Parallel Cross-Entropy Optimization

By: Gareth Evans

Department of Mathematics, The University of Queensland, Australia

With thanks to: Jonathan M. Keith and Dirk P. Kroese

Contents

1. Motivation
2. The Cross-Entropy Method
3. Parallel Cross-Entropy
4. Results

Motivation

A common problem when solving complex optimization or estimation problems is the prohibitively large computational time required. For certain problems, such as genetic sequence segmentation, this time can be in the order of days or weeks or more.

Motivation

A common problem when solving complex optimization or estimation problems is the prohibitively large computational time required. For certain problems, such as genetic sequence segmentation, this time can be in the order of days or weeks or more.

One possible approach to decrease this computation time is to carry out calculations simultaneously, that is, to use an algorithm with a parallel implementation.

Previous Work

- Previous small scale parallel implementation work done by Sergei Porotsky (private communication).
- Another paper by Zhanhua Bai and Qiang Lv presented an approach to parallel CE but violated several CE assumptions.

The Cross-Entropy method

The Cross-Entropy or CE method can be used for two types of problems:

- Estimation
- Optimization

The Cross-Entropy method

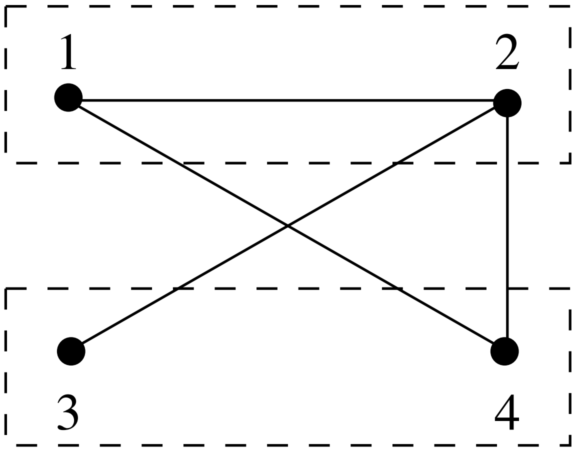
The Cross-Entropy or CE method can be used for two types of problems:

- Estimation
- Optimization

Here we focus on the parallel implementation of the CE method for optimization problems, although similar techniques will be possible for estimation.

The Max-Cut Problem

The Max-Cut problem is: Given a graph which two sets of the vertices maximizes the costs of the edges between them?



The Max-Cut Problem

For example, if we had the cut $\{\{1,3\},\{2,4\}\}$ with the following cost matrix

$$\begin{pmatrix} 0 & c_{12} & 0 & c_{14} \\ c_{21} & 0 & c_{23} & c_{24} \\ 0 & c_{32} & 0 & 0 \\ c_{41} & c_{42} & 0 & 0 \end{pmatrix}$$

the cost of the cut would be $c_{12} + c_{14} + c_{23}$.

CE Optimization

There are two key parts to any CE optimization algorithm.

CE Optimization

There are two key parts to any CE optimization algorithm.

- The representation and random generation of a sample from a sampling distribution.

CE Optimization

There are two key parts to any CE optimization algorithm.

- The representation and random generation of a sample from a sampling distribution.
- The updating of the sampling distribution based on the best of the previous sample (follows from CE minimization).

CE Optimization

There are two key parts to any CE optimization algorithm.

- The representation and random generation of a sample from a sampling distribution.
- The updating of the sampling distribution based on the best of the previous sample (follows from CE minimization).

For the Max-Cut problem we represent each cut as a binary vector $\boldsymbol{x} = (x_2, \ldots, x_n)$ where $x_i = 1$ if vertex i is in the same partition as vertex 1.

Max-Cut updating formula

The updating formula for the probability vector p at the t-th iteration is given by

$$\hat{p}_{t,i} = \frac{\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_k) \geq \hat{\gamma}\}} I_{\{X_{ki}=1\}}}{\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_k) \geq \hat{\gamma}\}}}.$$

The CE Algorithm the Max-Cut problem

Algorithm 1 (Max-Cut)

1. *Initialize* P *with* $p_i = \frac{1}{2}\ \forall i > 1$
2. *Generate a sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *of binary vectors and score each cut.*
3. *Compute the sample* $(1-\rho)$*-quantile* $\hat{\gamma}_t$ *of the performances and update* P *via CE formula.*
4. *Apply smoothing.*
5. *Determine if stopping condition is met otherwise reiterate from step 2.*

What to parallelize?

