Using Tarjan's Red Rule for Fast Dependency Tree Construction

Dan Pelleg and Andrew Moore

Carnegie-Mellon University

© Dan Pelleg

Motivation

- Given a data-set, want to "understand" it.
- A Bayes' net fits the bill, but expensive to find.
- Compromise: look for a simpler structure — **dependency tree**.

- Typical application: 10^6 points, $100 - 1000$ attributes.
- Astrophysics data.
- Gene-expression data.

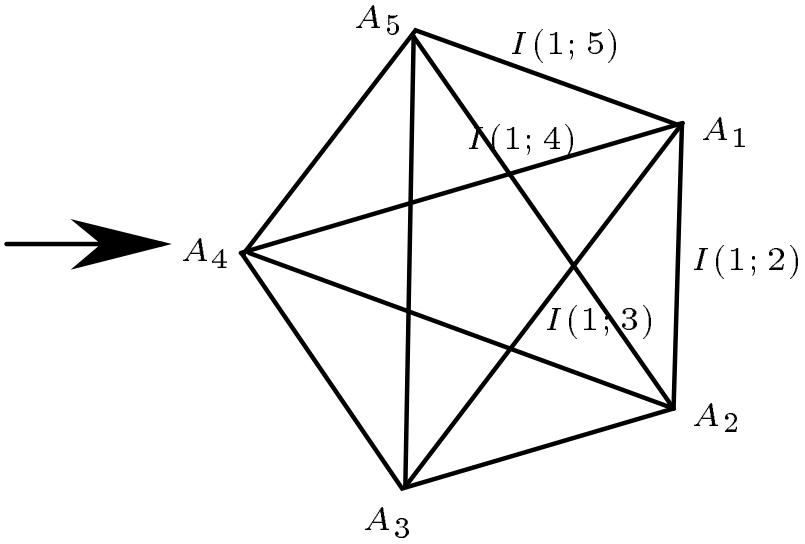
The Chow-Liu algorithm

	A_1	A_2	A_3	$\cdots$	A_N
X_1	·	·	·	·	·
X_2	·	·	·	·	·
X_3	·	·	·	·	·
X_4	·	·	·	·	·
X_5	·	·	·	·	·
⋮					
X_R	·	·	·	·	·

