boldsymbol{\nu}_t)$ be a real positive function, non-increasing in the first argument, where $\boldsymbol{\nu}_t = \boldsymbol{\nu}(\Lambda_t)$ is a vector of *parameters of the function* $\phi(\cdot)$ that may be dependent on the previous history (on $\Lambda_t$).

For all $k$ satisfying $p_{t,I}(k) > 0$ define the cross-entropy component:

$$C_{t+1,I}(k) = \frac{\sum\limits_{j\,:\,X_I = a_k(I)} \phi(L(\boldsymbol{X}_j), \Lambda_t, \boldsymbol{\nu}_t)}{\sum\limits_{j} \phi(L(\boldsymbol{X}_j), \Lambda_t, \boldsymbol{\nu}_t)} \ , \tag{2.4}$$

where $X_I$ denotes the $I$-th component of $\boldsymbol{X}_j$ (for simplicity we drop the index $j$).

It should be mentioned that the above formula is based on the Kulback-Liebler cross-entropy (see [7], [10], [11], [13]). Notice that as the combination $X_I = a_k(I)$ appears in relatively "good" solutions more frequently, the value of $C_{t+1,I}(k)$ is greater. Therefore, the purpose of $C_{t+1,I}(k)$ is to reward "good" values of $X_I$.

**Remark 2.2** *When dealing with a maximization problem, $\phi(L(\boldsymbol{X}), \Lambda_t, \boldsymbol{\nu}_t)$ must be a real positive function, non-decreasing in the first argument.*

The general rule for the probability update can be stated as follows. For every $k$ satisfying $p_{t,I}(k) > 0$:

$$p_{t+1,I}(k) = F(C_{t+1,I}(k), p_{0,I}(k), p_{1,I}(k), \ldots, p_{t,I}(k), \boldsymbol{\mu}_t) \ , \tag{2.5}$$

where $F$ is a real function taking values from $[0, 1]$ and non-decreasing in the first argument; $\boldsymbol{\mu}_t = \boldsymbol{\mu}(\Lambda_t)$ is a vector of *parameters of the method* that may be dependent on the previous history. Since $F$ is non-decreasing in the first argument, the "rewarding property" of (2.4) remains valid in the case of (2.5) as well (that is, as the combination $X_I = a_k(I)$ appears in relatively "good" solutions more frequently, the probability $p_{t+1,I}(k)$ associated with $\{X_I = a_k(I)\}$ is greater).

The general cross-entropy algorithm can be written as follows:

**Algorithm 2.3 General CE Algorithm:**

***Step 1***

*a) Choose a suitable representation as described above.*

*Assume that $\mathcal{I}$ denotes a set of indices of the random matrix in this representation.*
*Let $I \in \mathcal{I}$, and assume that the $I$-th component of the random matrix can take values from the following set: $\{a_1(I), \ldots, a_{R(I)}(I)\}$ .*

*b) Define* $p_{0,I}(k)$ $(\forall I \in \mathcal{I},\ \forall k \in \{1,\ldots,R(I)\})$ .

*c) Set* $t = 0,\ \Lambda_{-1} = \emptyset$ .

***Step 2***

*a) Let* $\boldsymbol{X}_1,\ldots,\boldsymbol{X}_{K_t}$ *be the best solutions obtained during the previous iterations. Generate solutions* $\boldsymbol{X}_{K_t+1},\ldots,\boldsymbol{X}_{N_t}$ *according to the probabilities* $\{p_{t,I}(k)\}$ .
*Set* $\Lambda_t = \Lambda_{t-1} \cup \{\boldsymbol{X}_{K_t+1},\ldots,\boldsymbol{X}_{N_t}\}$ .
*Set* $\boldsymbol{\mu}_t = \boldsymbol{\mu}(\Lambda_t),\ \boldsymbol{\nu}_t = \boldsymbol{\nu}(\Lambda_t)$ .

*b) For all* $k$ *satisfying* $p_{t,I}(k) > 0$ *obtain* $C_{t+1,I}(k)$ *from (2.4), and calculate* $p_{t+1,I}(k)$ *according to (2.5).*

***Step 3***

*Set* $t = t + 1$. *Repeat Step 2 until Stopping Rule is satisfied.*

# 3 Convergence of the Cross-Entropy Method

In this section we prove the convergence of a certain class of CE algorithms. As mentioned above, the idea and technique of the proof were taken from [5].

