Dagstuhl talk, *Efficient Cluster Compensation For Lin-Kernighan Heuristics*, David Neto, June 2002.

Efficient Cluster Compensation for Lin-Kernighan Heuristics

David M. Neto
University of Toronto
(currently Altera Corporation)

June 2002

Outline

- Lin-Kernighan heuristic
- Why clustered data is difficult
- Efficient cluster compensation
- Computing cluster distance
- Experiments for the TSP
- Summary

Lin-Kernighan heuristic

Heuristics are critical to many best practice implementations for combinatorial optimization.

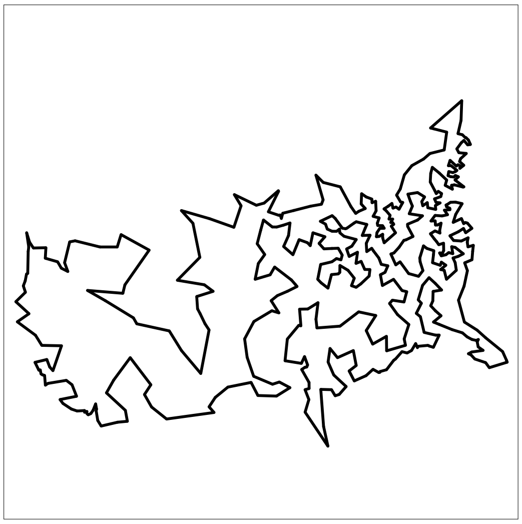
Heuristics exhibit complex behaviour.

Lin-Kernighan heuristic is a “champion” heuristic for the Traveling Salesman Problem.

Two concrete problems (1)

Traveling Salesman Problem

- Complete undirected weighted $G = (V, E, \omega)$. Find a tour of minimum weight.

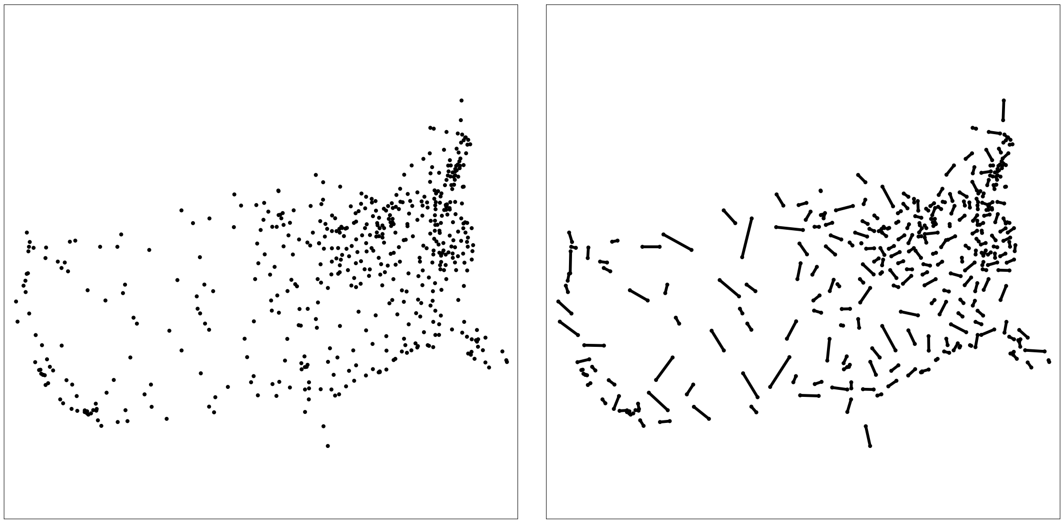


- Related decision problem is NP-complete.

Two concrete problems (2)

Minimum weight perfect matching

- Undirected weighted graph $G = (V, E, \omega)$.

