A Tutorial on the Cross-Entropy Method

Slava Vaisman

The University of Queensland

r.vaisman@uq.edu.au

1. The Cross Entropy Method for Optimization

2. Application: Finding Minimum Label Spanning Trees

Abstract

Obtaining high-quality solutions to hard optimization problems is of crucial importance to machine learning, operational research, and many other engineering problems. To cope with hard problems, we introduce a tutorial on the Cross-Entropy algorithm, which relies on rigorous developments in the fields of information theory and stochastic simulation. Our experience with the Cross Entropy method, indicates that for many problems, it is very reliable and robust as compared to its counterparts, and that this algorithm can obtain optimal or near-optimal solutions while using a reasonable computational effort. Finally, the Cross Entropy method is easy to program and apply for various tasks.

The Cross-Entropy Method

- The Cross-Entropy (CE) method is a sequential procedure which similarly to other evolutionary algorithms, can be used to gradually change the sampling distribution of a random search such that the optimal solution is more likely to occur during the corresponding algorithm execution.
- However, the CE method is distinctive in the sense that it is not directly motivated by a pure evolutionary reasoning.
- Instead, it relies on information theory and stochastic simulation.
- The CE method is very versatile; it can be used for rare-event estimation, discrete, continuous, and even noisy optimization.
- For now, we restrict our attention to the discrete optimization context only. In particular, consider the optimization problem:

$$\min_{\boldsymbol{x} \in \mathcal{X}} S(\boldsymbol{x}),$$

where $S : \mathcal{X} \to \mathbb{R}$ is a fitness function, and $\mathcal{X}^* \subseteq \mathcal{X}$ is the set of optimal solutions.

A general optimization framework

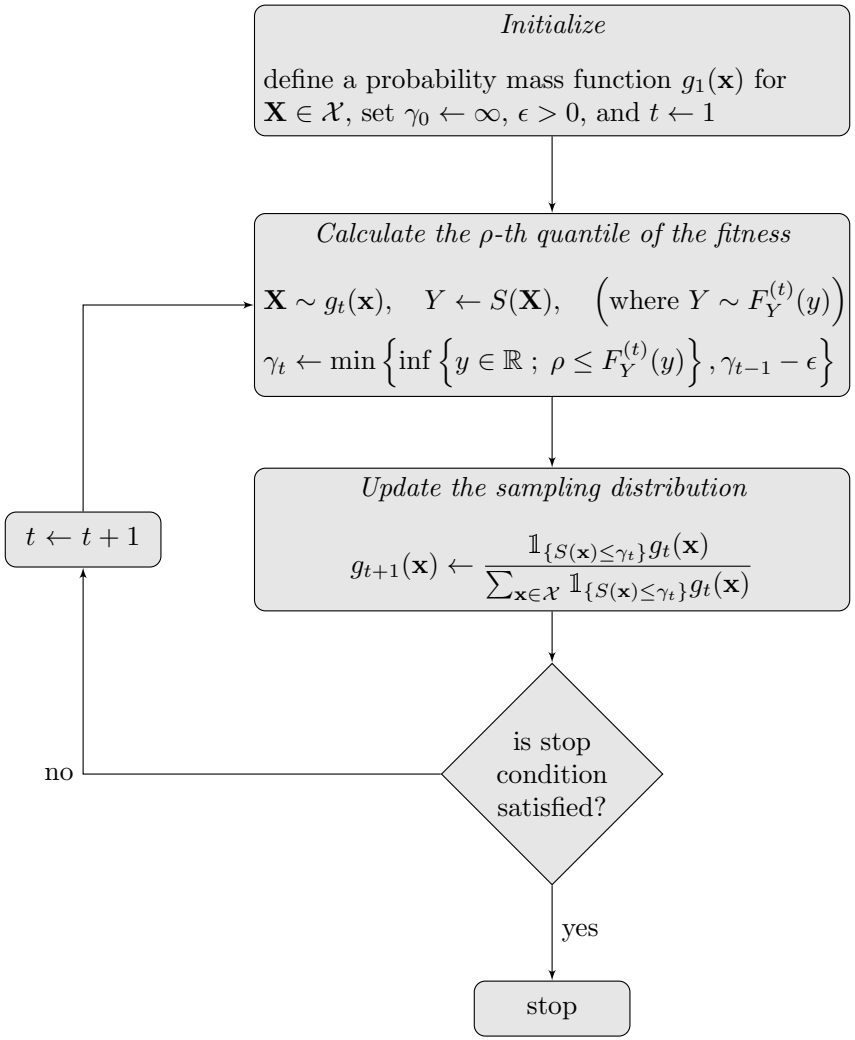
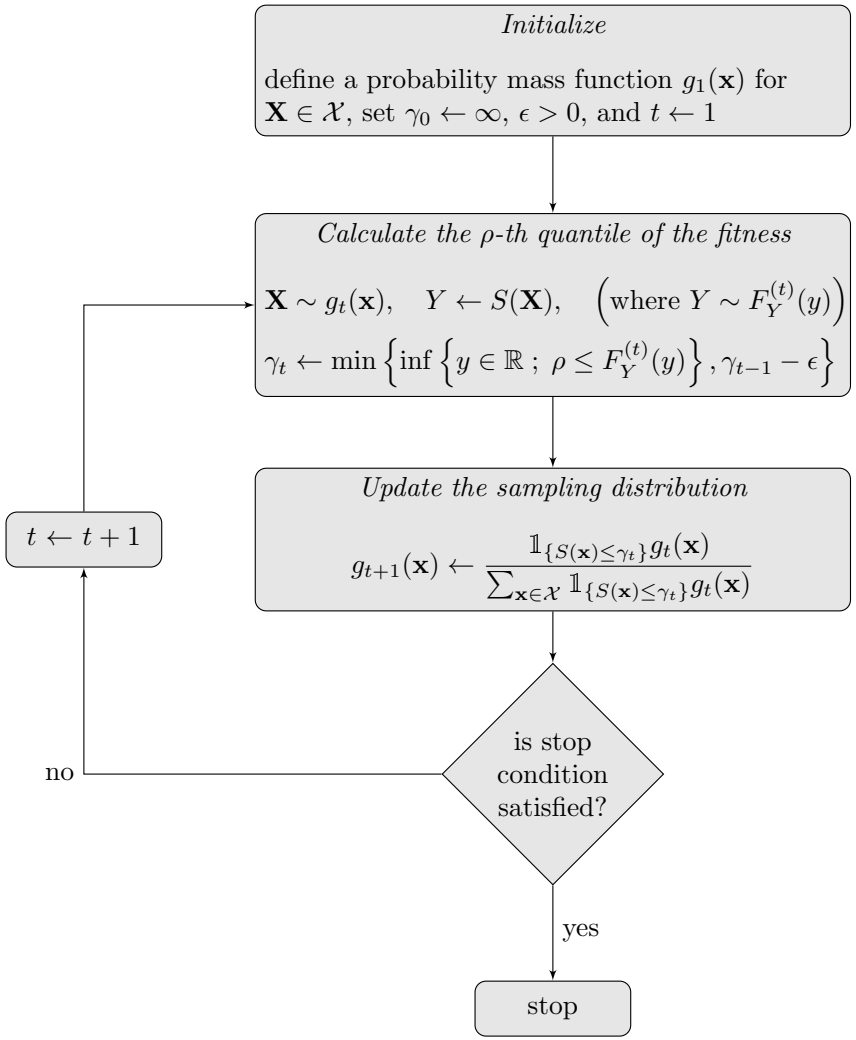