Firstly, we give a definition of so-called construction graph according to [5].

**Definition 3.1** *Let an instance of a combinatorial optimization problem be given. By a construction graph for this instance, we understand a directed graph* $\boldsymbol{C} = (\boldsymbol{V}, \boldsymbol{E})$ *together with a function* $\Phi$ *with the following properties:*

1. *In* $\boldsymbol{C}$*, a unique node is marked as start node.*
2. *Let* $\boldsymbol{W}$ *be the set of directed paths* $w$ *in* $\boldsymbol{C}$ *satisfying the following conditions:*

   *i)* $w$ *starts at the start node of* $\boldsymbol{C}$*;*

   *ii)* $w$ *contains each node of* $\boldsymbol{C}$ *at most once;*

   *iii) the last node on* $w$ *has no successor node in* $\boldsymbol{C}$ *that is not contained in* $w$ *(i.e.,* $w$ *cannot be prolonged without violating (ii)).*

   *Then function* $\Phi$ *maps a subset* $\bar{\boldsymbol{W}}$ *of the set* $\boldsymbol{W}$ *onto the set of feasible solutions of the given problem instance. In other words: to each path* $w$ *in* $\bar{\boldsymbol{W}}$*, there corresponds (via* $\Phi$*) a feasible solution, and to each feasible solution, there corresponds (via* $\Phi^{-1}$*) at least one path in* $\bar{\boldsymbol{W}}$.

The path generation on the construction graph is managed by the following rule. Assuming the current node is $r$, we generate a successor node according to the probabilities

$$Prob\{S=s\} = \begin{cases} \dfrac{p_{t,(r,s)}}{\sum\limits_{(r,u)\ is\ feasible} p_{t,(r,u)}}, & \text{if } (r,s) \text{ is feasible,} \\ \\ 0, & \text{otherwise.} \end{cases} \tag{3.6}$$

Consider the general CE-based algorithm on the construction graph and define the following partial case of it (assume we deal with a minimization problem).

Let $n$ denote the number of vertices in the (construction) graph. For each path $w \in \bar{\boldsymbol{W}}$ we build solution $\boldsymbol{X}$ according to the following rule:

$$X_{(r,s)} = \begin{cases} 1, & \text{the edge } (r,s) \text{ belongs to the tour } w, \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

Add the following artificial components

$$X_{(r,f)} = \begin{cases} 1, & \text{the node } r \text{ doesn't belong to the tour } w, \\ 0, & \text{otherwise,} \end{cases} \tag{3.8}$$

where $f$ does not correspond to any node. (Actually, two the last equations define the function $\Phi$).

Obviously, in our case $\mathcal{I} = \{1,\dots,n\}\times\{1,\dots,n+1\}$ (the $(n+1)-th$ column corresponds to the artificial node $f$).

Let the memory size be $K_t \equiv 1$ (we store only the best solution obtained in all the previous iterations).

Define the parameter of the function $\phi(\cdot)$ by $\boldsymbol{\nu}(\Lambda_t) = \rho_t = 1/N$. Additionally, define

$$\phi(L(\boldsymbol{X}), \Lambda_t, \boldsymbol{\nu}(\Lambda_t)) = I_{\{L(\boldsymbol{X}_j) \le \bar{\gamma}_t\}} \ ,$$

where

$$\bar{\gamma}_t = L_{(\lceil \rho_t N \rceil)} = L_1 \ .$$

Hence,

$$\phi(L(\boldsymbol{X}), \Lambda_t, \boldsymbol{\nu}(\Lambda_t)) = \begin{cases} 1, & \boldsymbol{X} \text{ is one of the current best solutions,} \\ 0, & \text{otherwise.} \end{cases}$$

Obviously,

$$C_{t+1,(r,s)} = C_{t+1,(r,s)}(1) = \frac{\sum\limits_{j=1}^{N} I_{\{L(\boldsymbol{X}_j) \le \bar{\gamma}_t,\, X_{j,(r,s)}=1\}}}{\sum\limits_{j=1}^{N} I_{\{L(\boldsymbol{X}_j) \le \bar{\gamma}_t\}}} =$$

$$= \begin{cases} N^*_{(r,s)}/N^*, & (r,s) \text{ belongs to at least one of the current best solutions,} \\ 0, & \text{otherwise,} \end{cases} \tag{3.9}$$

where $N^*$ is the number of solutions within $\{\boldsymbol{X}_1, \ldots, \boldsymbol{X}_N\}$ having the best obtained value, $N^*_{(r,s)}$ is the number of times the edge $(r,s)$ belongs to one of them.