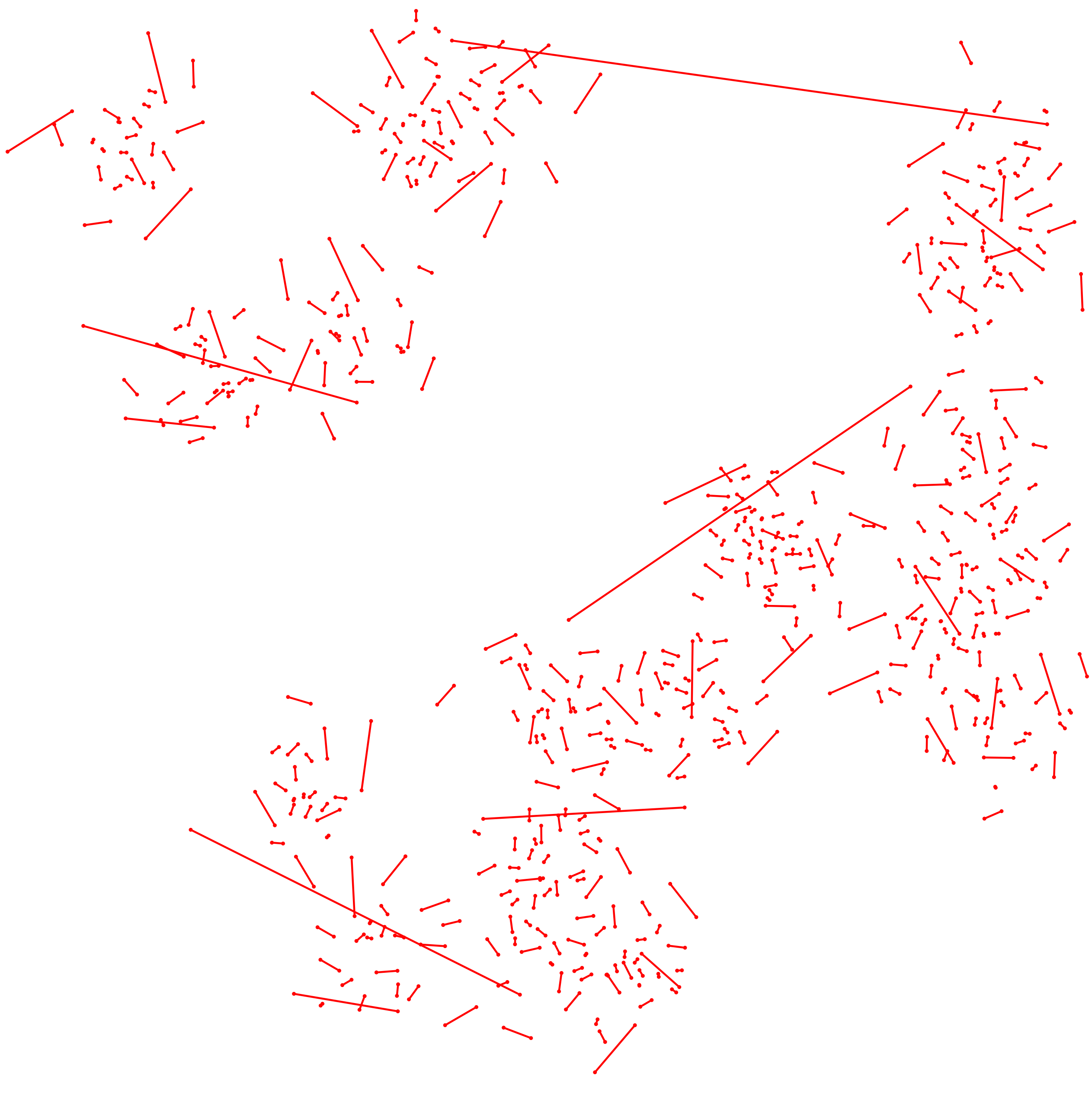


- General weighted matching solvable in polynomial time.

- We consider the case where G is complete.