© Dan Pelleg

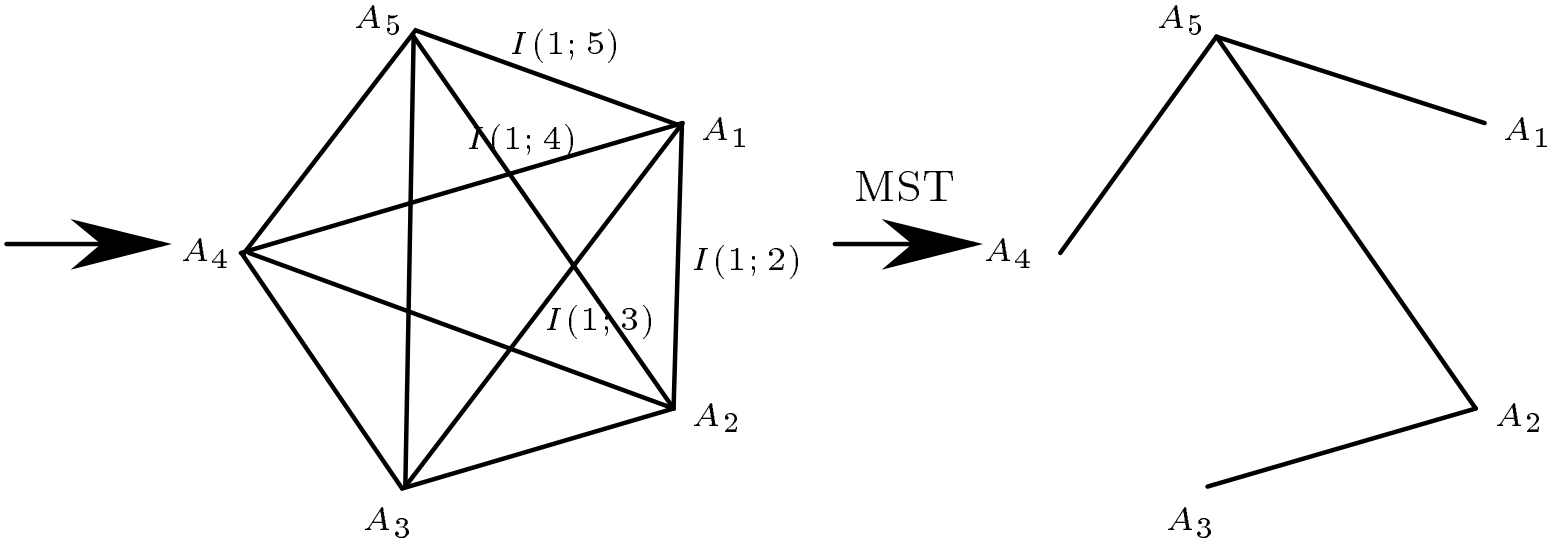
The Chow-Liu algorithm

	A_1	A_2	A_3	$\cdots$	A_N
X_1	·	·	·	·	·
X_2	·	·	·	·	·
X_3	·	·	·	·	·
X_4	·	·	·	·	·
X_5	·	·	·	·	·
⋮					
	·	·	·	·	·
X_R	·	·	·	·	·

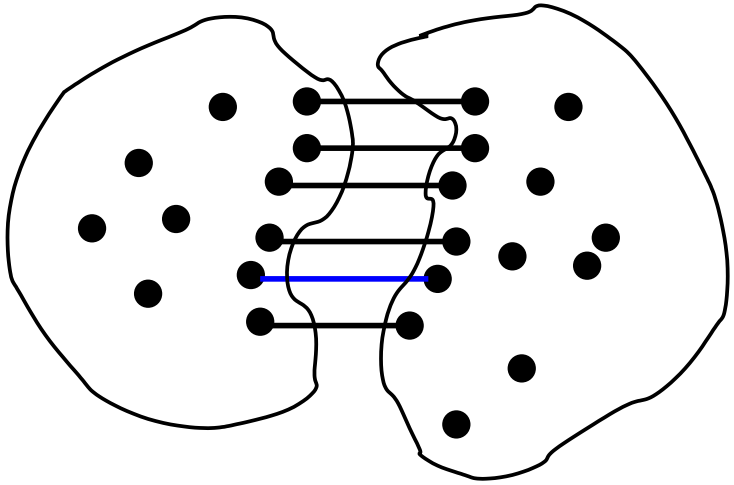


The Chow-Liu algorithm

	A_1	A_2	A_3	$\cdots$	A_N
X_1	·	·	·	·	·
X_2	·	·	·	·	·
X_3	·	·	·	·	·
X_4	·	·	·	·	·
X_5	·	·	·	·	·
⋮					
X_R	·	·	·	·	·

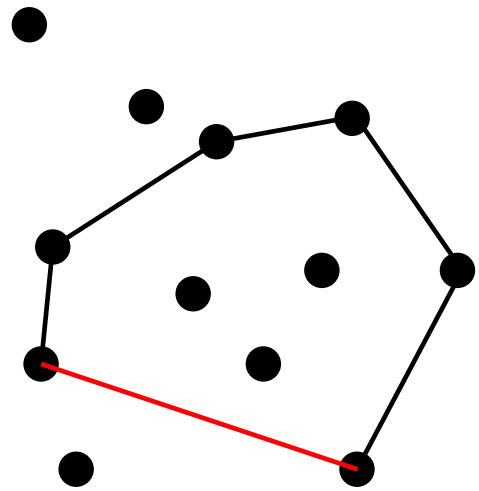


MST: using the blue-edge rule



Given a cut, the lightest edge across it **must** be part of the MST.

MST: using the red-edge rule



Given a cycle, the heaviest edge in it **must not** be part of the MST.

Idea: repeatedly use the red-edge rule

- Stop when all we have left is a tree.
- This tree must be the MST.

Tarjan: "bad idea".

An exclusive red-edge algorithm

Invariant: There is always a spanning tree in which all edges are not eliminated.

Repeat until all but $(N-1)$ edges are eliminated:

- Pick a non-tree, non-eliminated edge.
- Add the tree path connecting its endpoints to create a cycle.
- Apply the red-edge rule to this cycle:
  - If the non-tree edge is red, eliminate it.
  - If a tree edge is red, eliminate it and replace it with the non-tree edge.
- Output tree as the MST.

© Dan Pelleg

An observation regarding edge weights

- ✔ We don't really care about knowing the exact edge weights.
- ✔ All we need is their relative order.
- ✔ True for all MST algorithms.

This still doesn't give us an advantage. But if we change the rules a little...