Define the parameter of the method $\boldsymbol{\mu}_t = \alpha_t$ and let

$$\begin{cases} p_{0,(r,s)} = 1/|\tilde{\boldsymbol{E}}(r)| \ , \\ p_{t+1,(r,s)} = (1-\alpha_t)p_{t,(r,s)} + \alpha_t C_{t+1,(r,s)} \ , \end{cases}$$

where $\tilde{\boldsymbol{E}}(r)$ is the number of arcs outgoing from the vertex $r$ (the artificial arc $(r,f)$ is included).

Formally, we can write the considered algorithm (we will denote this algorithm by Graph-Based CE Algorithm/Conservative Modification - GBCE/CM) in the following way.

**Algorithm 3.1 GBCE/CM:**

***Step 1***

*For each $(r,s)$, edge of the construction graph or artificial edge, set $p_{t+1,(r,s)} = 1/|\tilde{\boldsymbol{E}}(r)|$. Set $t = 0$. Generate an arbitrary tour $\boldsymbol{X}_1$ using the rule (3.6), and use it as the current best found tour.*

***Step 2***

*Generate $N-1$ solutions by generation corresponding vectors $\boldsymbol{X}_j$ $(j = 2, \ldots, N)$ according to the rule (3.6). Add $\boldsymbol{X}_1$, which is the best solution found in the previous iterations.*

*For each $(r,s)$ update the corresponding probability using the following rule:*

$$p_{t+1,(r,s)} = (1-\alpha_t)p_{t,(r,s)} + \alpha_t C_{t+1,(r,s)} \ , \tag{3.10}$$

*where*

$$C_{t+1,(r,s)} = \begin{cases} N^*_{(r,s)}/N^*, & (r,s) \text{ belongs to at least one of the current best solutions,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.11}$$

***Step 3***

*Set $t = t+1$. Choose $\alpha_t$. Repeat Step 2 until Stopping Rule is satisfied.*

In this step we must note that the above algorithm is very close to the algorithm GBAS/tdev defined by Gutjahr ([5]). Therefore, we can analyze it using the same technique.

Similar to Gutjahr, we consider a stochastic process with states $(\boldsymbol{P}_t, \hat{\boldsymbol{X}}_{t-1})$ $(t = 1, 2, \ldots)$, where $\boldsymbol{P}_t$ is the matrix of probabilities at the beginning of the iteration $t$, $\hat{\boldsymbol{X}}_{t-1}$ is the best found solution at the end of the iteration $t-1$. The proof that this process is an inhomogeneous Markov process is analogous to Gutjahr's one.

**Theorem 3.1** *Consider an optimization (minimization) problem on the construction graph and assume that the optimal solution is unique.*

*Let, in the algorithm GBCE/CM,*

$$\alpha_t \leq 1 - \frac{\log(t+1)}{\log(t+2)} \quad (t \geq T) \tag{3.12}$$

*for some* $T \geq 0$, *and*

$$\sum_{t=0}^{\infty} \alpha_t = \infty \ . \tag{3.13}$$

*Then, with probability one, the states* $(\boldsymbol{P}_t, \hat{\boldsymbol{X}}_{t-1})$ *of the assigned Markov process converge as* $t \to \infty$ *to the state* $(\boldsymbol{P}^*, \boldsymbol{X}^*)$, *where* $\boldsymbol{X}^*$ *is the optimal solution, and* $\boldsymbol{P}^*$ *is defined as*

$$\boldsymbol{P}^*_{(r,s)} = \begin{cases} 1, & \text{if } (r,s) \in \boldsymbol{X}^*, \\ 0, & \text{otherwise.} \end{cases} \tag{3.14}$$

*For a fixed repetition, its probability to obtain* $\boldsymbol{X}^*$ *at the iteration* $t$ *tends to* 1, *as* $t \to \infty$.