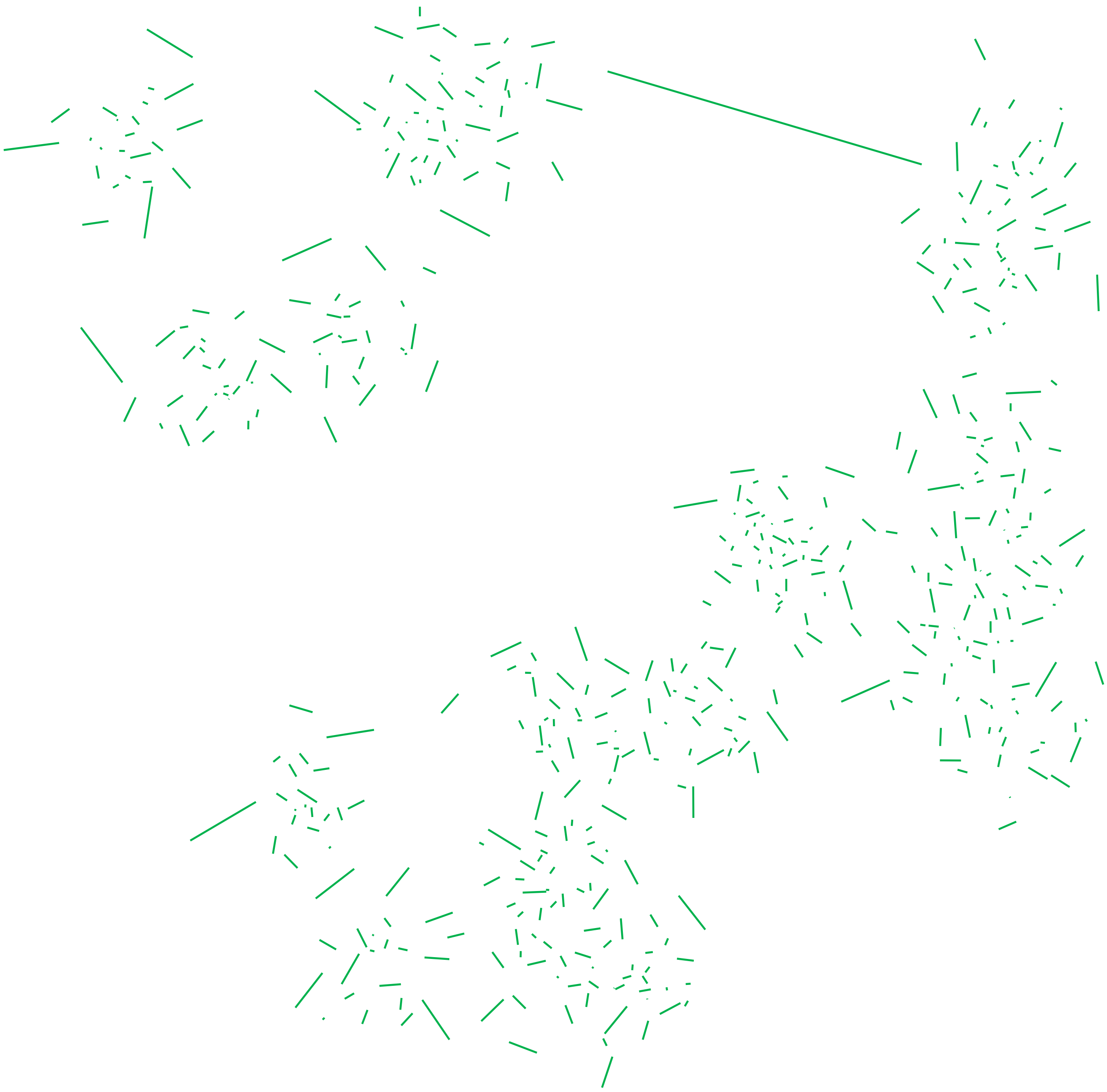
Randomized Greedy matching, weight 10672634

Clustered random problem (Johnson)