A procrastinating oracle

- Now suppose there's an oracle that, given two edges, can tell us which is heavier.

$ There's a fixed charge per query.

- But the oracle may refuse to answer.
- When it does answer, it doesn't lie.

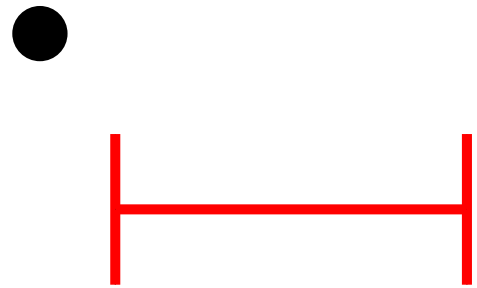
Representing edges as intervals

We want to avoid scanning the full data-set for a given edge.

- Scan just a sample.
- Derive a confidence interval using the CLT.
  - Or Hoeffding bounds.
- Now need to deal with intervals instead of point-estimates.
- To make our lives easier, we'll still use the whole data for tree edges.

Comparing points and intervals

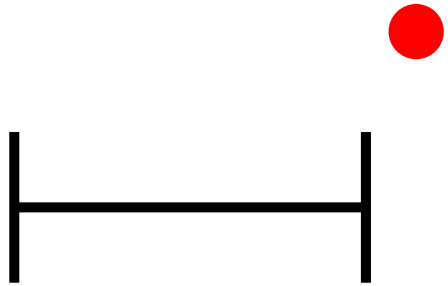
Case 1:



This is where we save work.

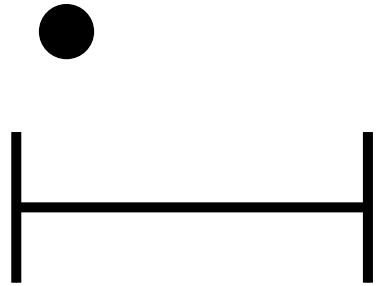
Comparing points and intervals

Case 2:



Comparing points and intervals

Case 3:



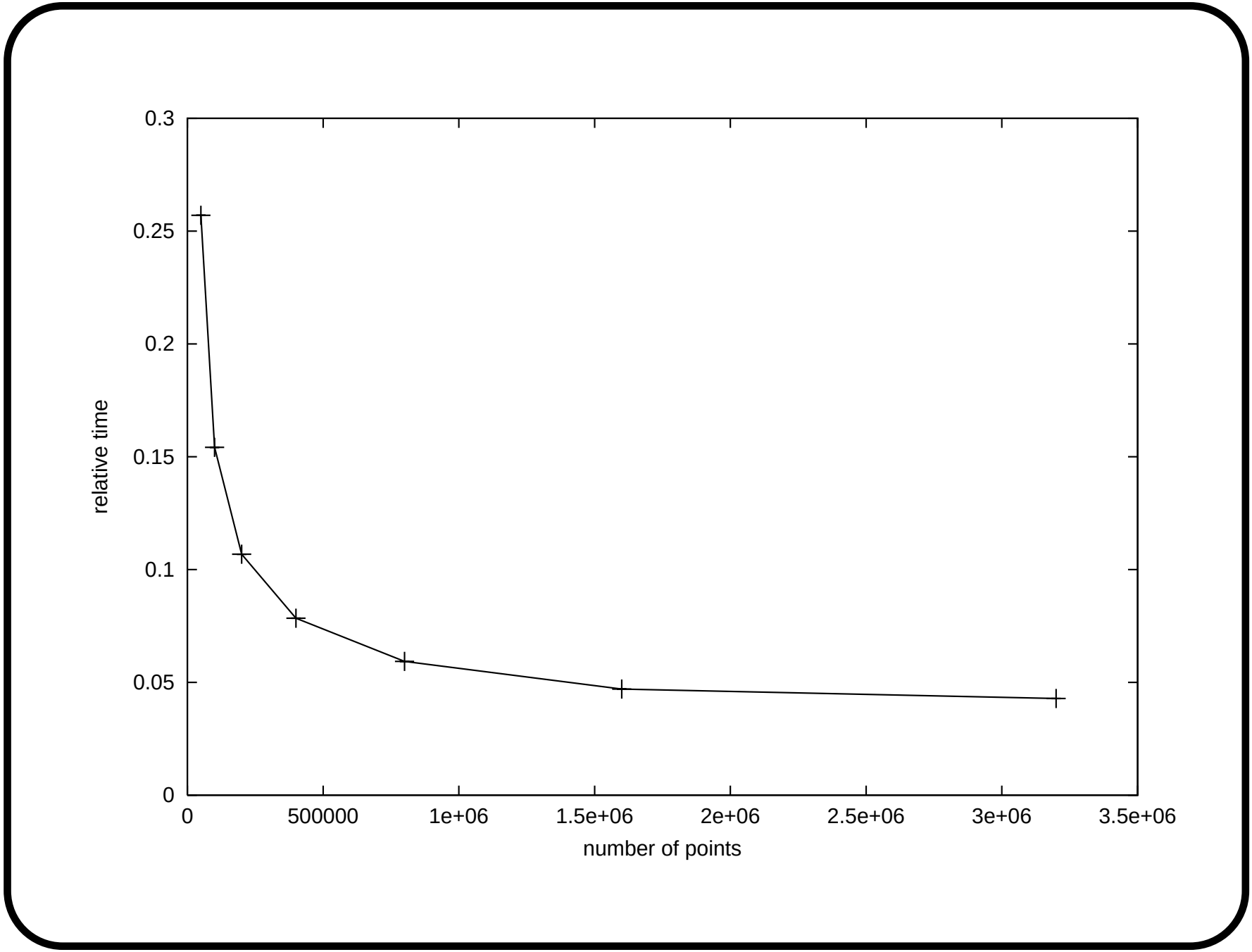
The full algorithm

- Use a small sample to estimate edge weights, build MST from that.
- Keep on picking non-tree, non-eliminated edges and trying to apply the red-edge rule.
  - Try to reach a conclusion using data at hand.
  - If impossible, double the sample-size for the non-tree edge and try again.
  - If still can't decide, leave that edge alone for now and pick another one.
  - A successful resolution means that one edge is eliminated.
    * It's either the tree edge or the non-tree edge.
- Whenever an edge goes into the tree, full data for it is scanned and a point estimate derived.

