A Graph Partitioning Algorithm

on the Planted Partition Model

Mikael Onsjö
mikael.onsjo@chalmers.se

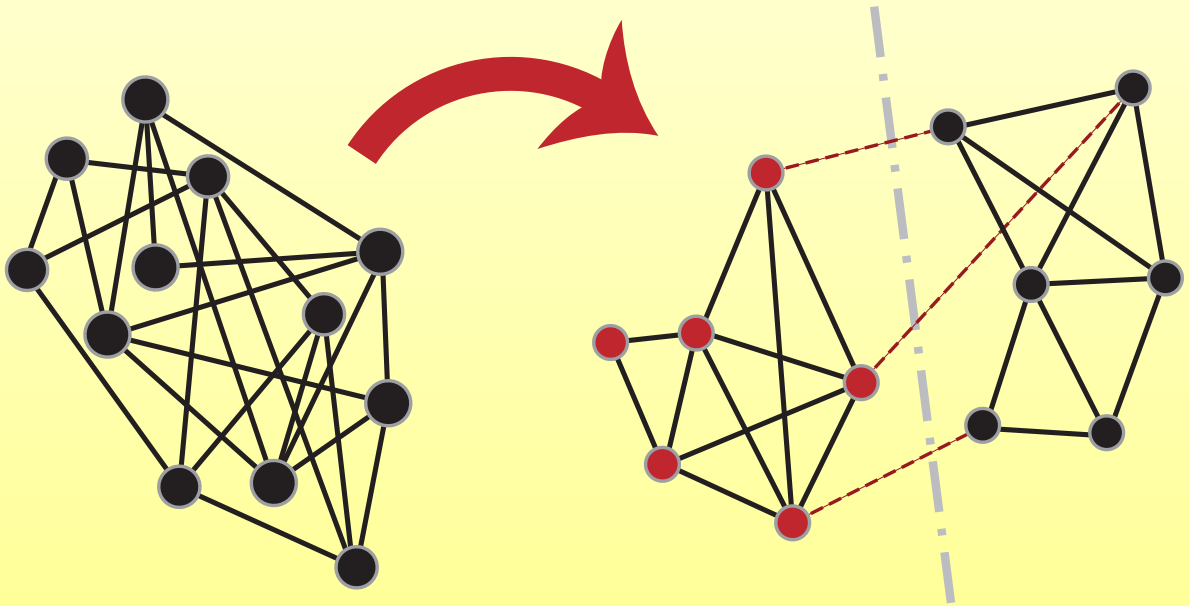
Department of Computer Science and Engineering
Chalmers University of Technology

Göteborg 2007

CHALMERS

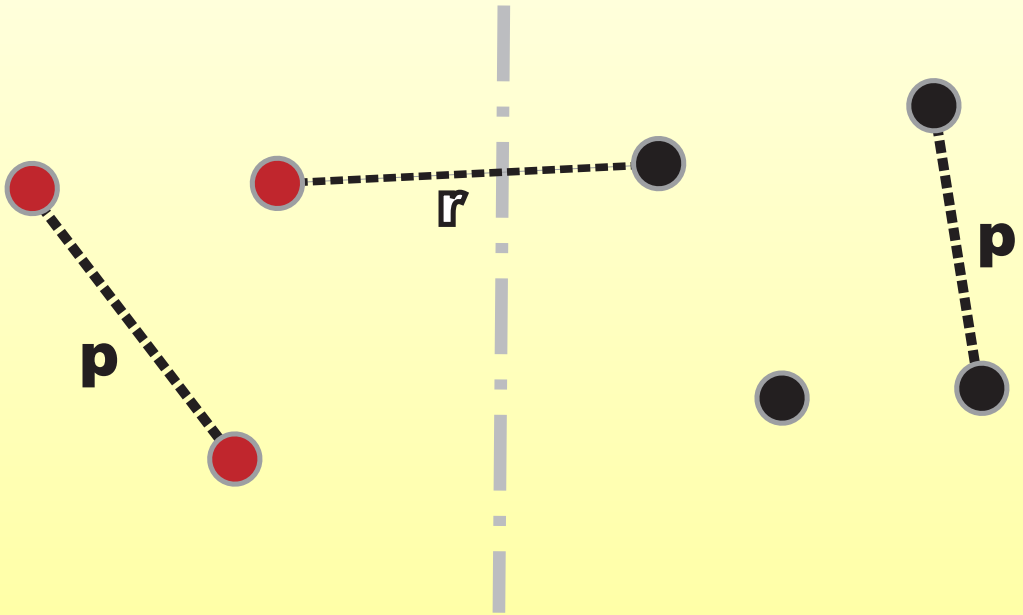
Graph partitioning is the task of splitting a graph into a specific number of partition of specific sizes such as to minimize the number of edges that cross between partitions.