LK-opt matching, weight 8204145

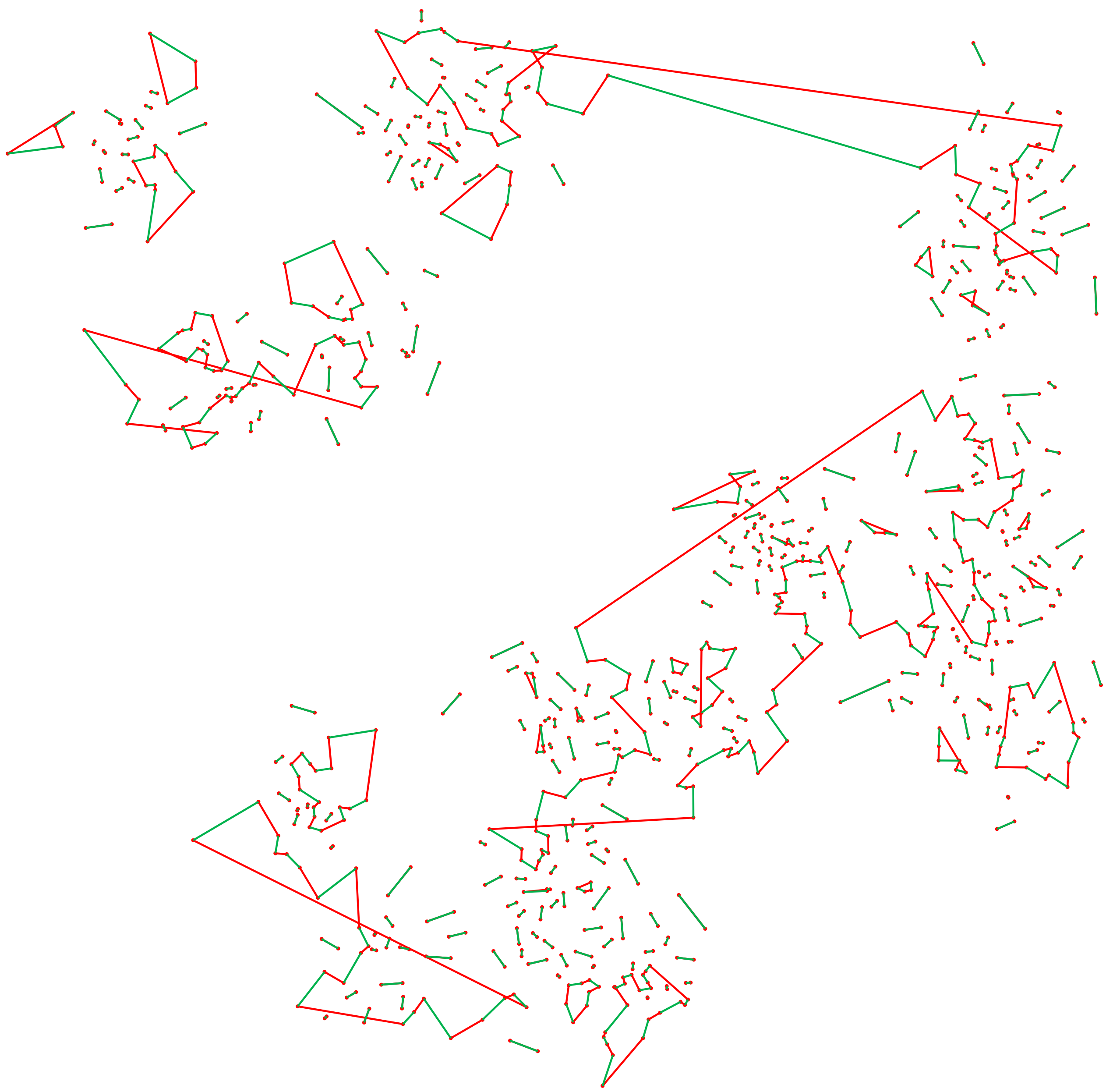
Clustered random problem (Johnson)



Randomized Greedy matching, weight 10672634

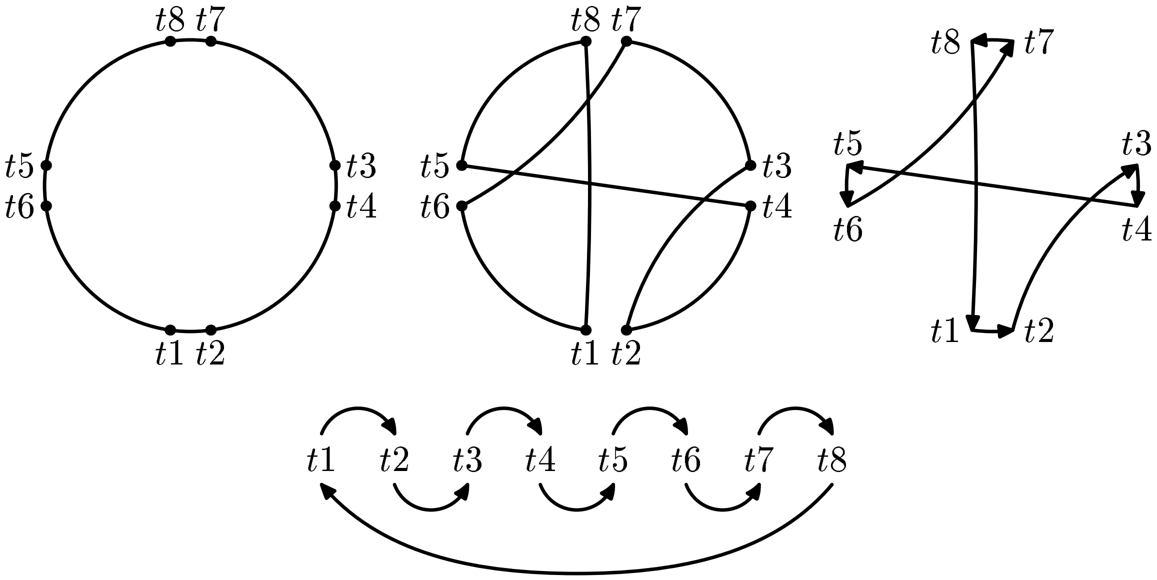
Lk-optimized matching, weight 8204145

Clustered random problem (Johnson)