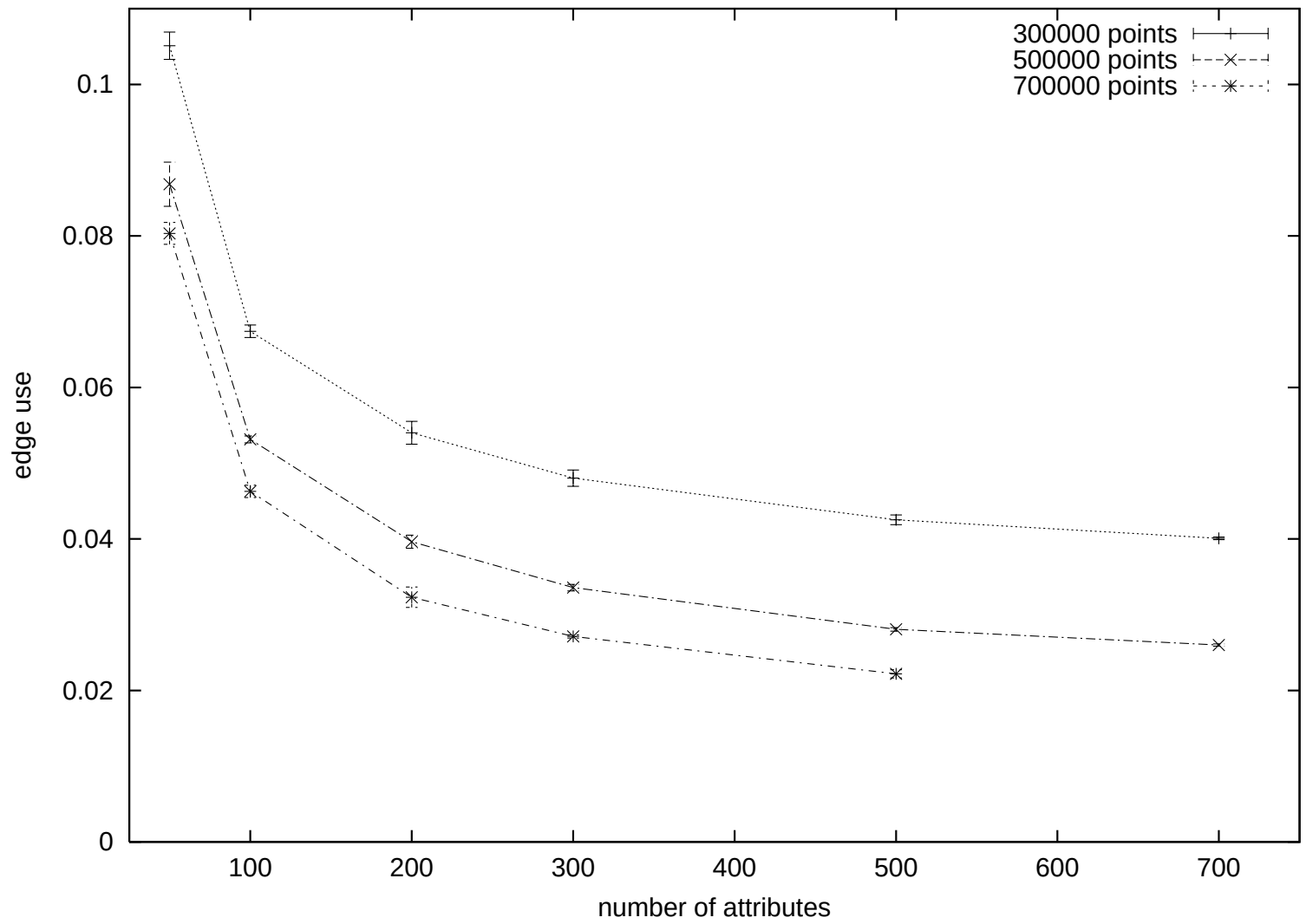
Experimental Results

- How does it scale with the number of points?
- How does it scale with the number of attributes?
- How often does is output a sub-optimal tree?
- How bad are the sub-optimal trees?