**Proof**

The proof is similar to the one given at [5] with some minor changes.

We have

$$P\{\boldsymbol{X}^* \text{ is never obtained}\} =$$

$$= \prod_{t=0}^{\infty} P \left\{ \begin{array}{l} \boldsymbol{X}^* \text{ isn't obtained} \\ \text{at the iteration } t \end{array} \ \middle| \ \begin{array}{l} \boldsymbol{X}^* \text{ wasn't obtained} \\ \text{before the iteration } t \end{array} \right\} \ . \tag{3.15}$$

For each fixed arc $(r, s)$, we can calculate the following lower bound on its corresponding probability at the beginning of the iteration $t$. Without lost of generality, let $T \geq 1$. Then, for $t \geq T$

$$p_{t,(r,s)} \geq \left[\prod_{i=0}^{t-1}(1-\alpha_i)\right] p_{0,(r,s)} \geq \left[\prod_{i=0}^{T-1}(1-\alpha_i)\right] \left[\prod_{i=T}^{t-1} \frac{\log(i+1)}{\log(i+2)}\right] p_{0,(r,s)} =$$

$$= \left[\prod_{i=0}^{T-1}(1-\alpha_i)\right] p_{0,(r,s)} \frac{\log(T+1)}{\log(t+1)} = \frac{const}{\log(t+1)} \ . \tag{3.16}$$

Hence, the probability to obtain $\boldsymbol{X}^*$ at a fixed repetition of the iteration $t \geq T$ is greater than

$$\left(\frac{const}{\log(t+1)}\right)^{|\boldsymbol{X}^*|} \ ,$$

where $|\boldsymbol{X}^*|$ is the number of edges in the path corresponding to $\boldsymbol{X}^*$. Notice that this bound is valid for all scenarios of what can happen before the iteration $t$, so we can use it for conditional probabilities as well. Consequently,

$$P \left\{ \begin{array}{l} \boldsymbol{X}^* \text{ isn't obtained} \\ \text{at the iteration } t \end{array} \ \middle| \ \begin{array}{l} \boldsymbol{X}^* \text{ wasn't obtained} \\ \text{before the iteration } t \end{array} \right\} =$$

$$= \prod_{n=1}^{N} P \left\{ \begin{array}{l} \boldsymbol{X}^* \text{ isn't obtained} \\ \text{at the repetition } n \text{ of the iteration } t \end{array} \;\middle|\; \begin{array}{l} \boldsymbol{X}^* \text{ wasn't obtained} \\ \text{before the iteration } t \end{array} \right\} \leq$$

$$\leq \left[ 1 - \left( \frac{const}{\log(t+1)} \right)^{|\boldsymbol{X}^*|} \right]^N . \tag{3.17}$$

Hence, the upper bound on (3.15) can be written as

$$\prod_{t=0}^{T-1} 1 \cdot \prod_{t=T}^{\infty} \left[ 1 - \left( \frac{const}{\log(t+1)} \right)^{|\boldsymbol{X}^*|} \right]^N = \prod_{t=T}^{\infty} \left[ 1 - \left( \frac{const}{\log(t+1)} \right)^{|\boldsymbol{X}^*|} \right]^N . \tag{3.18}$$

Calculating the logarithm of (3.18), and noting that $\sum_t (\log(t+1))^{-a} = \infty$ for each positive integer $a$, we obtain

$$N \sum_{t=T}^{\infty} \log \left[ 1 - \left( \frac{const}{\log(t+1)} \right)^{|\boldsymbol{X}^*|} \right] \leq -N \sum_{t=T}^{\infty} \left( \frac{const}{\log(t+1)} \right)^{|\boldsymbol{X}^*|} = -\infty .$$

Therefore, expression (3.18) equals to zero, and, consequently,

$$P\{\boldsymbol{X}^* \text{ is never obtained}\} = 0 .$$

Hence, the event that in some iteration the optimal solution is reached has probability one.