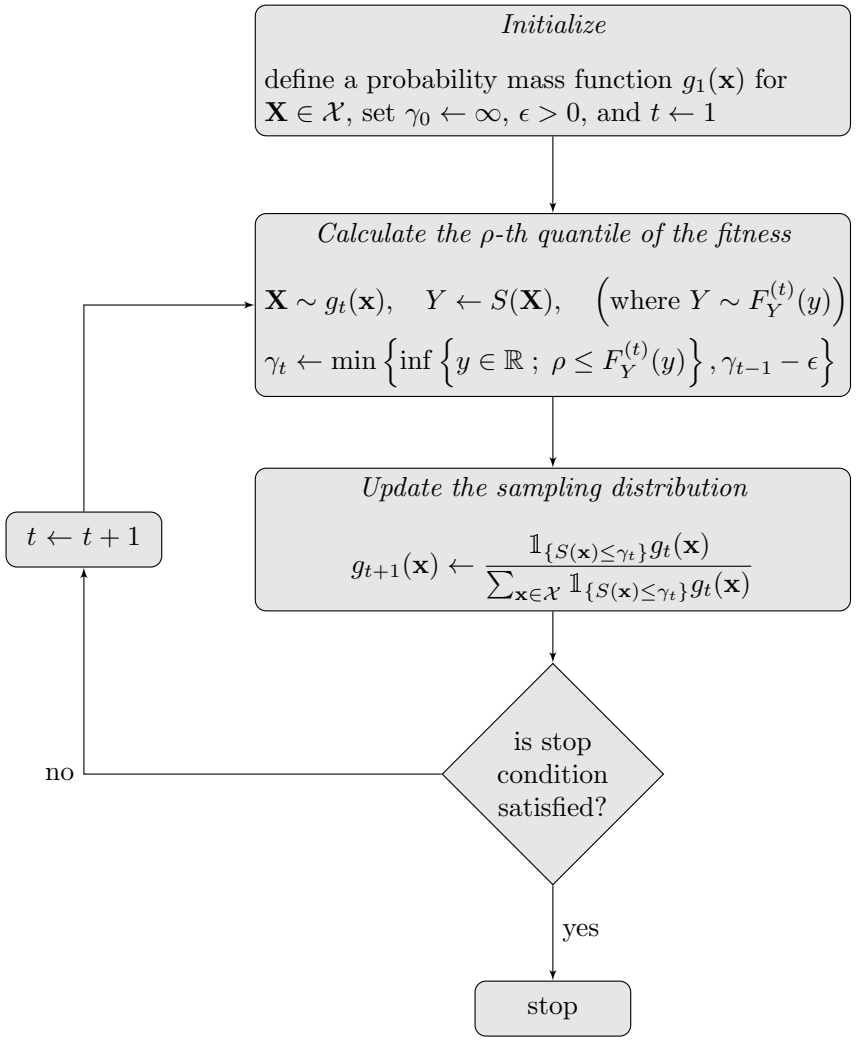
Figure 1: A general discrete optimization framework.

A general optimization framework



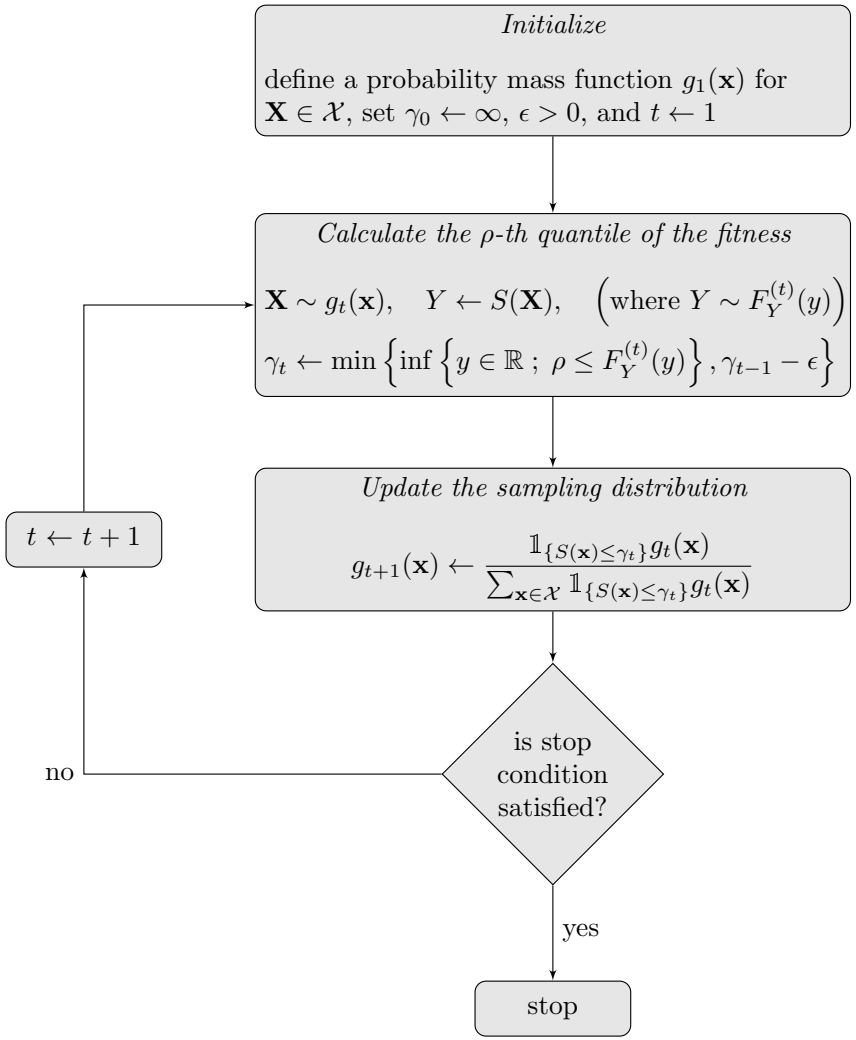
- The general discrete optimization framework begins with the *initialization* step in which a *probability mass function* (pmf) $g_1(\boldsymbol{x})$ for $\boldsymbol{X} \in \mathcal{X}$ is defined.
- Designing such pmf is generally easy, since one can select almost any distribution. A natural choice can be, for example, a uniform distribution on the $\mathcal{X}$ set.

A general optimization framework



- The following task is to *calculate the ρ-th quantile of the fitness* $Y = S(\boldsymbol{X})$, where $\boldsymbol{X} \sim g_t(\boldsymbol{x})$.
- We assume that the *cumulative distribution function* (cdf) of Y is $F_Y^{(t)}(y)$, where t is the current iteration counter.
- As soon as the ρ-th quantile is available, one can *update the sampling distribution*.

A general optimization framework



- This updated distribution will be used in the consecutive iteration, provided that the stop condition is not satisfied.
- Finally, the procedure terminates when some predefined stopping criterion is met.
- For example, one might stop if for all $\boldsymbol{x} \in \mathcal{X}$, it holds that $1_{\{S(\boldsymbol{x}) \le \gamma_t\}} = 0$.

