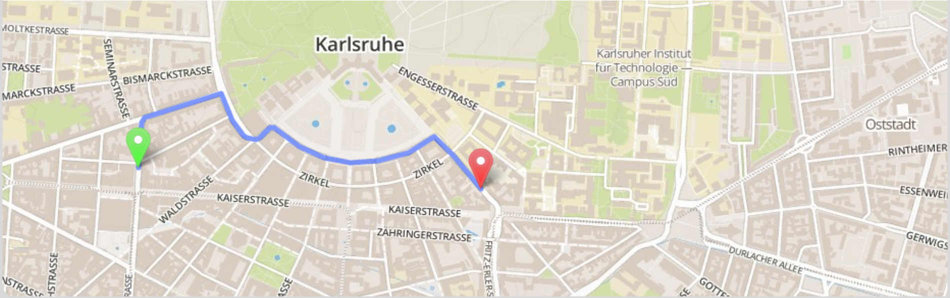
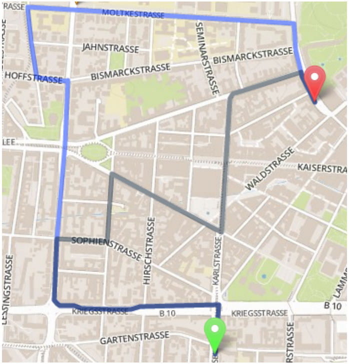
Engineering Parallel Bi-Criteria Shortest Path Search

Stephan Erb – *stephan.erb@student.kit.edu*

Institute of Theoretical Informatics

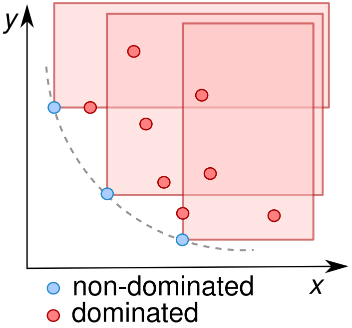


The Problem



Simultaneous optimization of **multiple** (possibly) conflicting **objectives**
(e.g., travel time and highway tolls)

Bi-Criteria Shortest Path Problem

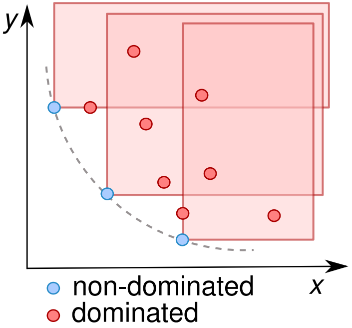


Graph $G = (V, E)$ with 2-dimensional edge weights $\vec{w} = (x, y)$

Pareto optimal path are non-dominated

A path p is dominated by a path q iff $(q \neq p) \wedge (q_x \leq p_x) \wedge (q_y \leq p_y)$

Bi-Criteria Shortest Path Problem



Graph $G = (V, E)$ with 2-dimensional edge weights $\vec{w} = (x, y)$

Pareto optimal path are non-dominated

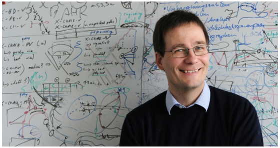
A path p is dominated by a path q iff $(q \neq p) \wedge (q_x \leq p_x) \wedge (q_y \leq p_y)$

Goal: Find all Pareto optimal paths from a start node to all other nodes **(one-to-all)**

Stephan Erb:
Engineering Parallel Bi-Criteria Shortest Path Search

Department of Informatics
Institute of Theoretical Informatics - Algorithms II

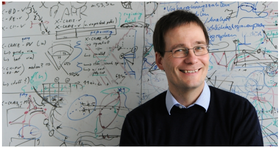
The Parallel Pareto Search Algorithm



Parallel Label-Setting Multi-Objective Shortest Path Search by P. Sanders and L. Mandow

IEEE International Parallel and Distributed Processing Symposium (IPDPS2013)

The Parallel Pareto Search Algorithm



Parallel Label-Setting Multi-Objective Shortest Path Search by P. Sanders and L. Mandow

IEEE International Parallel and Distributed Processing Symposium (IPDPS2013)

Provably efficient for two dimensions.
But is it also practical?