We have proved that with probability one the considered Markov process $(\boldsymbol{P}_t, \hat{\boldsymbol{X}}_{t-1})$ enters in some iteration into the set $\Theta \times \{\boldsymbol{X}^*\}$, where $\Theta$ is a set of all matrices $\boldsymbol{P}$ of dimension $n \times (n+1)$ with the following properties:

$$0 \leq p_{(r,s)} \leq 1 ,$$

$$\sum_s p_{(r,s)} = 1 . \tag{3.19}$$

(It is not difficult to verify that the last property is valid in all the iterations).

Now we are going to show that the process converges to the state $(\boldsymbol{P}^*, \boldsymbol{X}^*)$.

Notice that when the optimal solution $\boldsymbol{X}^*$ is attained for the first time, formula (3.11) can be rewritten in the following way:

$$C_{t+1,(r,s)} = \begin{cases} 1, & (r,s) \text{ belongs to the optimal solution,} \\ 0, & \text{otherwise.} \end{cases} \tag{3.20}$$

Let $t^*$ is the iteration where the optimal solution was obtained for the first time and consider an edge $(r,s)$, which *doesn't belong* to the path corresponding to the optimal solution. The probability corresponding to this edge at the beginning of iteration $t^* + \Delta t$ is equal to

$$p_{t^*+\Delta t,(r,s)} = \left[ \prod_{t=t^*}^{t^*+\Delta t-1} (1 - \alpha_t) \right] p_{t^*,(r,s)} . \tag{3.21}$$

Using (3.13) it is not difficult to see that

$$\prod_{t=0}^{\infty}(1-\alpha_t)=0$$

(to prove it, take the logarithm of the left-hand side).

Hence,

$$\lim_{\Delta t\to\infty} p_{t^*+\Delta t,(r,s)}=\left[\prod_{t=t^*}^{t^*+\Delta t-1}(1-\alpha_t)\right]p_{t^*,(r,s)}=0 \tag{3.22}$$

(for all the edges $(r,s)$ not belonging to the path corresponding to the optimal solution).

Since the normalization property (3.19) holds and the optimal solution is unique, we have

$$\lim_{\Delta t\to\infty} p_{t^*+\Delta t,(r,s)}=1 \tag{3.23}$$

(for all the edges $(r,s)$ belonging to the path corresponding to the optimal solution).

This proves that the process $(\boldsymbol{P}_t,\hat{\boldsymbol{X}}_{t-1})$ converges to the state $(\boldsymbol{P}^*,\boldsymbol{X}^*)$.

The last assertion of the theorem (for a fixed repetition, its probability to obtain $\boldsymbol{X}^*$ in the iteration $t$ tends to 1, as $t\to\infty$) follows immediately from the formulae (3.22) and (3.22).

□

**Corollary 3.1** *Let, in the algorithm GBCE/CM,*

$$\alpha_t=\frac{c_t}{(t+1)\log(t+2)}\quad(t\geq 0)\qquad with\, 0<\lim_{t\to\infty}c_t<1\ .$$

*Then the assertions of Theorem 3.1 hold.*

The proof can be found in [5].

Now consider another version of the CE method, namely, consider an algorithm, which is similar to Algorithm 3.1, but has one difference: instead of (3.10) the probability updating rule is as follows. Calculate the temporary non-normalized values

$$p^{temp}_{t+1,(r,s)}=\max((1-\alpha)p_{t,(r,s)}+\alpha C_{t+1,(r,s)},p^{min}_{t+1})\ ,$$

where $0<\alpha<1$ is a constant value, and after that, normalize them:

$$p_{t+1,(r,s)}=\frac{p^{temp}_{t+1,(r,s)}}{\sum_{(r,u)isfeasible}p^{temp}_{t+1,(r,u)}}\ .$$

This algorithm will be called the Graph-Based CE Algorithm/Conservative Modification with Lower Bound - GBCE/CMLB. (We have to notice that, due to the above normalization, the actual lower bound on the probabilities at the iteration $t$ is less than $p^{min}_t$). This algorithm is analogous to GBAS/tdlb from [5].