- Let's focus on the special case Graph Bisection:



CHALMERS

The planted partition model defines a different input distribution of graphs:



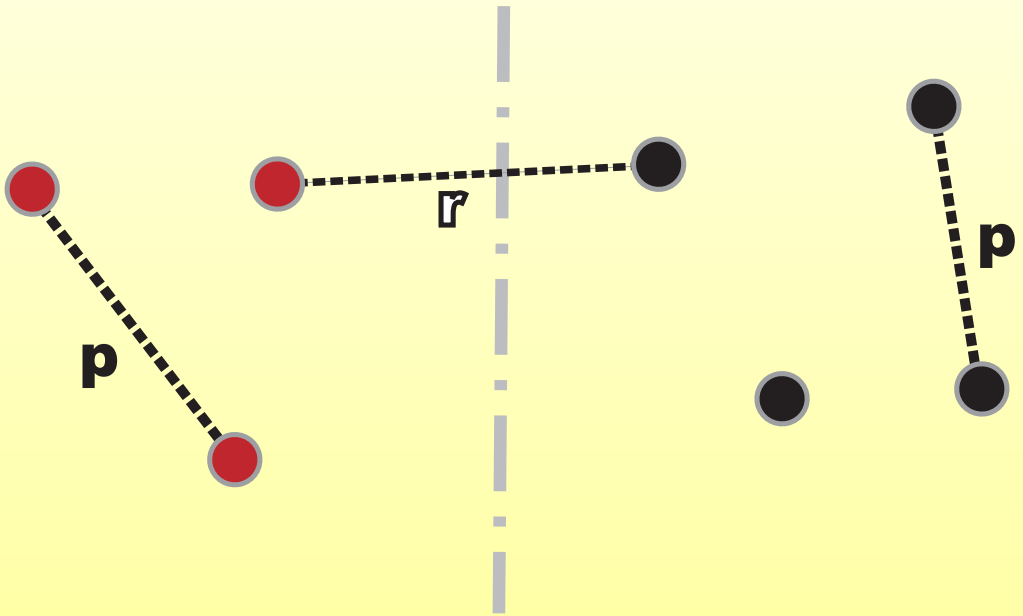
- All graphs are still possible but no longer equally likely

- The Graph Bisection problem is NP-complete
- Under a uniform input distribution the optimal and worst case solutions are likely to be equally good
- Under the planted partition model it holds that the planted solution is optimal *with high probability*

By *with high probability* (whp.) we mean that probability goes to one exponentially as the size of the graph increases.

The Planted Partition Model is applicable, for instance, in categorization and collaborative filtering on the Internet.
Ex: To see how it can arise:

- Assume there are two factions in a group of people
- People of the same faction could be assumed to talk to each other with a high probability p.
- People of a different faction talk to each other with a lower probability r.
- Identifying the factions only by observing who speaks to whom is then exactly the problem we study here.



CHALMERS

It is convenient to focus on finding the planted partition that we assume exists. To do this it is natural to define a slightly different problem:

Given a graph and probability parameters p and r, find the partition that is most likely to have generated the graph.

With appropriate definitions this can be expressed as:

$$max \prod_{\{i,j\}\in E} p^{\delta(i,j)} r^{1-\delta(i,j)} \cdot \prod_{\{i,j\}\in \overline{E}} (1-p)^{\delta(i,j)} (1-r)^{1-\delta(i,j)} \qquad (1)$$

We call this problem *Maximum Likelihood Partitioning* and MLP for short.

CHALMERS

Result: MLP is NP-complete

Result: The optimal solution to MLP under the planted partition model is the planted partition whp.

MLP and Graph Bisection are not equivalent but they both have the same optimal solution, that is the planted partition, whp.