Which steps in this algorithm can be parallelized? And how?

What to parallelize?

Which steps in this algorithm can be parallelized? And how?

- The sample generation

What to parallelize?

Which steps in this algorithm can be parallelized? And how?

- The sample generation
- By the (equal) division of the sampling and scoring over all processors

Results

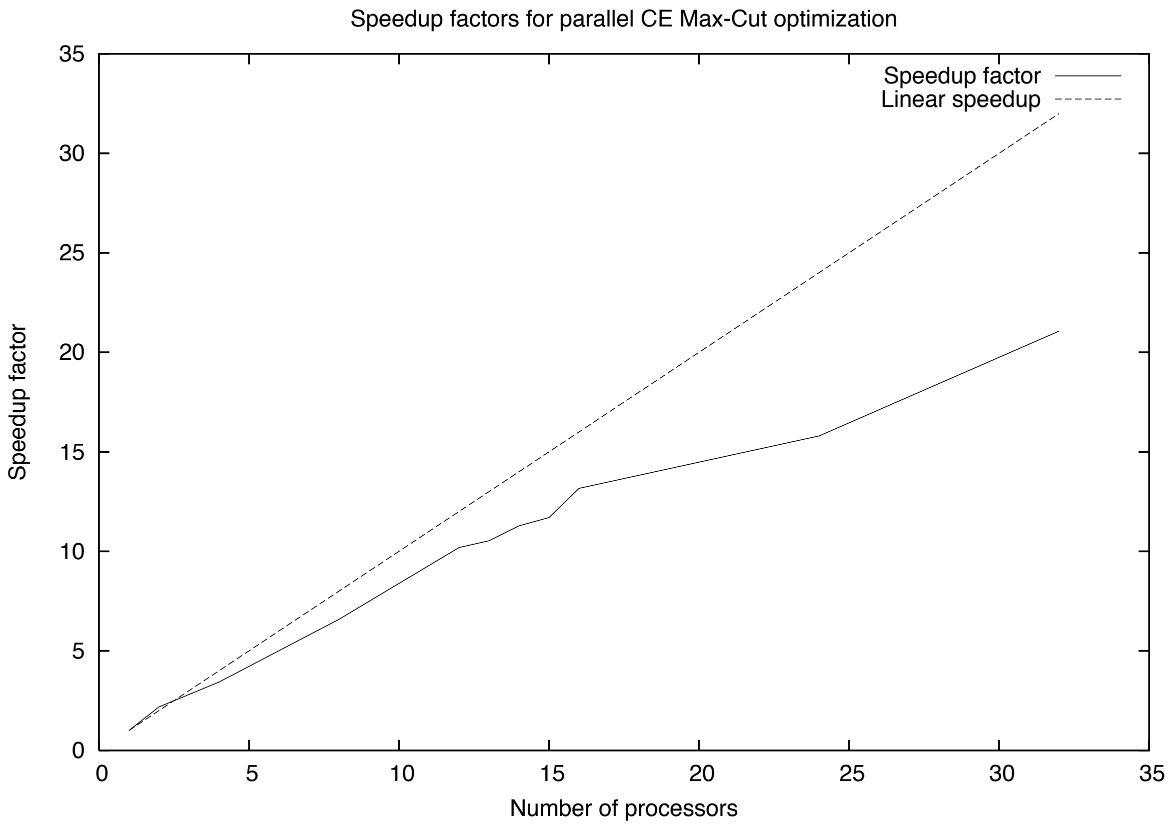


Figure 1: Speedup of the parallel Max-Cut CE algorithm

Results

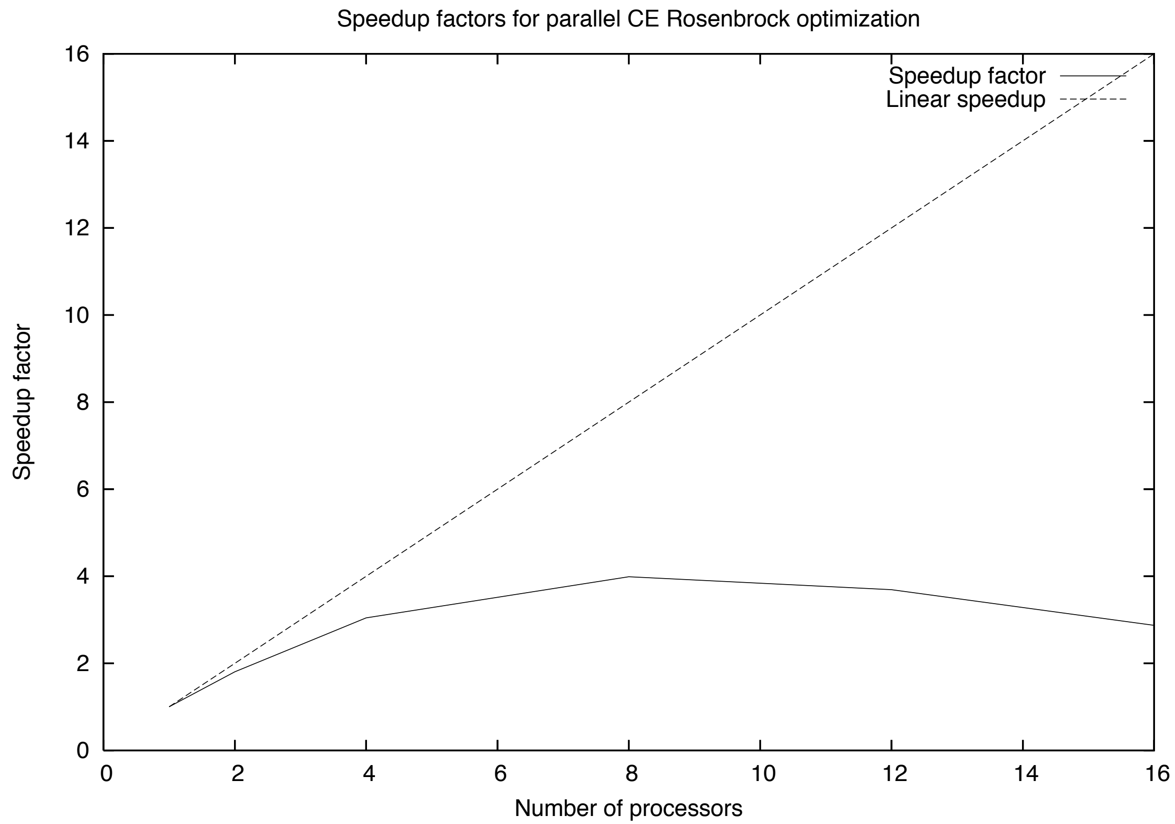


Figure 2: Speedup of the parallel 5-dimensional Rosenbrock CE algorithm

What to parallelize?

Which steps in this algorithm can be parallelized? And how?

- The sample generation
- By the (equal) division of the sampling and scoring over all processors

What to parallelize?

Which steps in this algorithm can be parallelized? And how?

- The sample generation
- By the (equal) division of the sampling and scoring over all processors
- The updating procedure

What to parallelize?

Which steps in this algorithm can be parallelized? And how?

- The sample generation
- By the (equal) division of the sampling and scoring over all processors
- The updating procedure
- By having each processor calculate their relative portion of the new sampling distribution before a single processor combines these partial updates.

Parallel updating

- After scoring their partial sample, each processor communicates to a single processor their best $\min(N/s, e)$ scores, where e is the size of the elite sample and s is the number of processors.

Parallel updating

- After scoring their partial sample, each processor communicates to a single processor their best $\min(N/s, e)$ scores, where e is the size of the elite sample and s is the number of processors.
- The single processor calculates how many cuts c_j from each processor j belong in the elite sample.