**Theorem 3.2** *Let, in the algorithm GBCE/CMLB,*

$$p_t^{min} = \frac{c_t}{\log(t+1)} \quad (t \geq 1) \qquad with \lim_{t\to\infty} c_t > 0 \ .$$

*Then the assertions of Theorem 3.1 hold.*

**Proof**

The proof is similar to the one of Theorem 3.1. There are two differences only.

In the part proving that with probability one at some iteration the optimal solution is reached, the lower bound for the probabilities at the iteration $t$ is

$$p_{t,(r,s)} \geq \frac{p_t^{min}}{|\tilde{\boldsymbol{E}}(r)|} = \frac{c_t}{|\tilde{\boldsymbol{E}}(r)|\log(t+1)} \geq \frac{c}{2|\tilde{\boldsymbol{E}}(r)|\log(t+1)} = \frac{const}{\log(t+1)}$$

for sufficiently large $t$, where $c = \lim_{t\to\infty} c_t$. This is the the same lower bound as in (3.16).

The proof of convergence of the probabilities to the optimal matrix (3.14) is based on the following argument. Assume the optimal solution was obtained for the first time at the iteration $t^*$. Consider the edge $(r,s)$ not belonging to the optimal solution. It is not difficult to show that for sufficiently large $\Delta t$

$$p_{t+\Delta t,(r,s)} \leq p_{t+\Delta t}^{min} \ .$$

Noting that $\lim_{t\to\infty} p_t^{min} = 0$, we obtain

$$\lim_{\Delta t\to\infty} p_{t+\Delta t,(r,s)} = 0 \ .$$

The remaining part of the proof is analogous to that of Theorem 3.1.

□

# 4 Conclusion

In this work we proved the asymptotical convergence of two cross-entropy algorithms to the optimal solution. This result is analogous to the ones proved for the ant colony optimization and simulated annealing. However, it should be noted that the presented result, as well as the results for two the mentioned above methods, is rather theoretical, because we cannot say anything about convergence rate, as well as about error in each iteration.

# References


[1] Allon, G., Raviv, T. and R.Y. Rubinstein (2001), "Application of the cross entropy method to the buffer allocation problem in simulation based environment".

[2] Dubin, U. (2002), "The cross-entropy method with applications for combinatorial optimization", Master Thesis, Technion, Haifa, Israel.

[3] Gutjahr, W.J. (2000), "A graph-based ant system and its convergence", *Future Generations Computing Systems* **16**, 873–888.

[4] Gutjahr, W.J. (2000), "A generalized convergence result for the graph-based ant system", Technical Report 91-016, Dept. of Statistics and Decision Support Systems, University of Vienna, Austria.

[5] Gutjahr, W.J. (2001), "Ant algorithms with the guaranteed convergence to the optimal solution", Technical Report, Dept. of Statistics and Decision Support Systems, University of Vienna, Austria.

[6] Homem de Mello, T. and R.Y. Rubinstein (2002), "Rare event estimation for static models via cross-entropy and importance sampling", Manuscript, Technion, Haifa, Israel.

[7] Margolin, L. (2002), "Cross-entropy method for combinatorial optimization", Master Thesis, Technion, Haifa, Israel.

[8] Rubinstein, R.Y. (1997), "Optimization of computer simulation models with rare events", *European Journal of Operations Research* **99**, 89–112.

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

[10] Rubinstein, R.Y. (2001), "Combinatorial optimization, cross-entropy, ants and rare events", in: Uryasev, S. and P.M. Pardalos (editors), *Stochastic Optimization: Algorithms and Applications*, Kluwer.

[11] Rubinstein, R.Y. (2001), "Combinatorial optimization via cross-entropy", in: Gass, S. and C. Harris (editors), *Enciclopedia of Operations Research and Management Science*, Kluwer.

[12] Rubinstein, R.Y. (2001), "Cross-entropy and rare-events for maximal cut and bipartition problems", Manuscript, Technion, Haifa, Israel (to appear in IEEE, Transactions on Simulation).

[13] Rubinstein, R.Y. (2002), "Cross-entropy method for combinatorial optimization and rare-events", Lecture Notes, Technion, Haifa, Israel.

[14] Walker, A.J. (1977), "An efficient method for generating discrete random variables with general distributions", ACM Transactions on Mathematical Software (TOMS), v.3, p.253–256.