A general (discrete) optimization framework

- Note that for each iteration t, the fitness of $\boldsymbol{X} \sim g_t(\boldsymbol{x})$ satisfies $S(\boldsymbol{X}) \leq \gamma_{t-1}$.
- In addition, since $\epsilon > 0$, this is not very hard to see that the sequence $\gamma_0, \gamma_1, \ldots,$ is strictly decreasing, since $\gamma_t \leq \gamma_{t-1} - \epsilon$ for all t.
- Moreover, upon the termination, the procedure will find a solution $\boldsymbol{x} \in \mathcal{X}$ that is at most ϵ far away from an optimal solution $\boldsymbol{x}^* \in \mathcal{X}^*$, that is, for such $\boldsymbol{x}$, it holds that $S(\boldsymbol{x}) \leq S(\boldsymbol{x}^*) + \epsilon$.
- Finally, if the fitness function has a discrete range, namely, if $S : \mathcal{X} \to \mathbb{N}$, than for any $\epsilon \in (0, 1)$, the procedure will find an optimal solution that satisfies $\boldsymbol{x}^* \in \mathcal{X}^*$.

Some problems

- In practice, however, there are two major problems with the above methodology.
- First, for almost all practical situations $\mathcal{X}$ is large and therefore, finding an exact ρ-th quantile is computationally infeasible.
- The second problem is the hardness of sampling from $g_t(\boldsymbol{x})$ for all $t > 1$;

$$g_t(\boldsymbol{x}) = \frac{1_{\{S(\boldsymbol{x}) \leq \gamma_{t-1}\}} g_{t-1}(\boldsymbol{x})}{\sum_{\boldsymbol{x} \in \mathcal{X}} 1_{\{S(\boldsymbol{x}) \leq \gamma_{t-1}\}} g_{t-1}(\boldsymbol{x})}.$$

  Note that the normalization constant of $g_t(\boldsymbol{x})$ is generally not available analytically.
- While the first issue can be resolved via simulation, that is, by finding a *sample fitness quantile* $\hat{\gamma}_t$ from $S(\boldsymbol{X}_1), \ldots, S(\boldsymbol{X}_N)$, where $X_i \sim g_t(\boldsymbol{x})$ for $1 \leq j \leq N$, the sampling from $g_t(\boldsymbol{x})$ remains hard.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$

- Define $\mathcal{X} = \{0,1\}^k$, $\boldsymbol{x} = (x_1, \ldots, x_k) \in \mathcal{X}$.
- Let $S(x)$ be the function to be minimized.
- We proceed with the definition of the probability distribution of the random variable $\boldsymbol{X} = (X_1, \ldots, X_k)$, where $X_i \in \{0,1\}$ for $i = 1, \ldots, k$.
- Since the sampling from the pmf defined in the *Update the sampling distribution* phase; namely, sampling from

$$g_{t+1}(\boldsymbol{x}) = \frac{1_{\{S(\boldsymbol{x}) \leq \gamma_t\}} g_t(\boldsymbol{x})}{\sum_{\boldsymbol{x} \in \mathcal{X}} 1_{\{S(\boldsymbol{x}) \leq \gamma_t\}} g_t(\boldsymbol{x})}, \tag{1}$$

is generally hard, we propose to approximate (1) using a parametric family

$$f(\boldsymbol{x}; \boldsymbol{p}_{t+1}) = \prod_{i=1}^{k} p_{t+1,i}^{x_i} (1 - p_{t+1,i})^{1-x_i},$$

where $\boldsymbol{p}_{t+1} = (p_{t+1,1}, \ldots, p_{t+1,k})$, and $\boldsymbol{p}_{t+1} \in [0,1]^k$.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$

- The parametric family

$$f(\boldsymbol{x}; \boldsymbol{p}_{t+1}) = \prod_{i=1}^{k} p_{t+1,i}^{x_i} (1 - p_{t+1,i})^{1-x_i}, \tag{2}$$

  is a joint pmf of k independent Bernoulli random variables.
- That is, one can sample $\boldsymbol{X} = (X_1, \ldots, X_k)$ component-wise, and independently for each X_i for $1 \leq i \leq k$; namely, $X_i \sim \mathrm{Ber}(p_{t+1,i})$.
- It is important to note that the sampling from (2) is easy as compared to the corresponding sampling from g_t.
- To summarize, in order to approximate the sequence of pmfs $\{g_t\}$ for all $t \in \mathbb{N} \setminus \{0\}$, we wish to obtain the corresponding parameter vectors $\{\boldsymbol{p}_t\}$, and use (2) instead.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Calculating the sample quantile

- The calculation of the sample fitness quantile $\hat{\gamma}_t$ is trivial, provided that $\boldsymbol{p}_t$ is readily available.
- Specifically, it is sufficient to sample $\boldsymbol{X}_j \sim f(\boldsymbol{x}; \boldsymbol{p}_t)$ for $1 \leq j \leq N$, and sort the $\{S(\boldsymbol{X}_j)\}_{j=1}^N$ set in the ascending order.
- Denote such an ordering by $S_{(1)} \leq \cdots \leq S_{(N)}$.
- Then, $\hat{\gamma}_t \leftarrow S_{(\lceil N \times \rho \rceil)}$ is the desired sample fitness (ρ-th) quantile.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

- Our final objective is to approximate the (optimal) sampling pmf

$$g_{t+1}(\boldsymbol{x}) = \frac{1_{\{S(\boldsymbol{x}) \leq \gamma_t\}} g_t(\boldsymbol{x})}{\sum_{\boldsymbol{x} \in \mathcal{X}} 1_{\{S(\boldsymbol{x}) \leq \gamma_t\}} g_t(\boldsymbol{x})},$$

  via $f(\boldsymbol{x}; \boldsymbol{p}_{t+1})$, where $f(\boldsymbol{x}; \boldsymbol{p}_{t+1})$ belongs to the parametric family

$$f(\boldsymbol{x}; \boldsymbol{p}_{t+1}) = \prod_{i=1}^{k} p_{t+1,i}^{x_i} (1 - p_{t+1,i})^{1-x_i},$$

  that is, to find $\boldsymbol{p}_{t+1}$.
- The latter will be accomplished via a minimization of the relative entropy (also denoted by Kullback-Leibler divergence) of $f(\boldsymbol{x}; \boldsymbol{p}_{t+1})$ with respect to $g_{t+1}(\boldsymbol{x})$.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

The formal characterization of the relative entropy concept is provided in the following Definition.

Definition (Relative entropy)

The relative entropy of a pmf $f(\cdot)$ with respect to a pmf $g(\cdot)$ is given by:

$$\mathcal{D}(g,f) = \mathbb{E}_g \ln\left(\frac{g(\boldsymbol{X})}{f(\boldsymbol{X})}\right) = \sum_{\boldsymbol{x}} \ln\left(\frac{g(\boldsymbol{x})}{f(\boldsymbol{x})}\right) g(\boldsymbol{x})$$
$$= \sum_{\boldsymbol{x}} g(\boldsymbol{x}) \ln g(\boldsymbol{x}) - \sum_{\boldsymbol{x}} g(\boldsymbol{x}) \ln f(\boldsymbol{x}).$$