Dijkstra's Algorithm

Procedure DijkstraSearch(G,s)

$L[v] = \infty$ for all $v \in V$; $L[s] = 0$

PriorityQueue $Q = \{(s, 0)\}$ // *tentative labels*

while *Q is not empty* **do**

(u, l) = $Q.removeMinimum()$

foreach $(u, v) \in E$ **do**

$l' = l + weight((u, v))$ // *candidate label*

if l' *represents new best path to v* **then**

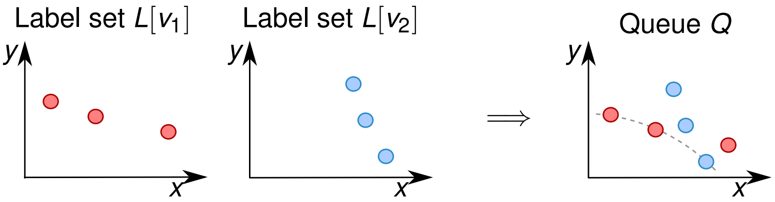
update $L[v]$

update Q

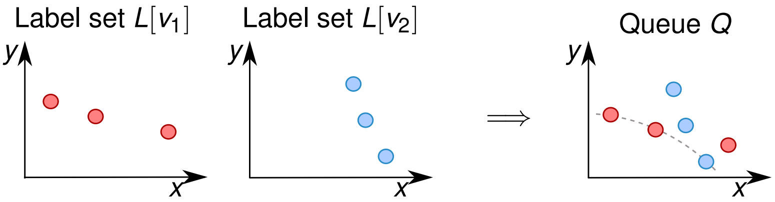
Label Setting Property:

Shortest path distance of node/labels extracted from Q is final

Bi-Criteria Label Setting



Bi-Criteria Label Setting



Classic Approaches [1]:

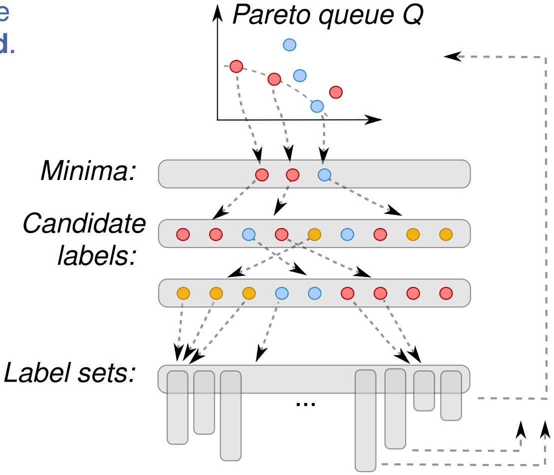
Extract **any** Pareto optimal label from Q

Sanders & Mandow:

Extract **all** Pareto optimal labels from Q

The Parallel Pareto Search Algorithm

All steps are **parallelized**.



Engineering Bulk Updates

Goal:

Apply sequence of k **sorted updates** (insertions & deletions)

Proposed Solution:

Parallel **red-black tree** of Frias and Singler [2]

Problem:

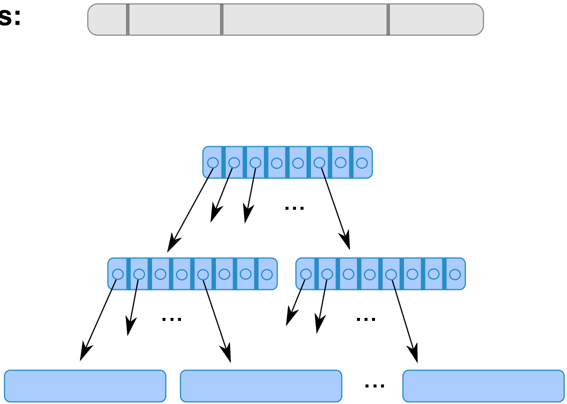
Binary trees can be multiple times slower than **cache-sensitive B-trees** [3]