Parallel updating

- After scoring their partial sample, each processor communicates to a single processor their best min$(N/s, e)$ scores, where e is the size of the elite sample and s is the number of processors.
- The single processor calculates how many cuts c_j from each processor j belong in the elite sample.
- Each processor calculates $q_{j,i} = \sum_{k=1}^{c_i} I_{\{S(\boldsymbol{X}_k) \geq \hat{\gamma}\}} I_{\{X_{ki}=1\}}$

Parallel updating

- After scoring their partial sample, each processor communicates to a single processor their best $\min(N/s, e)$ scores, where e is the size of the elite sample and s is the number of processors.
- The single processor calculates how many cuts c_j from each processor j belong in the elite sample.
- Each processor calculates $q_{j,i} = \sum_{k=1}^{c_i} I_{\{S(\boldsymbol{X}_k) \geq \hat{\gamma}\}} I_{\{X_{ki}=1\}}$
- The new sampling distribution updating formula is then $\hat{p}_{t,i} = \frac{\sum_{k=1}^{s} q_{k,i}}{\sum_{k=1}^{N} I_{\{S(\boldsymbol{X}_k) \geq \hat{\gamma}\}}}$

Results

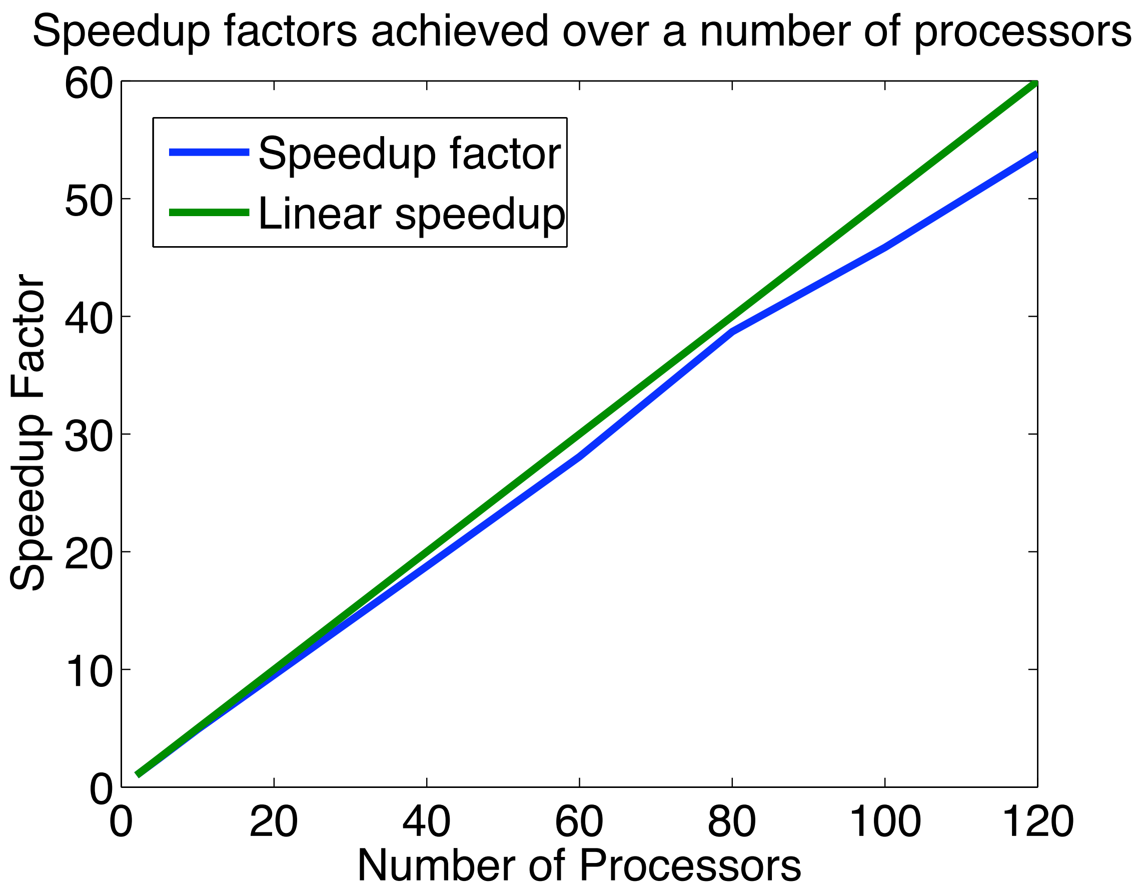


Figure 3: Speedup of the parallel Max-Cut CE algorithm using a graph with 2000 vertices