Lin-Kernighan local search

Difference between one solution the next is an *alternating* cycle. Removed edges outweigh added edges.



Inner loop constructs a search sequence $t_1, t_2, t_3, t_4, \ldots$

Maintain *cumulative gain criterion*:

$$0 < \mathit{cum}(2i) = \sum_{j \text{ odd}} \omega(t_j, t_{j+1}) - \sum_{j \text{ even}} \omega(t_j, t_{j+1})$$

Net gain at t_{2i} is $\mathit{cum}(2i) - \omega(t_{2i}, t_1)$.

Clustered data is difficult

No accounting for future closing up costs, $\omega(t_{2i}, t_1)$ for larger i.

Removing a long edge greatly increases cumulative gain; search sequences lengthen.

This is too optimistic if removed edge spans a large inter-cluster gap; eventually pay back the large gain on the return trip to t_1.

Idea: Make cumulative gain criterion more pessimistic; incorporate cluster structure in a lookahead term.

Cluster compensation

For path P, let

$$\textit{bottleneck}(P) = \max_{e \in P}\{\omega(e)\}.$$

Cluster distance in G from u to v:

$$c_G(u,v) = \min_{P \text{ from } u \text{ to } v}\{\textit{bottleneck}(P)\}.$$

$c_G(t_{2i}, t_1)$ is our *guess* for a lower bound on future closing-up costs.

Change cumulative gain criterion from:

$$0 < \textit{cum}(2i)$$

to:

$$0 < \textit{cum}(2i) - c_G(t_{2i}, t_1).$$

Computing cluster distance

Theorem: If T is a minimum spanning tree for G:

$$c_G = c_T.$$

Theorem: In $O(n \log n)$ time, construct binary tree *Top(T)* from T:

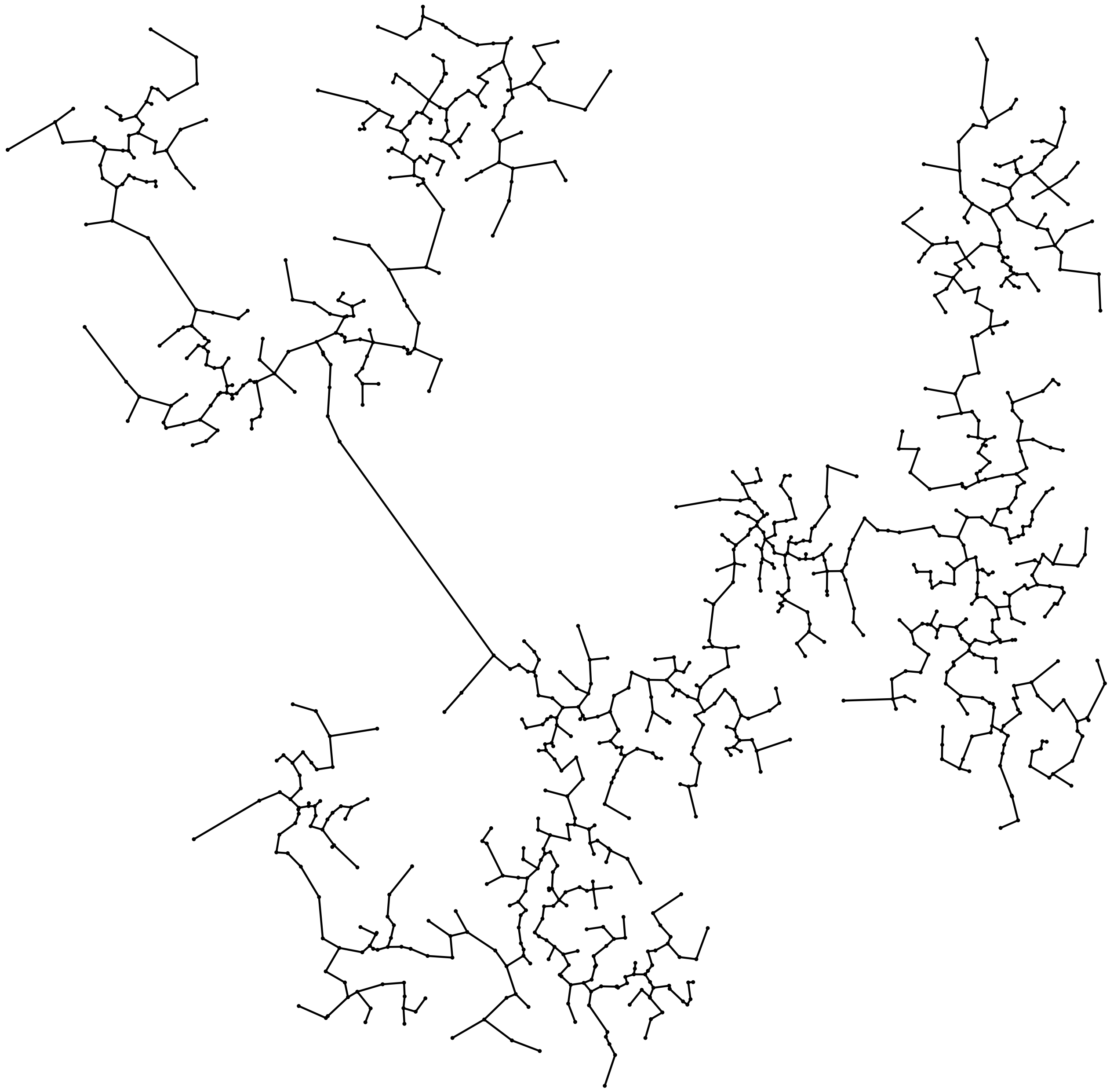
$$c_T(u, v) = w\left(\mathit{LCA}_{Top(T)}(u, v)\right).$$

With $O(n \log n)$ preprocessing, *LCA* in binary trees runs in $O(1)$ time.
[HT1984,SV1988,S1993]

Only $O(1)$ time extra overhead added to inner loop.

MST length (0*2^32+15905767.000000)

Clustered random problem (Johnson)



Experiments on DIMACS data

Excess over Held-Karp:

Instances	LK base %	LK cc %	ratio
TSPLIB	2.58	2.63	1.02
Unif.Euc.	1.88	1.92	1.02
Clus.Euc.	5.23	5.37	1.03
Rand.Matrix	3.67	3.62	0.98