Department of Informatics
Institute of Theoretical Informatics - Algorithms II

Parallel Bulk Updates for B-trees

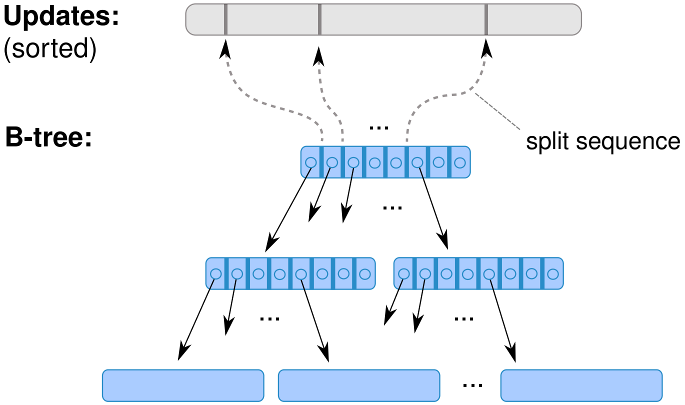
Updates:
(sorted)

B-tree:



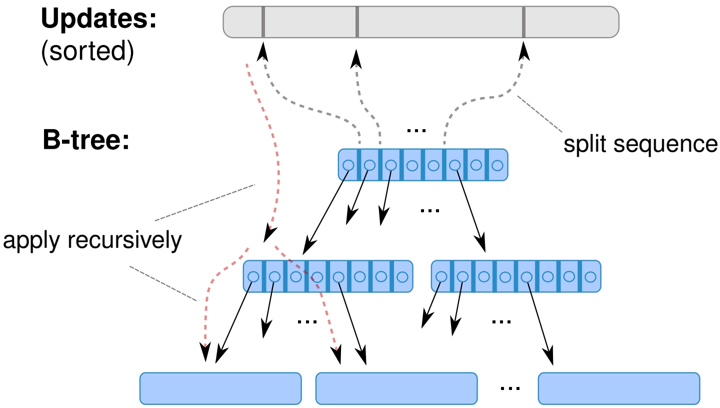
Department of Informatics
Institute of Theoretical Informatics - Algorithms II

Parallel Bulk Updates for B-trees



Department of Informatics
Institute of Theoretical Informatics - Algorithms II

Parallel Bulk Updates for B-trees

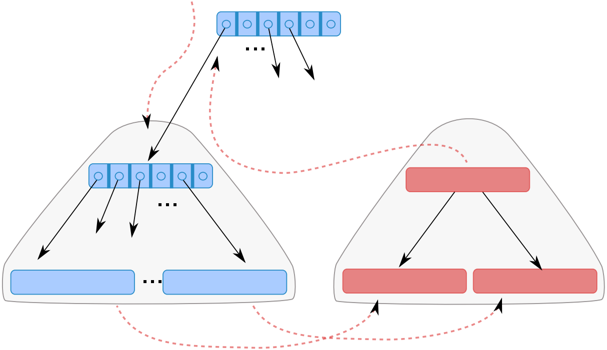


Subtrees updated **independently in parallel**.