Speedup as a function of #points



Speedup as a function of #attributes

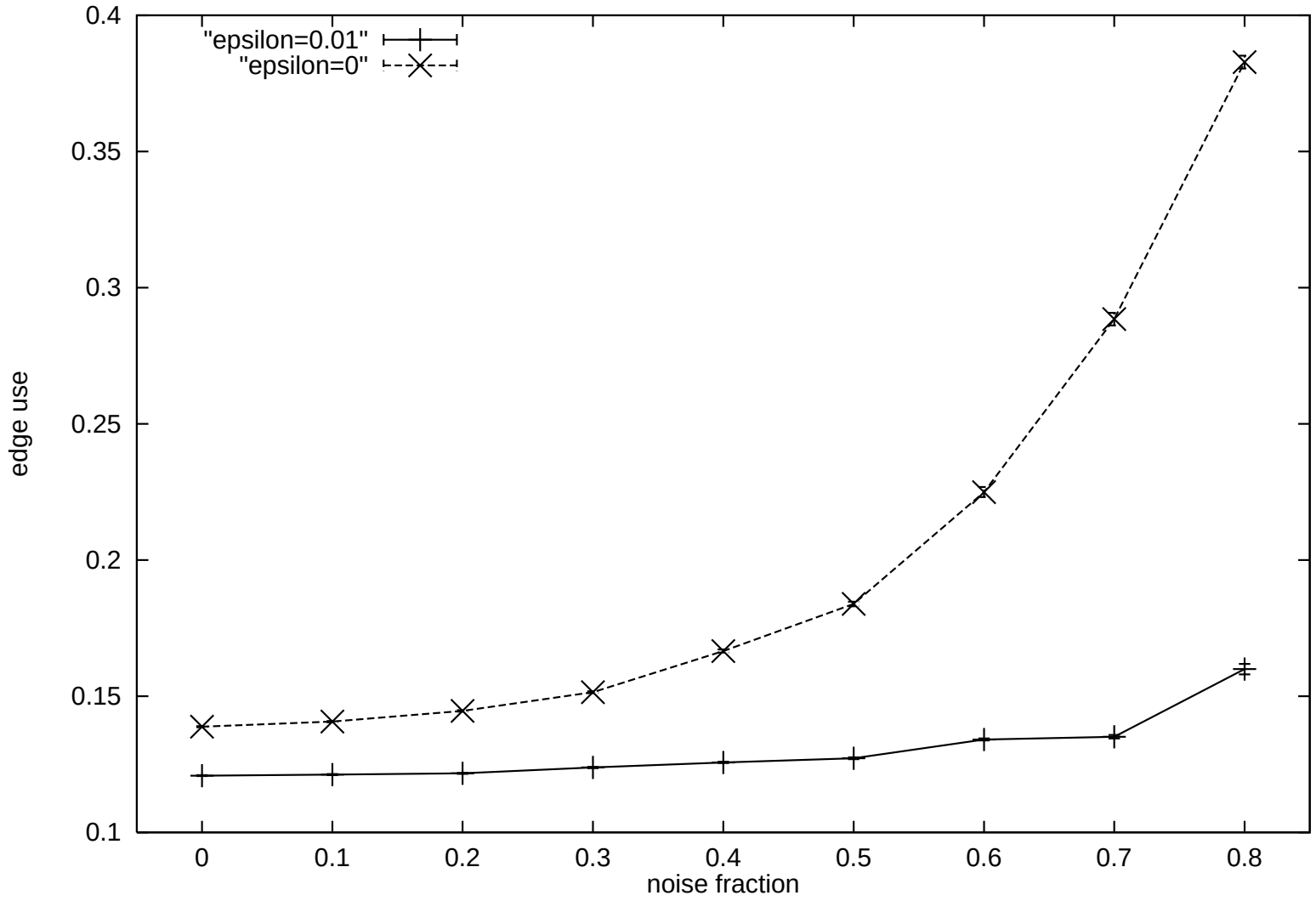


Theoretic best-case analysis

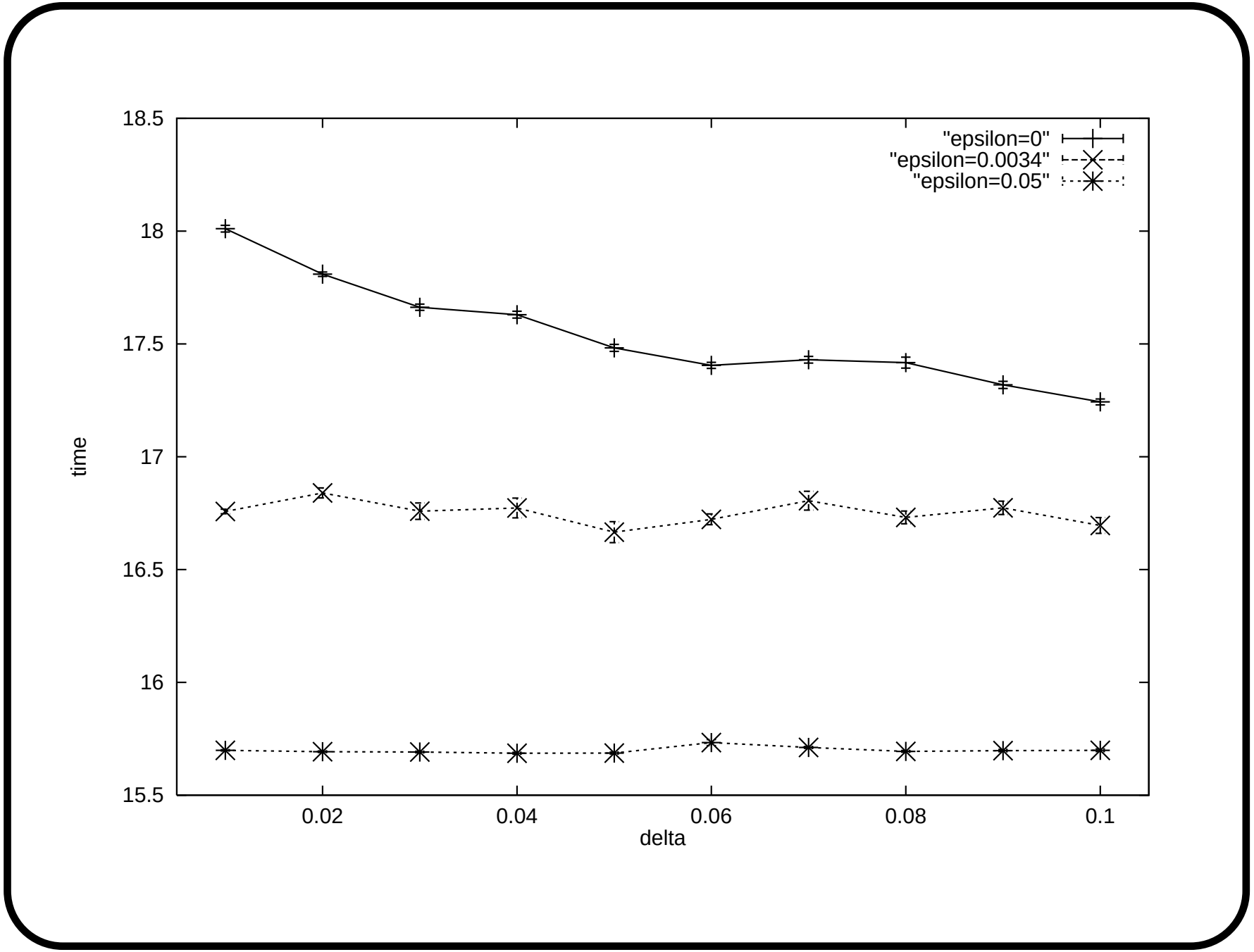
- Chow-Liu: $O(RN^2)$ to populate the weight matrix.
- Optimistic analysis for our algorithm:
  - The initial tree is the MST.
  - All the other edges are eliminated without looking at more data for them.
  - The only real cost is computing the edge weights for the tree edges.
  - $O(RN)$
- A best-case of N-fold speedup.
- Worst case: all edges become tree edges, no speedup.