Approach:

- We derive a Bayesian Network for MLP
- We use the Belief Propagation algorithm for deriving marginal probabilities in Bayesian Networks
- We make minor simplifications to the algorithm
- We show theoretical results
- We show experimental results

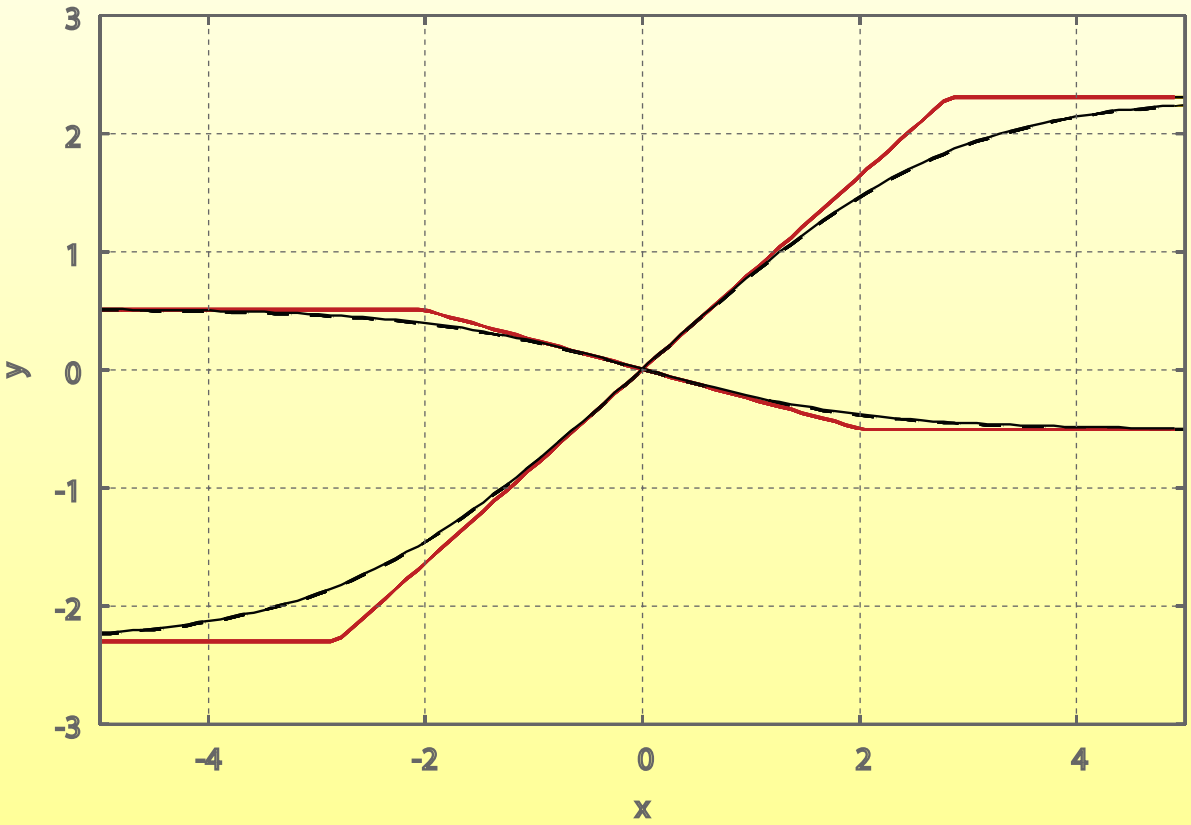
The algorithm is quite simple:

- Define a real variable, d_i, for each vertex
- Initially $d_i \leftarrow 0$ for all but one vertex: $d_0 \leftarrow \infty$
- Synchronously update the values using:

$$d_i^{n+1} \leftarrow \sum_j f_{edge_{ij}}(d_i^n) \tag{2}$$

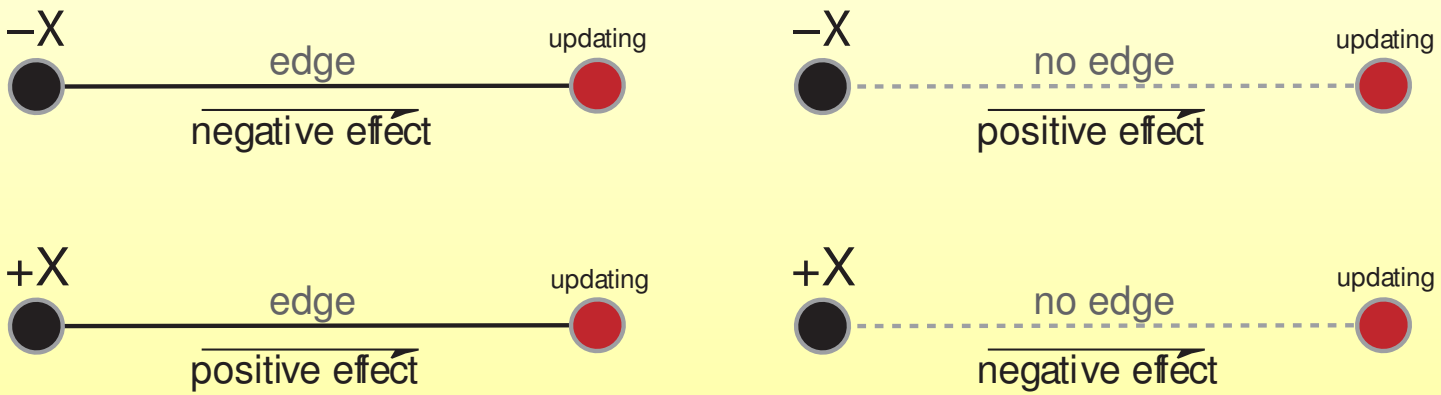
  where $f_{edge_{ij}}$ is either of two function depending on if there is an edge between i and j
- Stop when stable and classify according to $sign(d_i)$

The two "transfer functions" generally look like these:



CHALMERS

The updates "make sense":



Result: If $p - r = \Omega(n^{-1/2} \log(n/\delta))$ then the following holds with a probability larger than $1 - \delta$:

- The algorithm yields the planted partition within two iterations
- The planted partition is the optimal solution

Thus in linear time we succeed with a probability comparable to other popular linear time algorithms today.

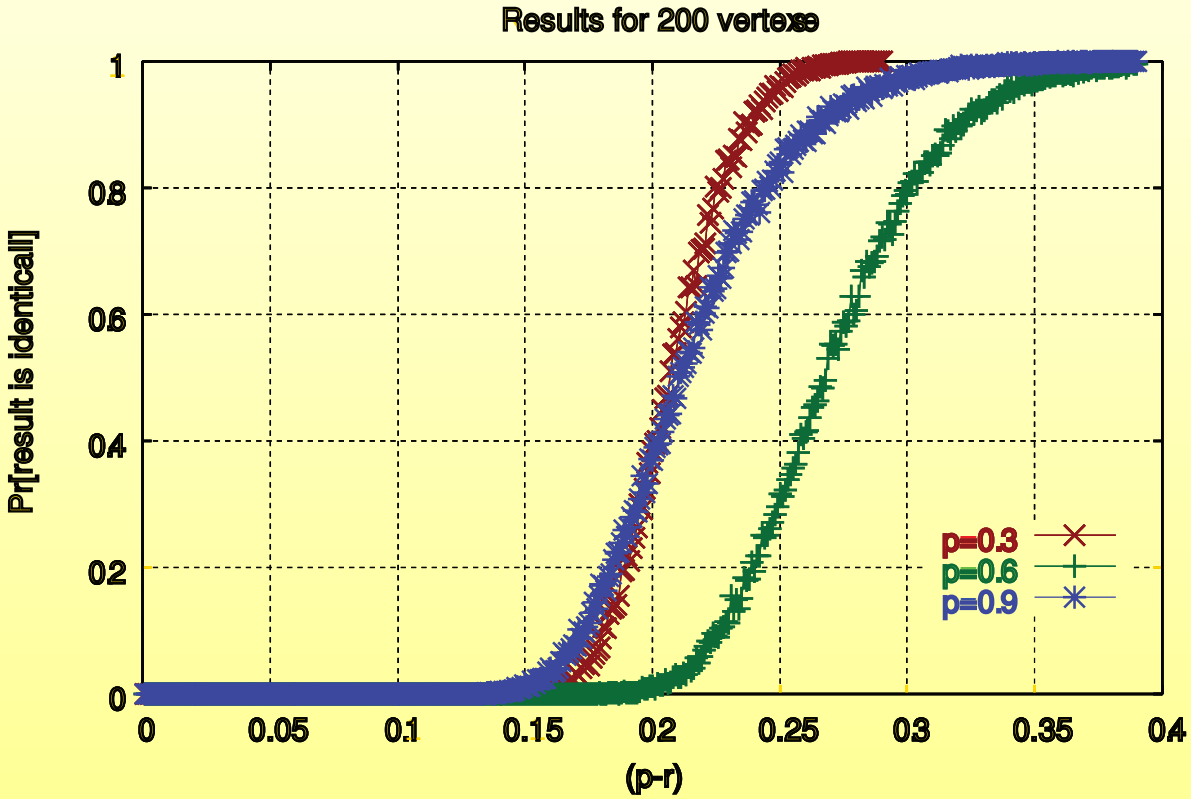
Experiments:

- It is difficult to experiment because comparing with exact algorithms is not tractable for interesting graph sizes.
- We find that our algorithm generally produces results equal or better to what was planted.
- We would need (intend) to compare the algorithm directly to other graph partitioning algorithms.

Simple experiments were conducted as follows:

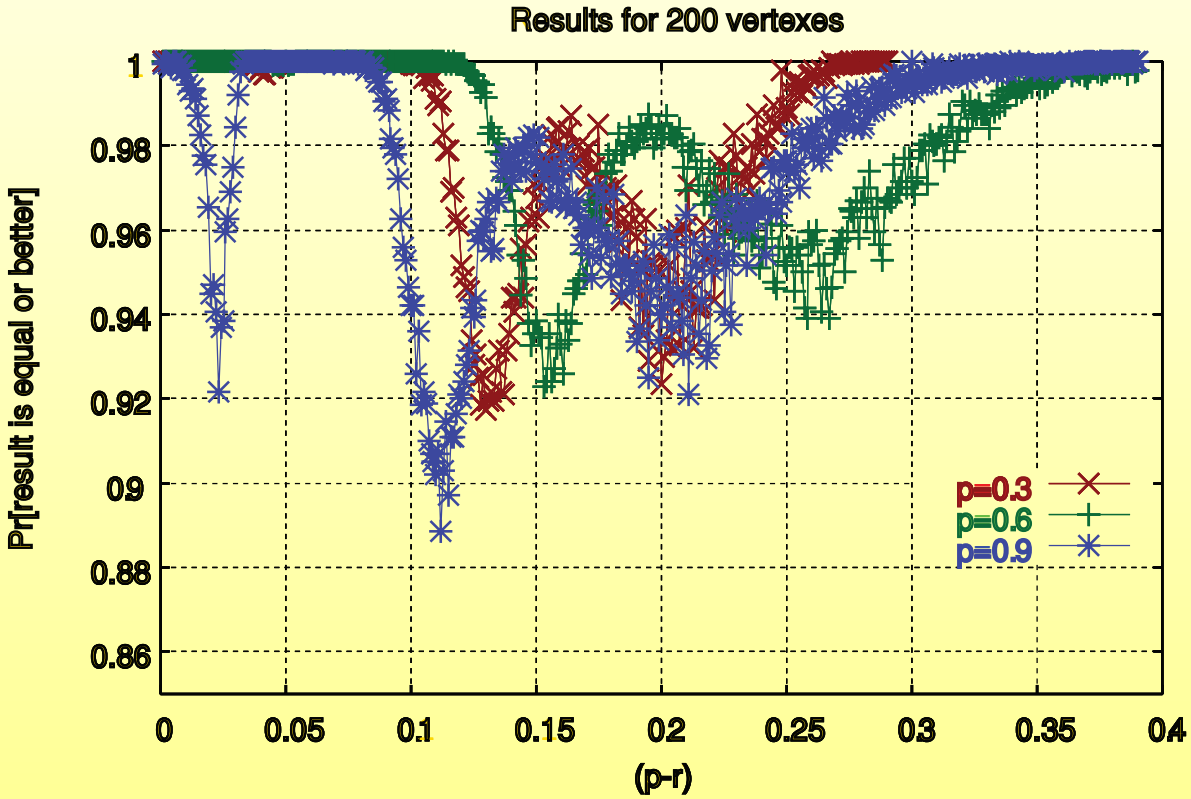
- Generate graph from some "planted partition"
- Run algorithm to obtain a partition
- Calculate the likelihoods of the planted and obtained partitions respectively, call them L and L'
- If the partitions are the same, we succeeded. If not:
  - If $L' = L$ we found an equally good partition
  - If $L' < L$ we found a suboptimal partition
  - If $L' > L$ the planted partition was obviously not optimal since we found something better.