Department of Informatics
Institute of Theoretical Informatics - Algorithms II

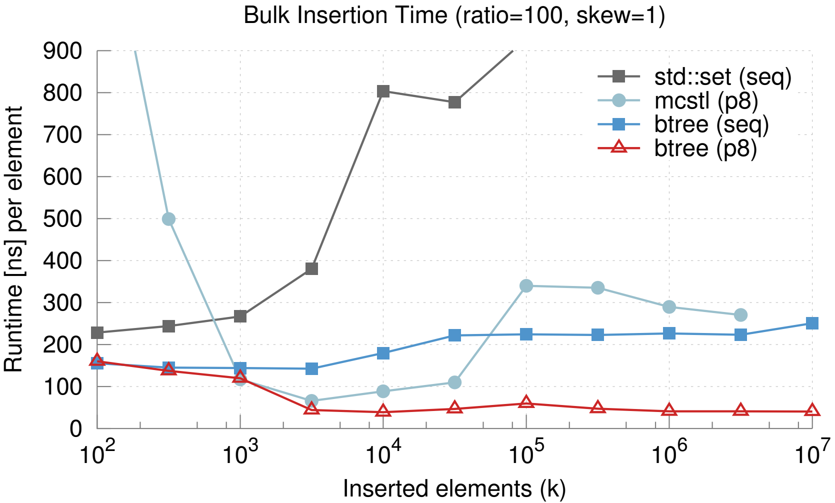
Parallel Rebalancing of B-trees

Parallel **partial-rebuilding** [4] of unbalanced subtrees:



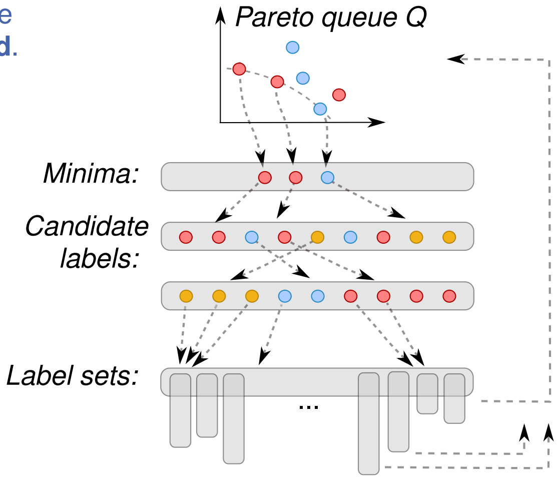
Amortized update bound of $\mathcal{O}(k/p \cdot \log N)$ for the application of k updates using p threads on a tree of size N.

Experimental Results



The Parallel Pareto Search Algorithm

All steps are **parallelized**.



Evaluation Setup

Karlsruhe Institute of Technology

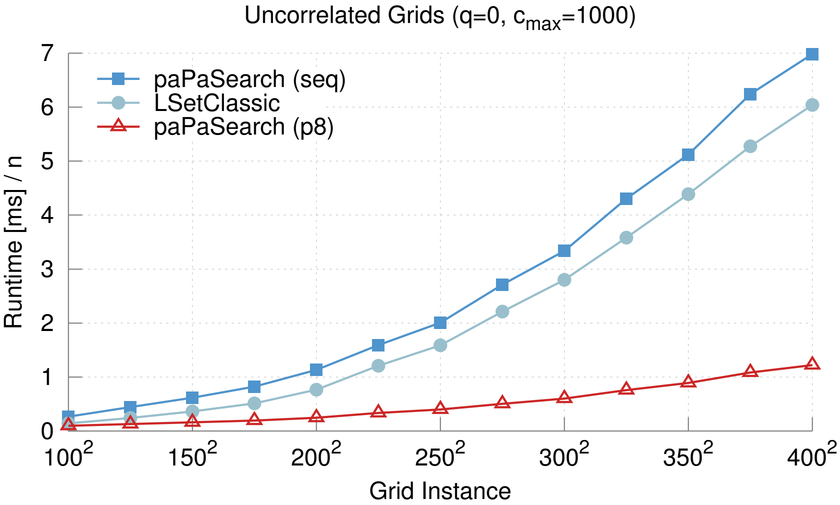
System:

- 2× Intel Xeon E5-2670 with **16 cores** in total, **2.6 GHz**, 2× 20MB L3 Cache
- 64 GB main memory