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

- It is convenient to think about the relative entropy as a *distance measure* between these two pmfs.
- While the relative entropy is not a regular distance measure in the sense that $\mathcal{D}(g, f)$ is generally not equal to $\mathcal{D}(f, g)$, it is possible to show that $\mathcal{D}(g, f) \geq 0$ and that the equality occurs if $g = f$.
- Under our setting, we wish to find $\boldsymbol{p}_{t+1}$ such that $\mathcal{D}(g_{t+1}(\boldsymbol{x}), f(\boldsymbol{x}, \boldsymbol{p}_{t+1}))$ is minimized. It holds that

$$\min_{\boldsymbol{p}_{t+1}} \mathcal{D}(g_{t+1}(\boldsymbol{x}), f(\boldsymbol{x}, \boldsymbol{p}_{t+1}) = \min_{\boldsymbol{p}_{t+1}} \Bigg(\sum_{\boldsymbol{x}} g_{t+1}(\boldsymbol{x}) \ln g_{t+1}(\boldsymbol{x}) \underbrace{- \sum_{\boldsymbol{x}} g_{t+1}(\boldsymbol{x}) \ln f(\boldsymbol{x}; \boldsymbol{p}_{t+1})}_{(*)} \Bigg)$$

- Note that the optimization problem is with respect to the $\boldsymbol{p}_{t+1}$ parameter. Thus, the *minimization* problem is equivalent to a *maximization* problem of the second term $(*)$ with respect to $\boldsymbol{p}_{t+1}$.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

Suppose that $\boldsymbol{p}_{t+1} = (p_{t+1,1}, \ldots, p_{t+1,k})$ and that $\boldsymbol{x} = (x_1, \ldots, x_k)$. Then, using the definitions of $g_{t+1}(\boldsymbol{x})$ and $f(\boldsymbol{x}; \boldsymbol{p}_{t+1})$, the corresponding maximization problem $(*)$, can be written in the form:

$$\begin{aligned}
\max_{\boldsymbol{p}_{t+1}} \sum_{\boldsymbol{x}} g_{t+1}(\boldsymbol{x}) \ln f(\boldsymbol{x}; \boldsymbol{p}_{t+1}) &= \max_{\boldsymbol{p}_{t+1}} \sum_{\boldsymbol{x}} \frac{1_{\{S(\boldsymbol{x}) \le \gamma_t\}} f(\boldsymbol{x}; \boldsymbol{p}_t)}{\sum_{\boldsymbol{x} \in \mathcal{X}} 1_{\{S(\boldsymbol{x}) \le \gamma_t\}} f(\boldsymbol{x}; \boldsymbol{p}_t)} \times \\
&\quad \times \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{x_i} (1 - p_{t+1,i})^{1-x_i} \right) \\
&= \max_{\boldsymbol{p}_{t+1}} \sum_{\boldsymbol{x}} 1_{\{S(\boldsymbol{x}) \le \gamma_t\}} f(\boldsymbol{x}; \boldsymbol{p}_t) \times \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{x_i} (1 - p_{t+1,i})^{1-x_i} \right) \\
&= \max_{\boldsymbol{p}_{t+1}} \mathbb{E}_{f(\boldsymbol{x}; \boldsymbol{p}_t)} 1_{\{S(\boldsymbol{X}) \le \gamma_t\}} \times \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{X_i} (1 - p_{t+1,i})^{1-X_i} \right),
\end{aligned}$$

where the second equality follows from the fact that the denominator $\sum_{\boldsymbol{x} \in \mathcal{X}} 1_{\{S(\boldsymbol{x}) \le \gamma_t\}} f(\boldsymbol{x}; \boldsymbol{p}_t)$ is both constant and does not depend on the optimization parameter $\boldsymbol{p}_{t+1}$. Thus, this denominator does not affect the optimization problem.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

- The exact evaluation of the expected value

$$\mathbb{E}_{f(\boldsymbol{x};\boldsymbol{p}_t)} 1_{\{S(\boldsymbol{X}) \le \gamma_t\}} \times \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{X_i} (1 - p_{t+1,i})^{1-X_i} \right)$$

  is generally not feasible, however, it can be approximated via sampling from the $f(\boldsymbol{x}; \boldsymbol{p}_t)$ pmf.
- In particular, we can work with the so-called *stochastic counterpart*. Namely, the solution of

$$\max_{\boldsymbol{p}_{t+1}} \mathbb{E}_{f(\boldsymbol{x};\boldsymbol{p}_t)} 1_{\{S(\boldsymbol{X}) \le \gamma_t\}} \times \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{X_i} (1 - p_{t+1,i})^{1-X_i} \right)$$

  can be approximated by:

$$\max_{\boldsymbol{p}_{t+1}} \frac{1}{N} \sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j) \le \gamma_t\}} \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{X_{j,i}} (1 - p_{t+1,i})^{1-X_{j,i}} \right),$$

  where $\boldsymbol{X}_j = (X_{j,1}, \ldots, X_{j,k}) \sim f(\boldsymbol{x}; \boldsymbol{p}_t)$ for $j = 1, \ldots, N$.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

Lemma

The function

$$\max_{\boldsymbol{p}_{t+1}} \frac{1}{N} \sum_{j=1}^{N} 1_{\{S(\boldsymbol{x}_j) \leq \gamma_t\}} \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{x_{j,i}} (1 - p_{t+1,i})^{1 - x_{j,i}} \right),$$

is concave and differentiable with respect to $\boldsymbol{p}_{t+1}$.

- The functions $\ln(p_i)$ and $\ln(1 - p_i)$ are concave for $1 \leq i \leq k$. To see this, note that:

$$\frac{\partial^2}{\partial p_i^2} \ln(p_i) = -\frac{1}{p_i^2}, \quad \frac{\partial^2}{\partial p_i^2} \ln(1 - p_i) = -\frac{1}{(1 - p_i)^2},$$

  for $1 \leq i \leq k$. In addition, $1_{\{S(\boldsymbol{x}_j) \leq \gamma\}}$ is non-negative and does not depend on $\boldsymbol{p}$ for $1 \leq j \leq N$.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

- Finally, $(1 - x_{j,i})$ and $x_{j,i}$ are non-negative, that is,

$$\frac{1}{N}\sum_{j=1}^{N} 1_{\{S(\boldsymbol{x}_j)\leq\gamma\}} \ln\left(\prod_{i=1}^{k} p_i^{x_{j,i}}(1-p_i)^{1-x_{j,i}}\right)$$
$$= \frac{1}{N}\sum_{j=1}^{N} 1_{\{S(\boldsymbol{x}_j)\leq\gamma\}} \left(\sum_{i=1}^{k} x_{j,i}\ln p_i + (1-x_{j,i})\ln(1-p_i)\right),$$

  is a non-negative weighted sum of concave functions ($\ln(p_i)$ and $\ln(1-p_i)$).
- Therefore,

$$\max_{\boldsymbol{p}_{t+1}} \frac{1}{N}\sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j)\leq\gamma_t\}} \ln\left(\prod_{i=1}^{k} p_{t+1,i}^{X_{j,i}}(1-p_{t+1,i})^{1-X_{j,i}}\right),$$

  is concave and differentiable with respect to $\boldsymbol{p}$.