Results

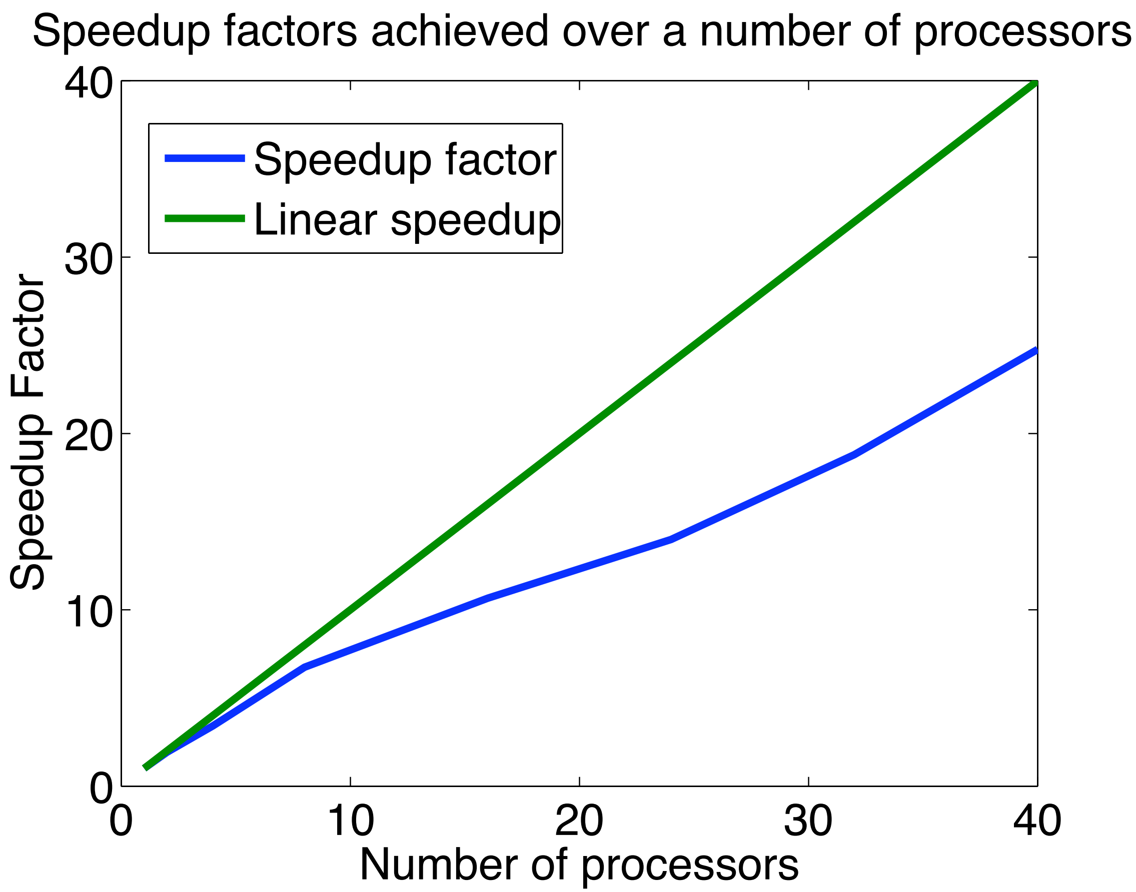


Figure 4: Speedup of the parallel Max-Cut CE algorithm using a graph with 1000 vertices