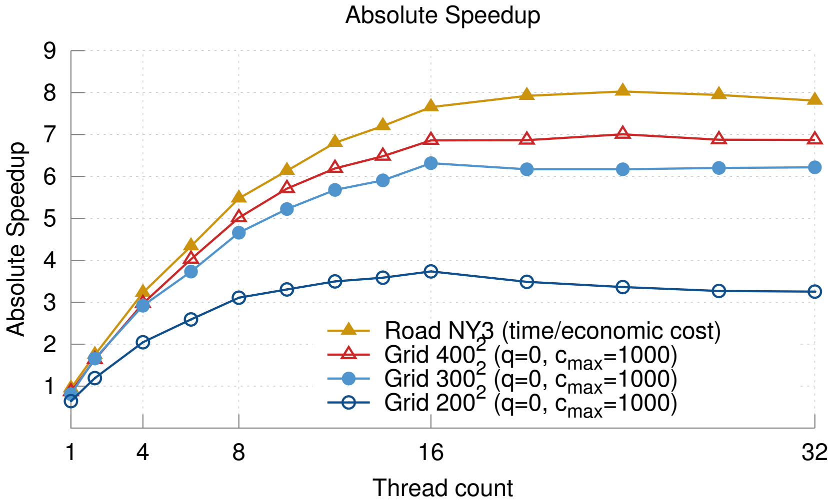
Tuned Competitor:

- highly tuned binary heap [5]
- fast dominance checks

Experimental Results

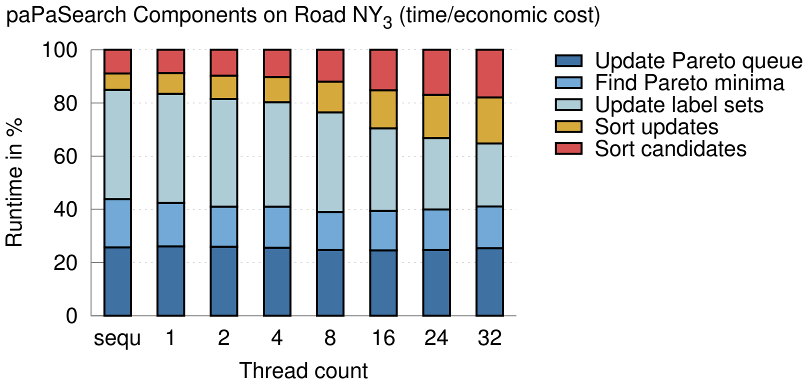


Experimental Results

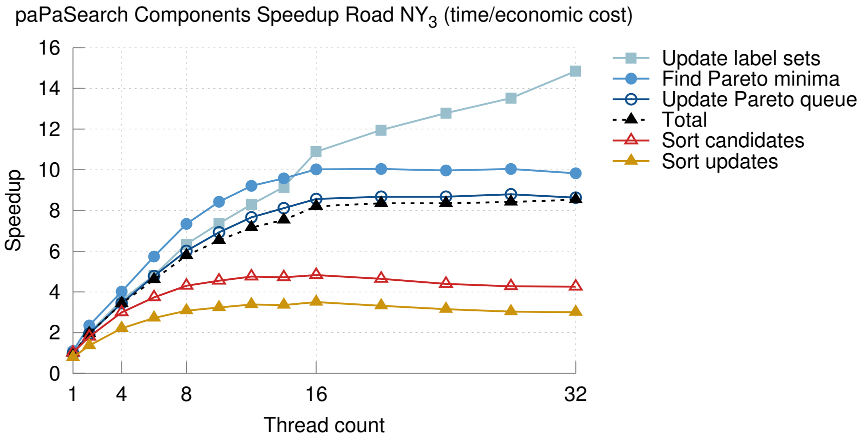


Stephan Erb:

Component Analysis

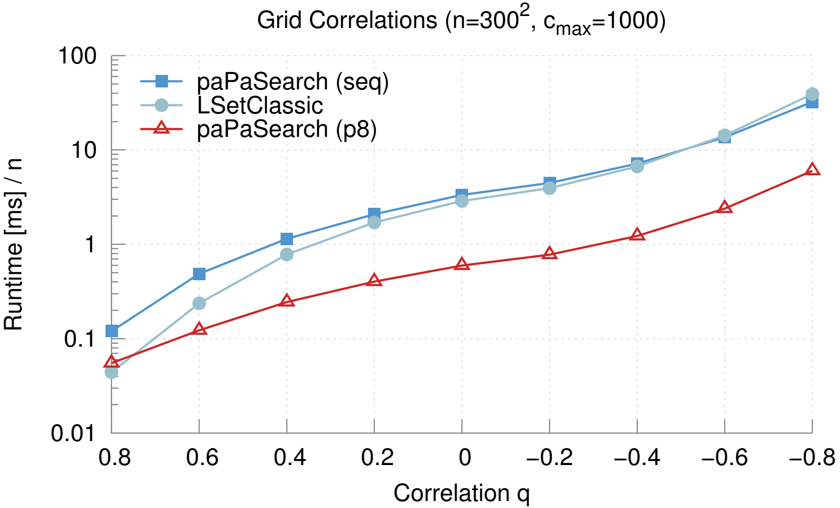


Component Analysis



Department of Informatics
Institute of Theoretical Informatics - Algorithms II

Experimental Results



Conclusion

Summary:

- Parallel label-setting is **practical** on modern **multiprocessors**
- It excels at **large & difficult instances**

Future Work:

- Delta-stepping [6] to increase work per iteration
- scalable sorting of small collections?
- Point-to-point search

Department of Informatics
Institute of Theoretical Informatics - Algorithms II

Thank You!

Results will be published on `http://algo2.iti.kit.edu`.

References I

[1] P. Hansen, “Bicriterion path problems,” in *Multiple Criteria Decision Making Theory and Application*, ser. Lecture Notes in Economics and Mathematical Systems. Springer Berlin Heidelberg, 1980, vol. 177, pp. 109–127. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-48782-8_9

[2] L. Frias and J. Singler, “Parallelization of bulk operations for stl dictionaries,” in *Euro-Par 2007 Workshops: Parallel Processing*, ser. Lecture Notes in Computer Science, L. Bougé, M. Forsell, J. Träff, A. Streit, W. Ziegler, M. Alexander, and S. Childs, Eds. Springer Berlin Heidelberg, 2008, vol. 4854, pp. 49–58. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78474-6_8

References II

[3] T. M. Chilimbi, M. D. Hill, and J. R. Larus, “Cache-conscious structure layout,” in *Proceedings of the ACM SIGPLAN 1999 conference on Programming language design and implementation*, ser. PLDI ’99. New York, NY, USA: ACM, 1999, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/301618.301633

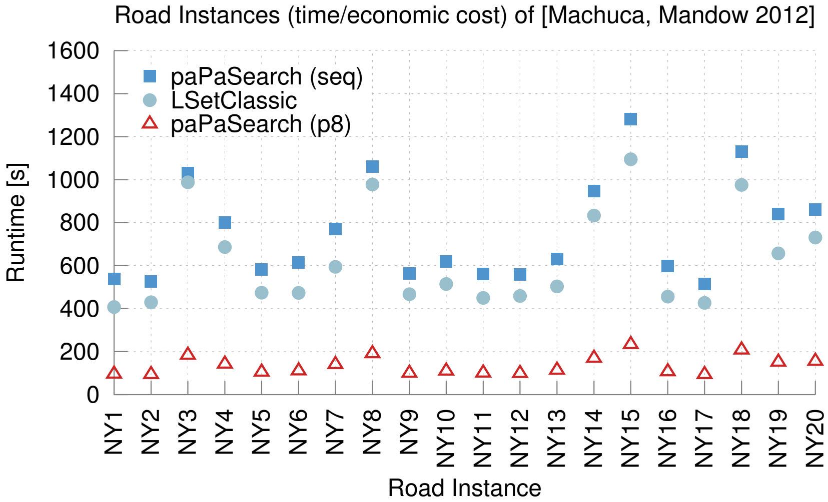
[4] M. Overmars, *The design of dynamic data structures*. Springer Verlag, 1983.