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

Since the function

$$\max_{\boldsymbol{p}_{t+1}} \frac{1}{N} \sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j) \leq \gamma_t\}} \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{X_{j,i}} (1 - p_{t+1,i})^{1 - X_{j,i}} \right), \tag{3}$$

is concave and differentiable with respect to $\boldsymbol{p}_{t+1}$, the optimal parameter $\boldsymbol{p}^*_{t+1} = (p^*_{t+1,1}, \ldots, p^*_{t+1,k})$ which maximizes (3), can be obtained by solving:

$$\frac{1}{N} \sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j) \leq \gamma_t\}} \nabla \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{X_{j,i}} (1 - p_{t+1,i})^{1 - X_{j,i}} \right) = 0,$$

Moreover, it holds that

$$p^*_{t+1,i} = \frac{\sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j) \leq \gamma_t\}} X_{j,i}}{\sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j) \leq \gamma_t\}}} \quad \forall\, 1 \leq i \leq k.$$

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

Lemma

Let $\boldsymbol{p} = (p_1, \ldots, p_k)$, *such that* $\boldsymbol{p} \in [0,1]^k$ *and let* $\boldsymbol{x}_j = (x_{j,1}, \ldots, x_{j,k}) \in \{0,1\}^k$ *for* $1 \leq j \leq N$. *Suppose that* $S : \{0,1\}^k \to \mathbb{R}$, $\gamma \in \mathbb{R}$, *and* $N \in \mathbb{N}$. *Then, the function*

$$w(p_1, \ldots, p_k) = \frac{1}{N} \sum_{j=1}^{N} 1_{\{S(\boldsymbol{x}_j) \leq \gamma\}} \ln \left(\prod_{i=1}^{k} p_i^{x_{j,i}} (1 - p_i)^{1 - x_{j,i}} \right)$$

is maximized for

$$p_i^* = \frac{\sum_{j=1}^{N} 1_{\{S(\boldsymbol{x}_j) \leq \gamma\}} x_{j,i}}{\sum_{j=1}^{N} 1_{\{S(\boldsymbol{x}_j) \leq \gamma\}}} \quad \forall\, 1 \leq i \leq k.$$

Please note that this is a closed-form solution!

Developing the CE algorithm for $\mathcal{X} = \{0,1\}^k$ — Approximating the sampling pmf

We saw that w is concave and differentiable with respect to $p_1, \ldots, p_k$. Thus, it is sufficient to solve

$$\nabla w = \frac{1}{N} \sum_{j=1}^{N} 1_{\{S(\mathbf{x}_j) \le \gamma\}} \nabla \ln \left(\prod_{i=1}^{k} p_i^{x_i} (1-p_i)^{1-x_i} \right) = 0.$$

It is not very hard to see that for all $1 \le i \le k$, it holds that:

$$\begin{aligned}
\frac{\partial}{\partial p_i} w &= \frac{1}{N} \sum_{j=1}^{N} 1_{\{S(\mathbf{x}_j) \le \gamma\}} \left(\frac{x_{j,i}}{p_i} - \frac{(1-x_{j,i})}{1-p_i} \right) = 0 \\
\Rightarrow \quad & \sum_{j=1}^{N} 1_{\{S(\mathbf{x}_j) \le \gamma\}} \left(\frac{x_{j,i}(1-p_i) - p_i(1-x_{j,i})}{p_i(1-p_i)} \right) = 0 \\
\Rightarrow \quad & \sum_{j=1}^{N} 1_{\{S(\mathbf{x}_j) \le \gamma\}} \left(\frac{x_{j,i} - p_i}{p_i(1-p_i)} \right) = 0 \quad \Rightarrow \quad \sum_{j=1}^{N} 1_{\{S(\mathbf{x}_j) \le \gamma\}} x_{j,i} - \sum_{j=1}^{N} 1_{\{S(\mathbf{x}_j) \le \gamma\}} p_i = 0 \\
\Rightarrow \quad & p_i^* = \frac{\sum_{j=1}^{N} 1_{\{S(\mathbf{x}_j) \le \gamma\}} x_{j,i}}{\sum_{j=1}^{N} 1_{\{S(\mathbf{x}_j) \le \gamma\}}}.
\end{aligned}$$

The CE algorithm for $\mathcal{X} = \{0,1\}^k$

Input: A function $S(\boldsymbol{x})$, a sample size $N \in \mathbb{N}$, a smoothing parameter $\alpha \in (0,1)$, and a rarity parameter $\rho \in (0,1)$.

1. Set $t \leftarrow 1$ and $\boldsymbol{p}_t \leftarrow (p_1, \ldots, p_k)$, such that $p_i = 0.5$ for $1 \leq i \leq k$.
2. While termination criterion is not fulfilled **do**:
   1. Sample $\boldsymbol{X}_j \sim f(\boldsymbol{x}; \boldsymbol{p}_t)$, and calculate $S(\boldsymbol{X}_j)$ for $1 \leq j \leq N$.
   2. Let $S_{(1)} \leq \cdots \leq S_{(N)}$ be the elements of the $\{S(\boldsymbol{X}_j)\}_{j=1}^N$ set sorted in an ascending order.
   3. $\hat{\gamma}_t \leftarrow S_{(\lceil N \times \rho \rceil)}$ `/* Find the (sample)` ρ`-quantile */`
   4. Find $\boldsymbol{p}_{t+1}$

$$\tilde{p}_{t+1,i} \leftarrow \frac{\sum_{j=1}^N 1_{\{S(\boldsymbol{X}_j) \leq \hat{\gamma}_t\}} X_{j,i}}{\sum_{j=1}^N 1_{\{S(\boldsymbol{X}_j) \leq \hat{\gamma}_t\}}}, \quad \forall 1 \leq i \leq k$$

   `/* note that` $X_{j,i}$ `is the` i`-th component of` $\boldsymbol{X}_j$ `*/`

   5. $\tilde{\boldsymbol{p}}_{t+1} \leftarrow (\tilde{p}_{t+1,1}, \ldots, \tilde{p}_{t+1,k})$
   6. $\boldsymbol{p}_{t+1} \leftarrow (1-\alpha)\, \boldsymbol{p}_t + \alpha\, \tilde{\boldsymbol{p}}_{t+1}$ (smooth)
   7. $t \leftarrow t+1$
3. Return $\boldsymbol{X} = (X_1, \ldots, X_k) \sim f(\boldsymbol{x}; \boldsymbol{p}_t)$.

CE for continuous optimization problems

- The CE algorithm can be almost immediately applied for continuous optimization problems.
- For binary vectors ($\mathcal{X} = \{0,1\}^k$), we used the pmf:

$$f(\boldsymbol{x}; \boldsymbol{p}_{t+1}) = \prod_{i=1}^{k} p_{t+1,i}^{x_i} (1 - p_{t+1,i})^{1-x_i}.$$

- In the $\mathcal{X} = \mathbb{R}^k$ case, one can define (for example),

$$f(\boldsymbol{x}; \boldsymbol{\mu}_{t+1}, \boldsymbol{\sigma}_{t+1}) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{1}{2}\left(\frac{x-\mu_i}{\sigma_i}\right)^2}, \tag{4}$$