Results

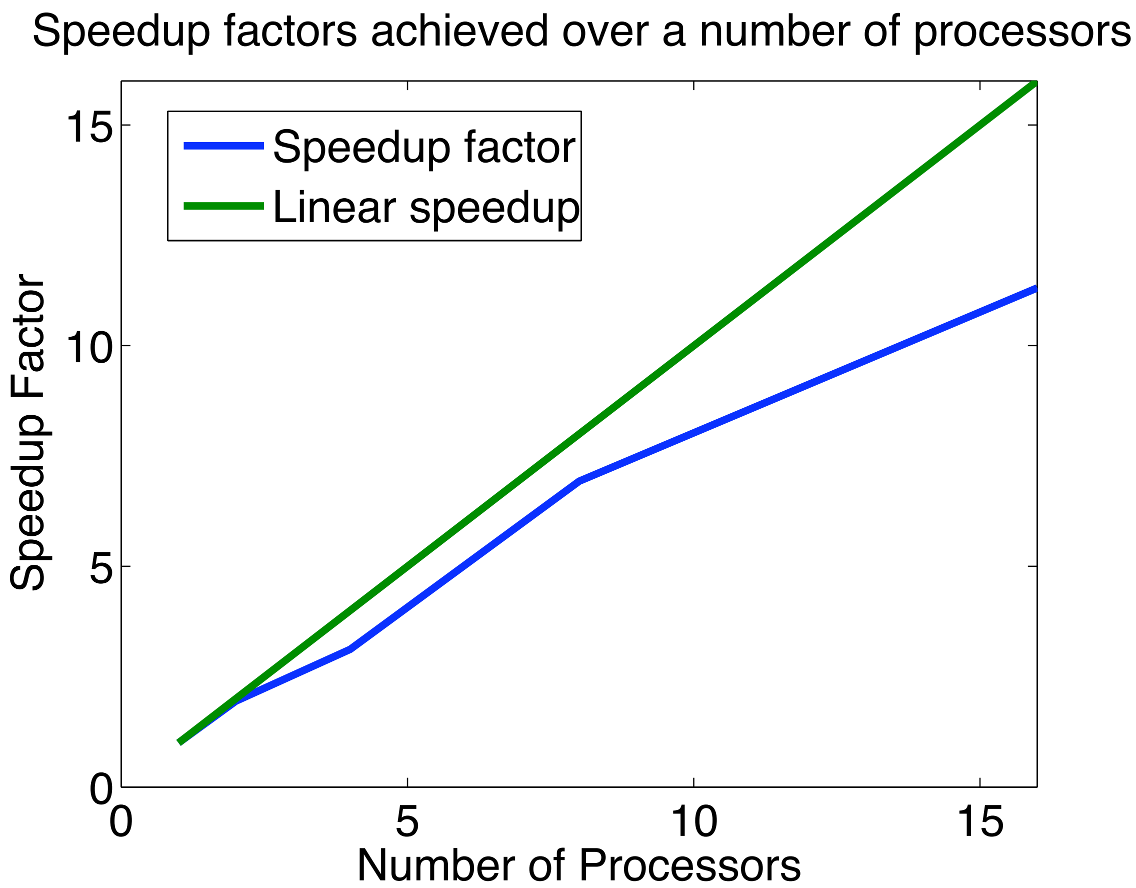


Figure 5: Speedup of the parallel 15-dimensional Rosenbrock CE algorithm

Discussion

- Communication overhead

Discussion

- Communication overhead
- Problem size

Discussion

- Communication overhead
- Problem size
- Division of updating

Discussion

- Communication overhead
- Problem size
- Division of updating
- Unequal division of work

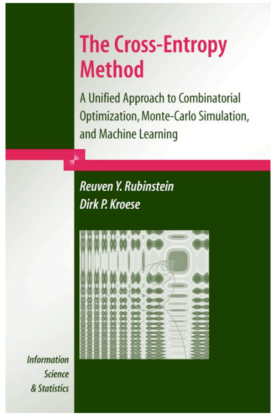
Questions?

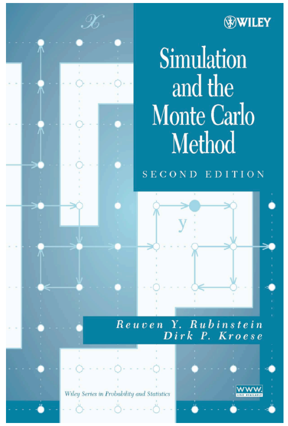
If you like the look of my work I am looking for a Postdoc :-)

See also ...

Books: R.Y. Rubinstein and D.P. Kroese.

- *The Cross-Entropy Method*, Springer-Verlag, 2004.
- *Simulation and the Monte Carlo Method*, 2nd Edition, Wiley & Sons, 2007.





The CE home page: `http://www.cemethod.org`

Special Issue: *Annals of Operations Research, 2009*. Monte Carlo Methods for Simulation, Optimization and Counting.