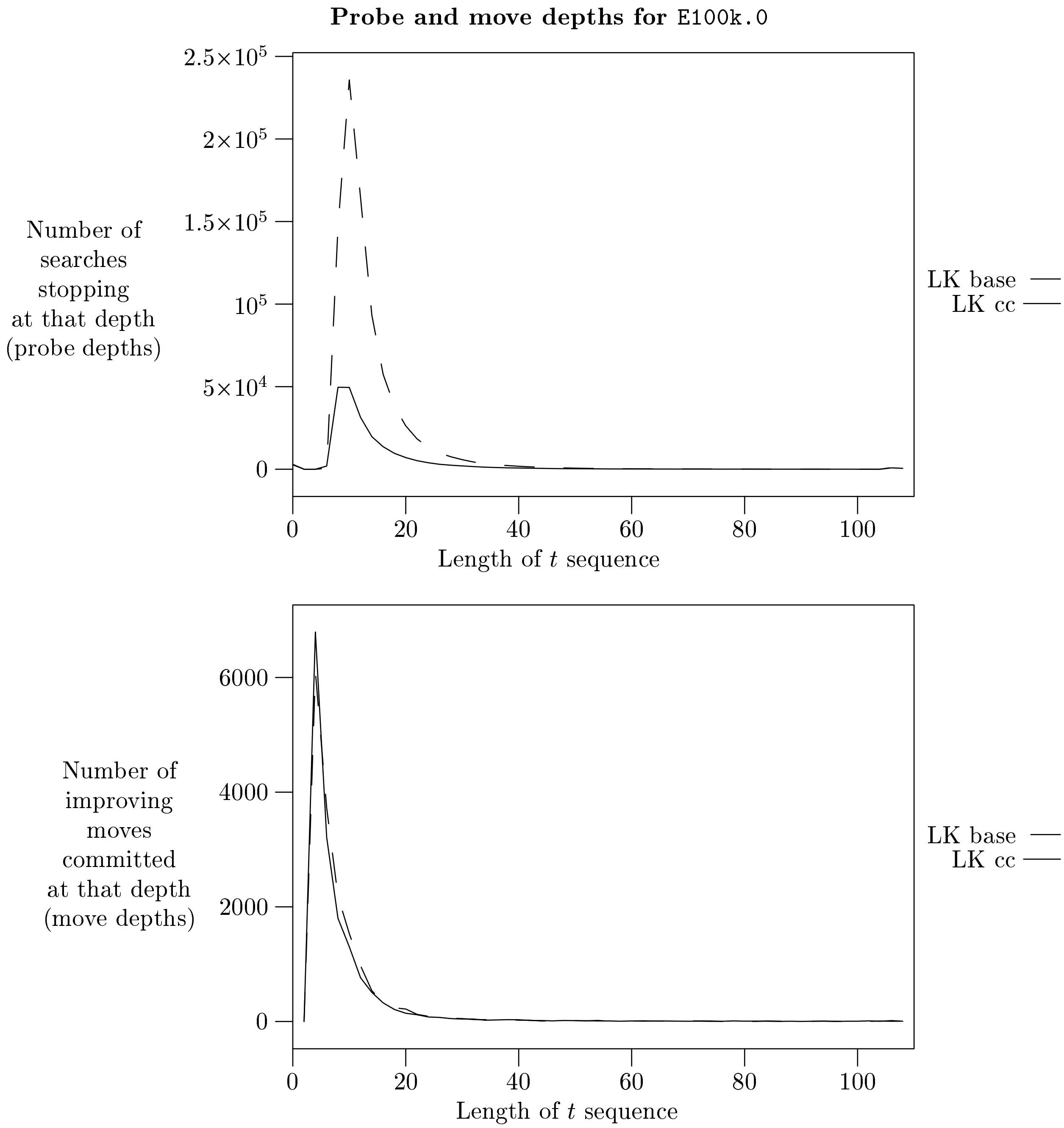
Running times:

Instances	LK base	LK cc	ratio
TSPLIB	4.2	2.5	0.60
Unif.Euc.	2.4	1.7	0.71
Clus.Euc.	51.2	20.7	0.41
Rand.Matrix	0.4	0.5	1.14

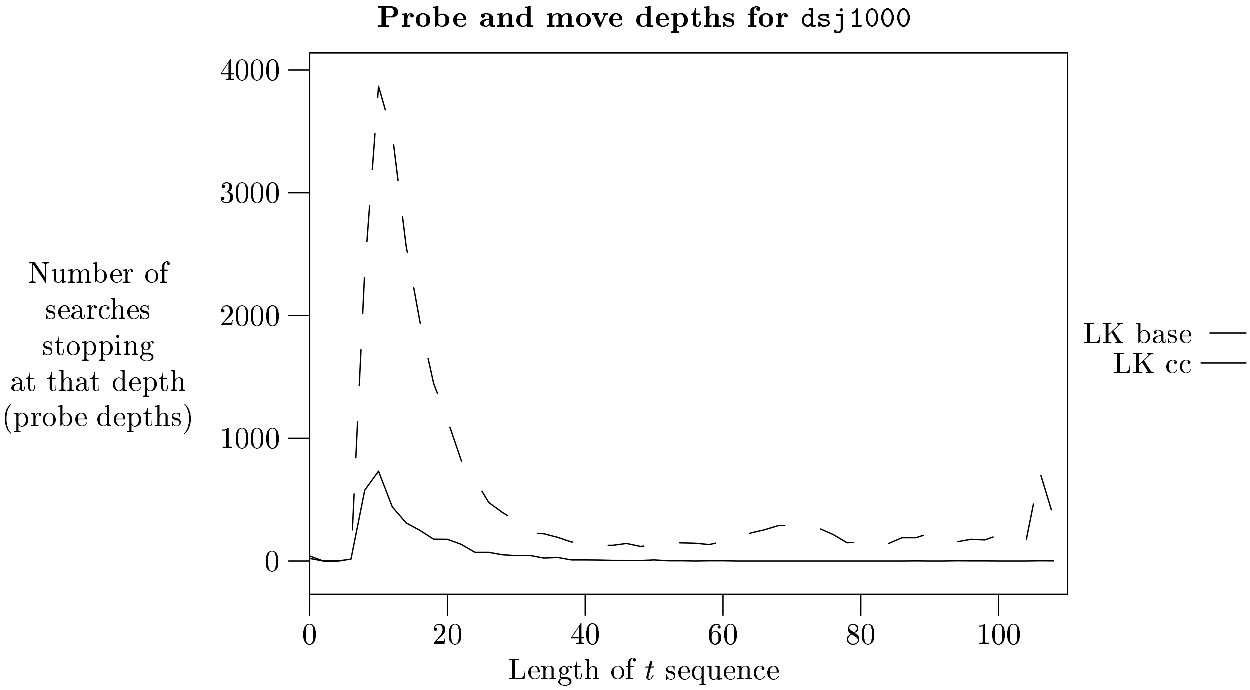
Cluster compensation saves time when:

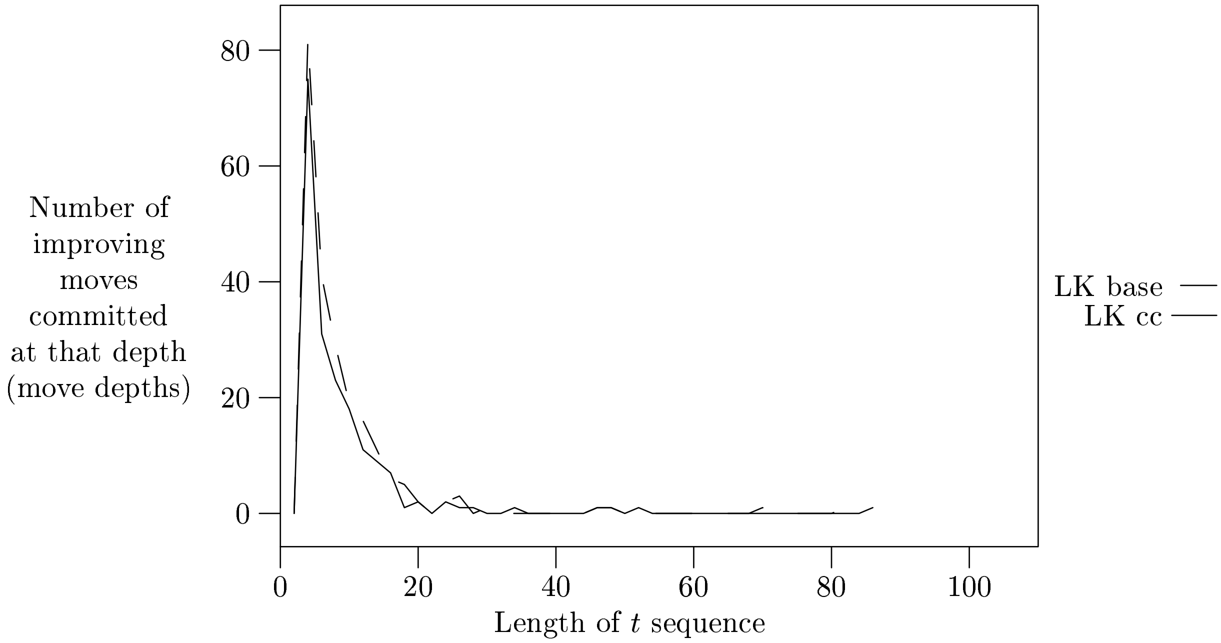
- computing MST is fast
- especially on sharply clustered inputs

Probe depths and move depths: E100k.0



Probe depths and move depths: dsj1000





Summary

- Intuition about why clustered data increases runtimes for Lin-Kernighan.
- Introduced efficient cluster compensation.
- Show that it reduces runtimes for most instance classes
  - the more clustered the better
  - as long as computing MST is fast
- Improvement shown in metrics independent of machine and programmer (probe and move depths)

Further work

- Still a performance gap depending on severity of clustering.

- Lin-Kernighan for maximization problems.

- More experiments!

PLS-completeness

PLS captures problems solved by "reasonable" local search algorithms.

Papadimitriou proved (stylized) Lin-Kernighan for TSP solves a PLS-complete problem.

We adapted his construction to show Lin-Kernighan with cluster compensation for TSP also solves a PLS-complete problem.