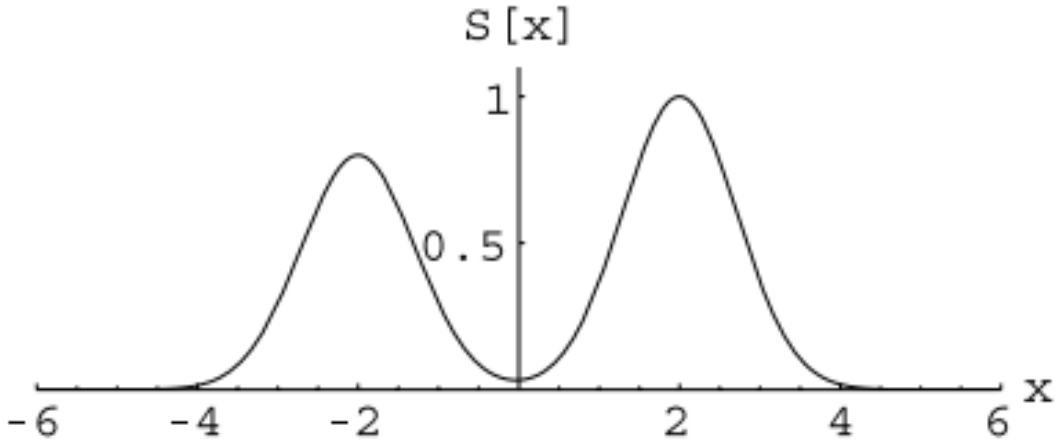
  where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_k)$ and $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_k)$.
- Similar to the binary case, one can solve the entropy minimization problem and obtain (for (4)), the following closed form solution:

$$\tilde{\mu}_i = \frac{\sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j) \le \hat{\gamma}_t\}} X_{j,i}}{\sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j) \le \hat{\gamma}_t\}}}, \quad \tilde{\sigma}_i^2 = \frac{\sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j) \le \hat{\gamma}_t\}} (X_{j,i} - \tilde{\mu}_i)}{\sum_{j=1}^{N} 1_{\{S(\boldsymbol{X}_j) \le \hat{\gamma}_t\}}}, \quad \forall 1 \le i \le k.$$

CE example

- We consider an example from The Cross-Entropy Method for Continuous Multi-Extremal Optimization, by Dirk P. Kroese, Sergey Porotsky, and Reuven Y. Rubinstein.
- The authors applied the CE method to several continuous optimization problems including the *clustering problem*.
- A small but illustrative example is as follows:

$$S(x) = e^{-(x-2)^2} + 0.8e^{-(x+2)^2}, \quad x \in \mathbb{R}.$$



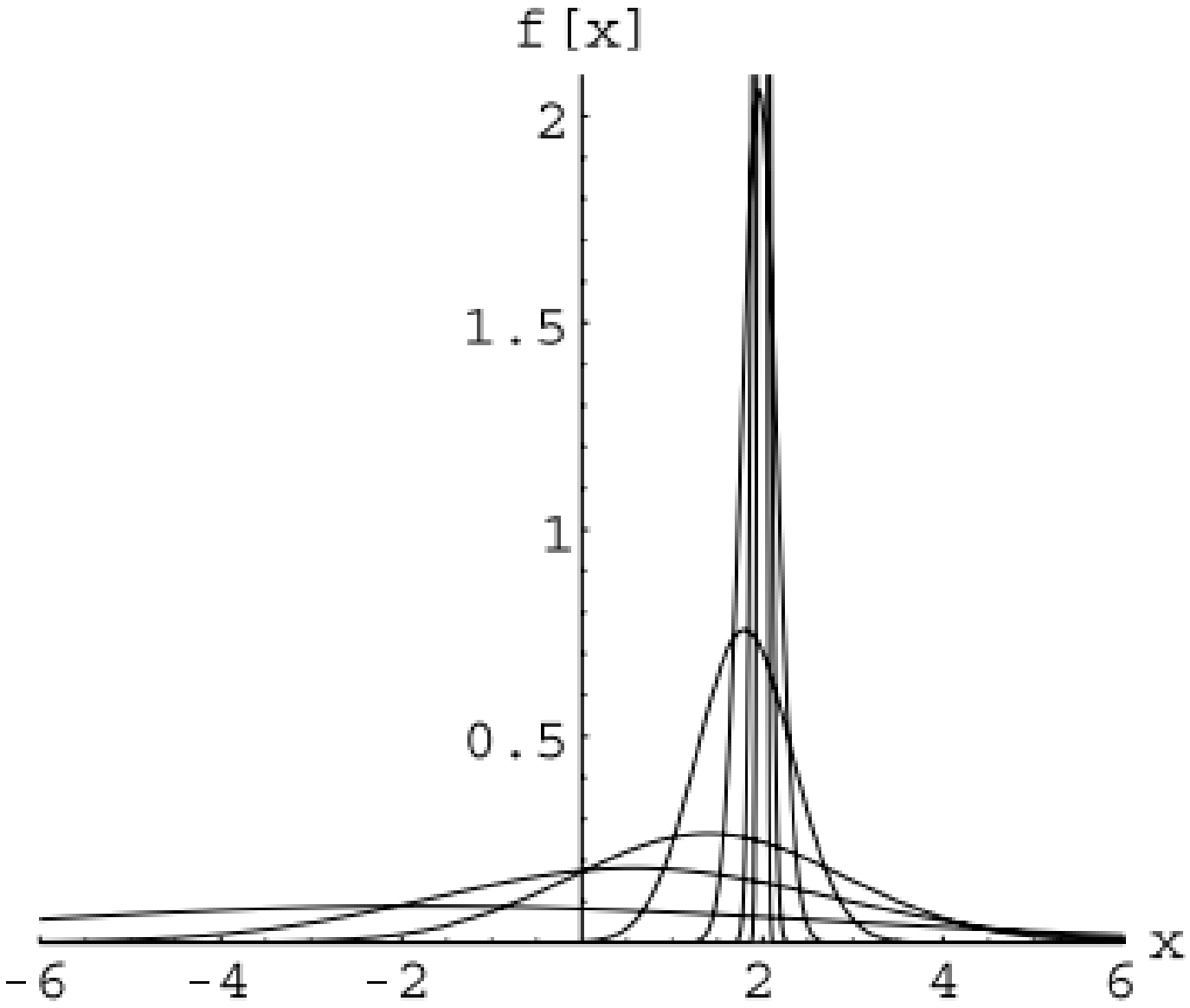
Complete implementation of the CE method for continuous optimization

```
S = inline('exp(-(x-2).^2) + 0.8*exp(-(x+2).^2)');
mu = -6;
sigma = 10;
Nel = 10;
N = 100;
eps = 1E-8;
t=0;
while sigma > eps
    t = t+1;
    x = mu + sigma*randn(N,1);
    SX = S(x);
    sortSX = sortrows([x SX],2);
    Xel = sortSX((N - Nel + 1):N,1);
    mu = mean(Xel);
    sigma = std(Xel);
    fprintf('%g %6.9f %6.9f %6.9f \n', t, S(mu),mu, sigma)
end
mu
```

CE dynamics

```
>> contCE
t S(x)         \mu          \sigma
1 0.218418918 0.765866494 1.762150307
2 0.994523199 2.074107378 0.184167844
3 0.999997222 2.001693158 0.019208654
4 0.999999301 1.999111124 0.001192886
5 1.000000090 1.999981571 0.000130492
6 1.000000090 1.999998597 0.000005735
7 1.000000090 1.999999600 0.000000427
8 1.000000090 1.999999628 0.000000038
9 1.000000090 1.999999641 0.000000003
```