The following graph plots $p - r$ against the probability of finding exactly the planted partition:



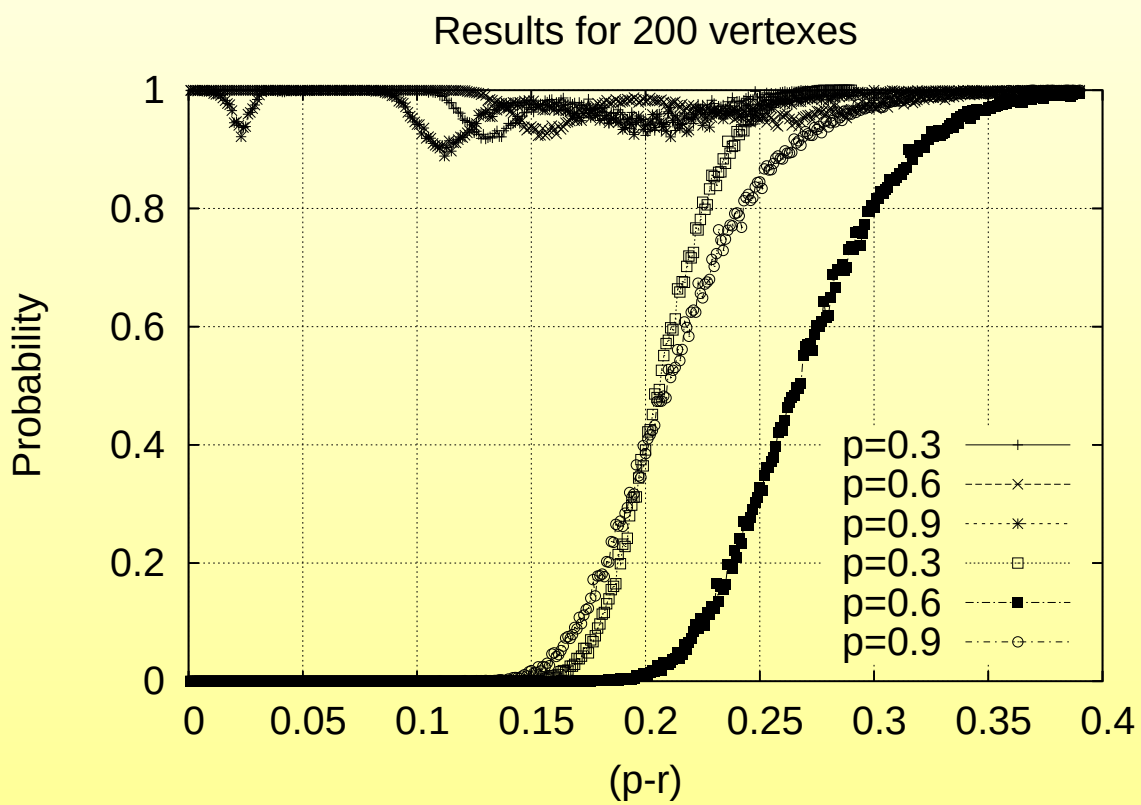
CHALMERS

The following graph plots $p - r$ against finding something that is as good or better than the planted partition.



CHALMERS

...to put them in relation:



CHALMERS

Some final notes:

- The best partitioning algoritms today have better theoretical bounds than ours but are generally more involved and take longer time.
- There are other fast algorithms that have very similar results to ours. BP makes an interesting motivation, though, and we would think this algorithm will work better in practice.
- Our algorithm assumes knowledge of p and r. Though these can be find by a binary search, future work include defining a similar algorithm for the parameterless version.

So to summarize:

- We present a simple linear algorithm for graph partitioning
- The algorithm is based on Belief Propagation
- The algorithm is analyzed using the planted partition model

...

...

- We show that the algorithm works with probability $1 - \delta$ if $p - r = \Omega(n^{-1/2} \log(n/\delta))$.
- The theoretical results are well comparable to those for other common algorithms of similar complexity.
- Experimentally we fail to show any significant weakness of the algorithm, it appears to work well enough.

Thanks for listening!
mikael.onsjo@chalmers.se

CHALMERS