© Dan Pelleg

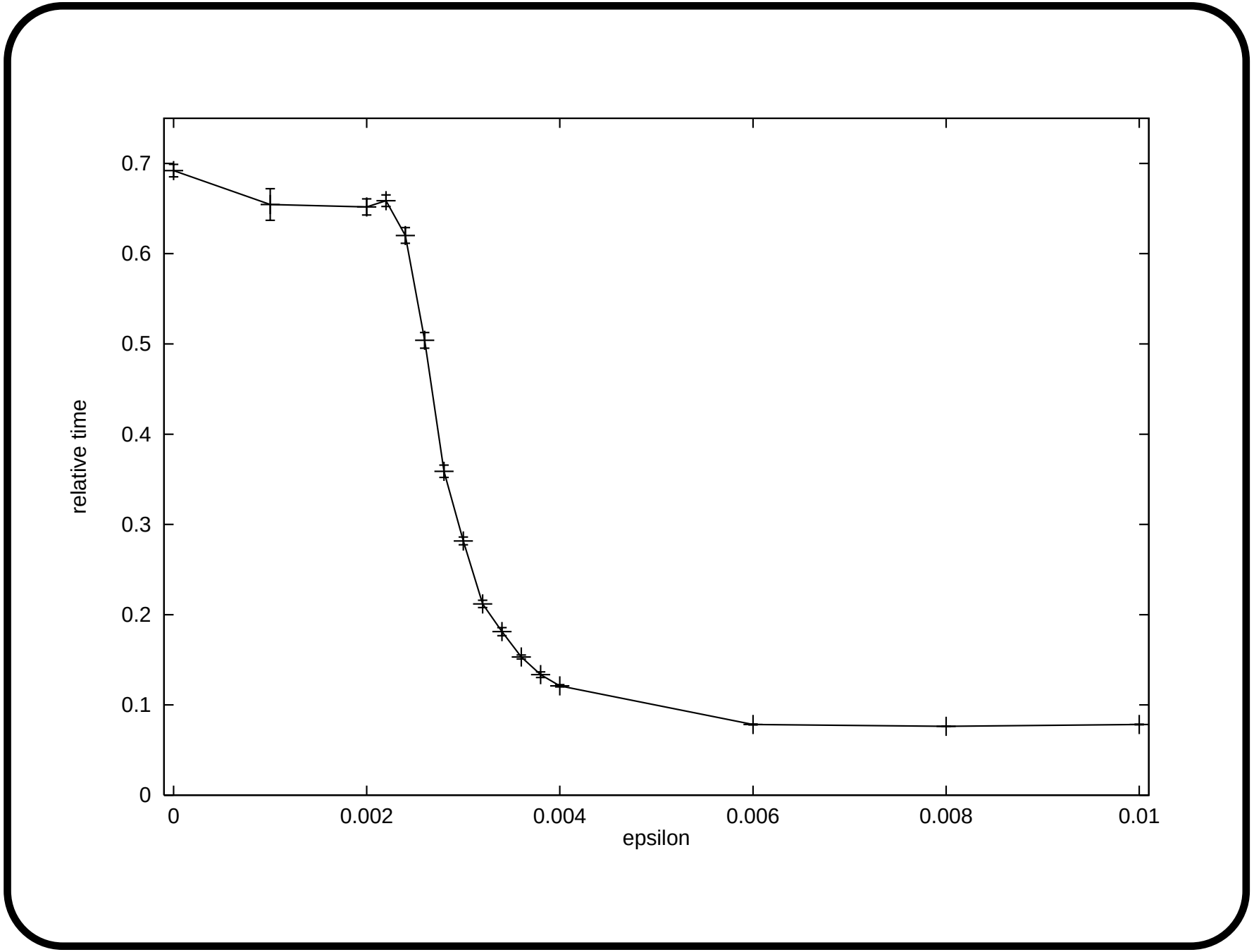
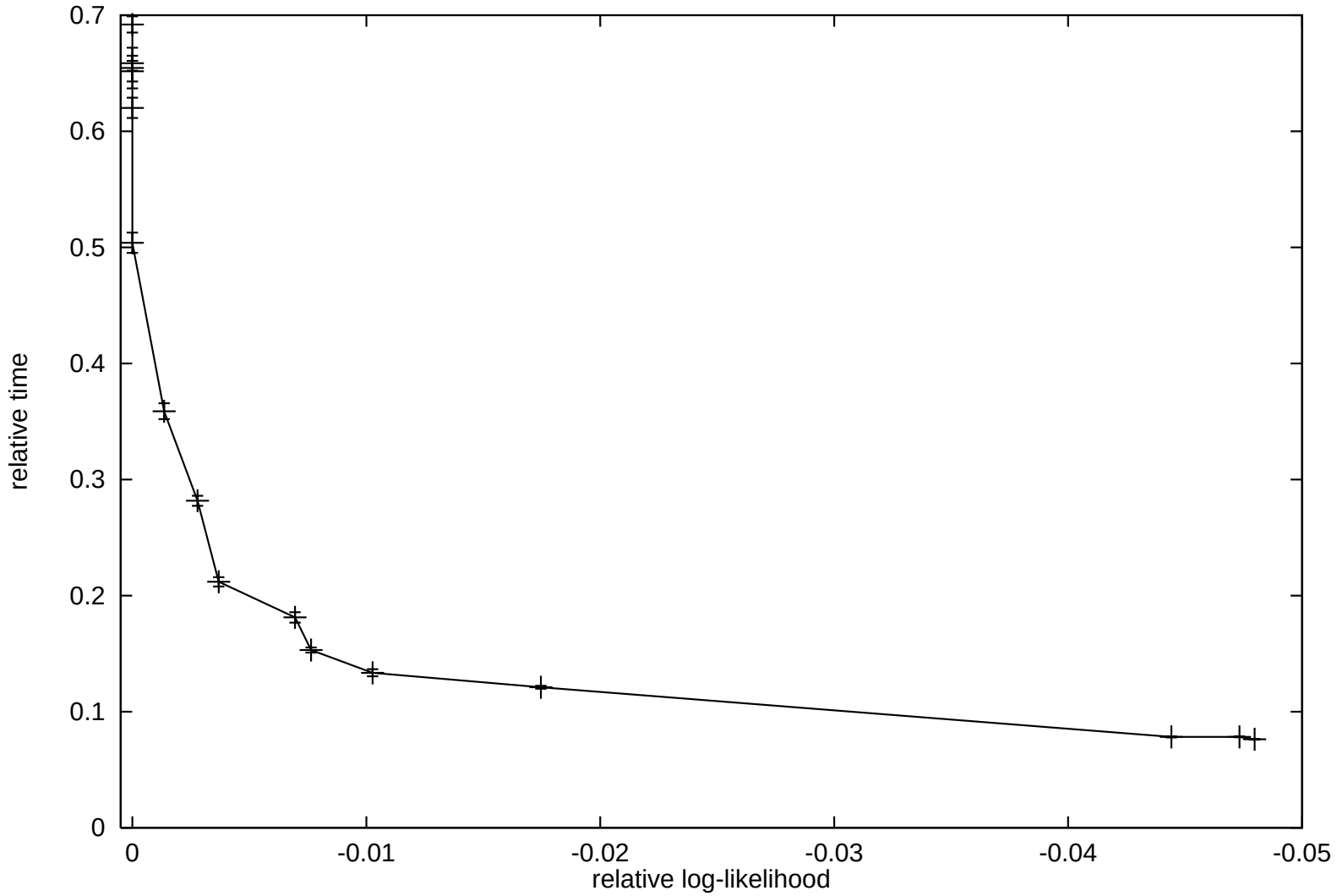
Resilience to noise



Effect of δ and ϵ



Effect of ϵ



Quality/speed tradeoff