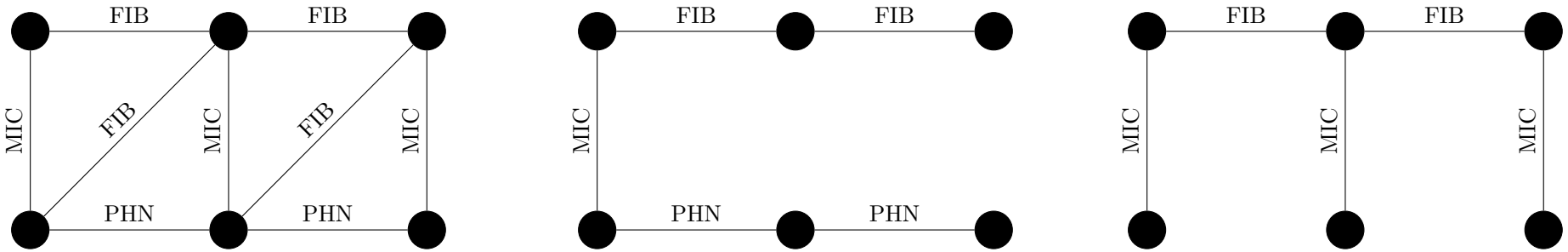
CE dynamics



Real life example — Minimum Label Spanning Trees

- Given a finite undirected graph $G = (V, E, l)$ with a vertex set V, an edge set E, and a labeling function $l : E \to \mathcal{L}$, where $\mathcal{L} = \{1, \ldots, k\}$ is a finite set of labels, the *minimum label(ing) spanning tree* (MLST) problem seeks to find a spanning tree of G which can be constructed using a minimal number of *different* labels.
- The MLST problem appears in several important practical applications such as data compression, and communication network design.
- For example, the next Figure depicts a communication network with optical fiber channels (FIB), telephone lines (PHN), and microwave links (MIC).
- In this case, a minimal label spanning tree can be constructed using only FIB and MIC links.

Real life example — Minimum Label Spanning Trees



(a) Full communication network.

(b) Regular spanning tree.

(c) Minimum label spanning tree.

Figure 2: Panel (a) shows a communication network with optical fiber channels, telephone lines, and microwave communication links. A regular spanning tree and a minimum label spanning tree of the network in panel (a) are depicted in panels (b) and (c), respectively. The spanning tree in panel (b) uses three different channel types, namely, optical fiber channels, telephone lines, and microwave links. On the other hand, the minimum label spanning tree in panel (c), uses only two different channel types, specifically, optical fiber channels and microwave links.

Real life example — Minimum Label Spanning Trees

- The MLST problem belongs to the NP-hard complexity class. The latter has necessitated an introduction of several heuristic and evolutionary methods.
- While some of these methods were shown to have rigorous theoretical performance guarantees, our study indicates that they might fail in practice. That is, the convergence time of these algorithms can be prohibitively large.
- We noted that the corresponding performance depends on the graph structure under consideration.
- To resolve this problem, we applied the CE method which is both theoretically sound, and appears to be less sensitive to different types of graphs.
- Our experimental study indicates that CE always manages to obtain optimal or near-optimal solutions regardless of the network's characteristics.

Real life example — Minimum Label Spanning Trees

To ensure a fair comparison, we examine four different approaches towards the solution of the MLST problem. In particular, we considered the following methods: the *maximum vertex covering algorithm* (MVCA), the (1+1) *Evolutionary Algorithm* ((1+1) EA), the *Global Simple Evolutionary Multiobjective Optimizer* (GSEMO), and the Genetic algorithm (GA).

- We choose these methods for the comparison with the CE algorithm, since they were shown to both have rigorous performance guarantee, and exhibit a good practical performance.
- Specifically, MVCA has an $H_k = \sum_{i=1}^{k} i^{-1}$ approximation factor guarantee.
- In addition, for the special case of the MLST problem in which each label appears at most b times, both the (1+1) EA and the GSEMO algorithms have a $2^{-1}(b+1)$ approximation ratio which is achieved in expected polynomial runtime with respect to $|V|$ and k.
- Moreover, GSEMO was shown to have a $2 \ln |V| + 1$ approximation factor guarantee for the general MLST problem.
- Finally, the GA algorithm was chosen because it is one of the most popular global evolutionary optimization methods.

Minimum Label Spanning Trees — the CE setting

- Consider a binary vector $\boldsymbol{x} = (X_1, \ldots, X_k)$, where $X_i = 1$ (for $1 \leq i \leq k$), stands for the fact that label i participates in the spanning tree construction, and $X_i = 0$ otherwise. That is, we can exploit the binary CE setting.
- We define the fitness function $S : \{0,1\}^k \to \mathbb{R}$

$$S(\boldsymbol{X}) = (\mathfrak{c}(\boldsymbol{X}) - 1) \times k \ln(k) + |\boldsymbol{X}| \quad \text{for } k \geq 3,$$

where $\mathfrak{c}(\boldsymbol{X})$ is the number of connected components in G', ($G' = (V, E')$, $E' = \{e \in E \mid X_{l(e)} = 1\}$), $\ln(k)$ stands for the natural logarithm of k, and $|\boldsymbol{X}| = \sum_{i=1}^{k} X_i$ is the total number of labels used in the spanning tree construction.

MLST — Test-case 1: the binary tree

- In order to benchmark the performance of all algorithms, we consider a binary tree of height 10, namely, the corresponding graph has 1024 vertices and 1023 edges.
- Next, we created 10 MLST problem instances using this tree as follows. For $j \in \{10, 20, \ldots, 100\}$, define $\mathcal{T}_j$ to be a binary tree of height 10, where the corresponding edge labels were assigned uniformly at random from the $\{1, \ldots, j\}$ set.
- For convenience, we force every label from the $\{1, \ldots, j\}$ set to be selected at least once. Since all edges are required to be present in the optimal solution, we conclude that the number of necessary labels in $\mathcal{T}_j$ is j.
- All algorithms (except of MVCA), were forced to comply with runtime thresholds of 5 seconds. Each algorithm was executed ten times and for each run, we used the $12345+i$, $i = 0, \ldots, 9$ seed. The results are summarized in the next slide.
- One can observe from the table that the GSEMO algorithm experienced convergence problems, when the number of used labels increased.

MLST — Test-case 1: the binary tree

Table 1: Average performance among ten runs of the MVCA, the (1+1) EA, the GSEMO, the GA, and the CE algorithms when applied to binary trees with various number of labels. The time-limit for all algorithms (except of the MVCA) is set to be 5 seconds. For the $\mathcal{T}_{60}$ case (*), GSEMO found the correct solution in 4 out of 10 runs. For $\mathcal{T}_{70}$, $\mathcal{T}_{80}$, $\mathcal{T}_{90}$, and $\mathcal{T}_{100}$, GSEMO failed to converge.

instance	MVCA	time (sec)	(1+1) EA	GSEMO	GA	CE
$\mathcal{T}_{10}$	**10**	5.15×10^{-3}	**10**	**10**	**10**	**10**
$\mathcal{T}_{20}$	**20**	1.56×10^{-2}	**20**	**20**	**20**	**20**
$\mathcal{T}_{30}$	**30**	2.75×10^{-2}	**30**	**30**	**30**	**30**
$\mathcal{T}_{40}$	**40**	4.98×10^{-2}	**40**	**40**	**40**	**40**
$\mathcal{T}_{50}$	**50**	7.95×10^{-2}	**50**	**50**	**50**	**50**
$\mathcal{T}_{60}$	**60**	8.78×10^{-2}	**60**	**60***	**60**	**60**
$\mathcal{T}_{70}$	**70**	1.20×10^{-1}	**70**	-	**70**	**70**
$\mathcal{T}_{80}$	**80**	1.65×10^{-1}	**80**	–	**80**	**80**
$\mathcal{T}_{90}$	**90**	2.23×10^{-1}	**90**	–	**90**	**90**
$\mathcal{T}_{100}$	**100**	2.71×10^{-1}	**100**	–	**100**	**100**

MLST — Test-case 2: the 2D-grid

- Here we consider a 32×32 2D-grid with $|V| = 1024$ vertices and $|E| = 1984$ edges.
- In particular, we examine five 32×32 2D-grid instances where each instance has a different number of labels. These grids are being constructed as follows.
- First, we define a label density d. Then, for each grid instance, we set the number of possible labels k to be $\lfloor |V| \times d \rfloor$, where $d \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$ and $\lfloor \cdot \rfloor$ is the ceiling function.
- In order to finalize the construction, labels are assigned to grid's edges uniformly at random. Each algorithm (except of the MVCA method), was given a 120 seconds runtime threshold. The Table on the next slide summarizes the obtained results.

MLST — Test-case 2: the 2D-grid

Table 2: Performance of the MVCA, the (1+1) EA, the GSEMO, the GA, and the CE algorithms when applied to 32×32 2D-grids with different label densities. The time-limit for all algorithms (except of the MVCA) is set to be 120 seconds.

label density d	MVCA	time (sec)	(1+1) EA	GSEMO	GA	CE
0.1	63	0.136	64	**59**	61	**59**
0.2	109	0.501	116	**103**	117	106
0.3	146	0.951	166	146	161	**144**
0.4	186	1.497	210	184	237	**177**
0.5	219	2.234	260	229	257	**216**

The Table is instructive in the sense that for this particular network architecture, (1+1) EA and the GA algorithms introduce the worse performance. The GSEMO algorithm is almost always better than MVCA (with the exception of the $d = 0.5$ case). However, when we consider the $0.1, 0.3, 0.4$, and the 0.5 densities, CE delivers superior results. For the 0.2 density, however, GSEMO outperforms the CE algorithm.

MLST — Test-case 2: the 2D-grid (why CE is worse when GA for $d = 0.2$?)

- The inferior performance of CE as compared to GSEMO for the 0.2 density case (note that CE and GSEMO achieve the fitness of 106 and 103, respectively), requires a careful discussion.
- It turns out that CE performance can be improved by increasing the sample size N. The reason for this suggestion is straight-forward.
- Recall that the CE algorithm estimates the optimal sampling distribution via the stochastic counterpart.
- Clearly, increasing the sample size N will improve the estimation, since we obtain a better approximation of the expected value

$$\mathbb{E}_{f(\boldsymbol{x};\boldsymbol{p}_t)} 1_{\{S(\boldsymbol{X}) \leq \gamma_t\}} \times \ln \left(\prod_{i=1}^{k} p_{t+1,i}^{x_i} (1 - p_{t+1,i})^{1-x_i} \right),$$

as N grows.

MLST — Test-case 2: the 2D-grid

- Next, the CE algorithm is executed on the 0.2 density grid instance (with $k = \lfloor 1024 \times 0.2 \rfloor = 204$), using the sample size $N = 20 \times k = 20 \times 204 = 4080$ (instead of the default $N = 10 \times 204 = 2040$ sample size).
- We still impose the 120 seconds runtime threshold.
- The CE algorithm manages to discover the best known solution that uses 102 labels; note that this is an improvement over the GSEMO algorithm solution that uses 103 labels.
- The next slide depicts the dynamic of the CE algorithm for the 0.2 density grid.

MLST — Test-case 2: the 2D-grid

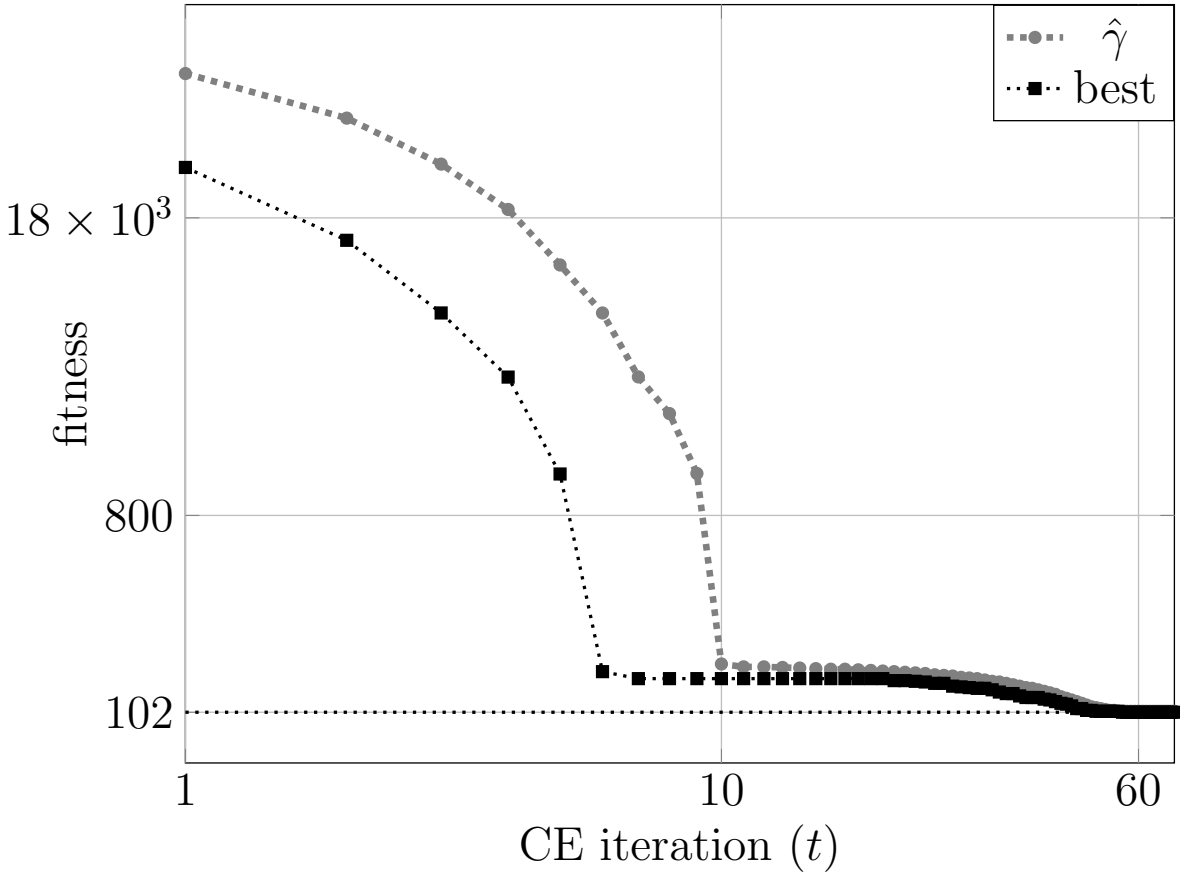


Figure 3: CE dynamics of the sample quantile $\hat{\gamma}$ and the best observed fitness as a function of the CE algorithm iteration for the 0.2 density grid with $N = 4080$ sample size.

Conclusion

- CE is a great global optimizer.
- CE is easy to program and apply.
- CE can be extended to continuous optimization problems (easily!)
- CE can handle noisy optimization problems.
- CE was shown to perform very well in many application domains.

If you would like to know more:

- *Simulation and the Monte Carlo Method, 3rd Edition* by Reuven Y. Rubinstein, Dirk P. Kroese.
- *Handbook of Monte Carlo Methods* by Dirk P. Kroese, Thomas Taimre, Zdravko I. Botev.
- *Fast Sequential Monte Carlo Methods for Counting and Optimization* by R.Y. Rubinstein, A. Ridder and R. Vaisman.

Thank You