[5] P. Sanders, “Fast priority queues for cached memory,” *Journal of Experimental Algorithmics (JEA)*, vol. 5, Dec. 2000. [Online]. Available: http://doi.acm.org/10.1145/351827.384249

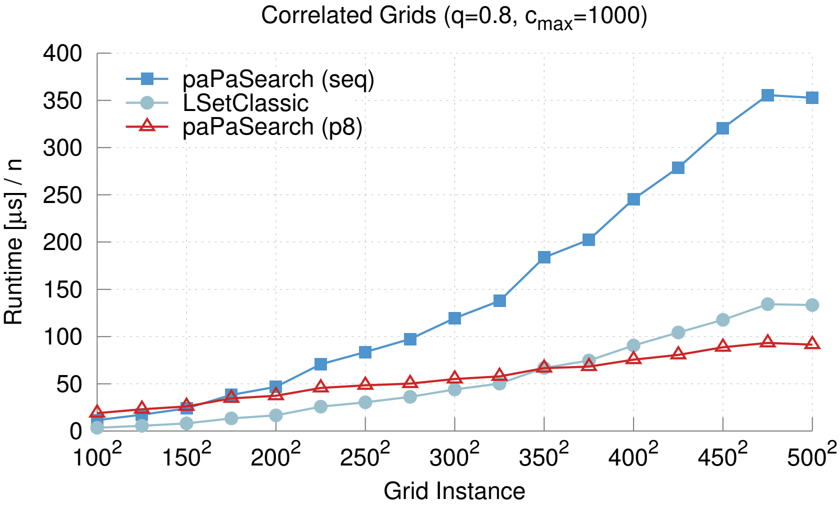
[6] U. Meyer and P. Sanders, “δ-stepping: a parallelizable shortest path algorithm,” *Journal of Algorithms*, vol. 49, no. 1, pp. 114 – 152, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0196677403000762

Backup

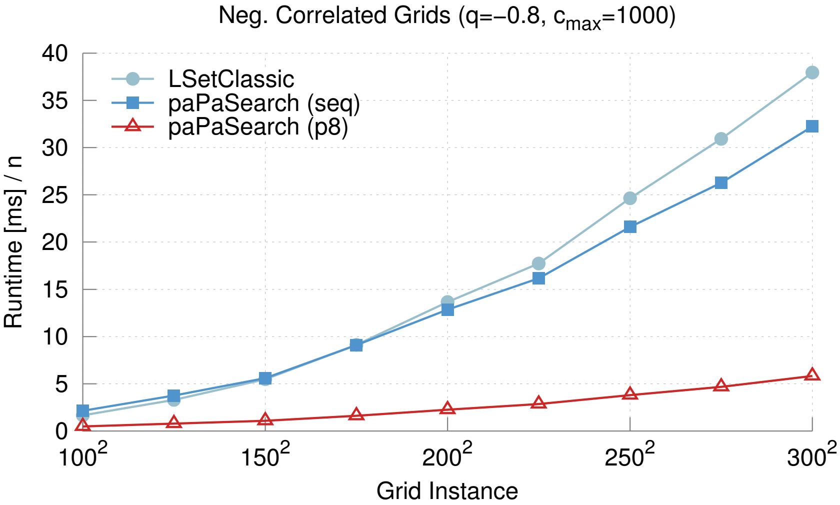
Experimental Results



Experimental Results

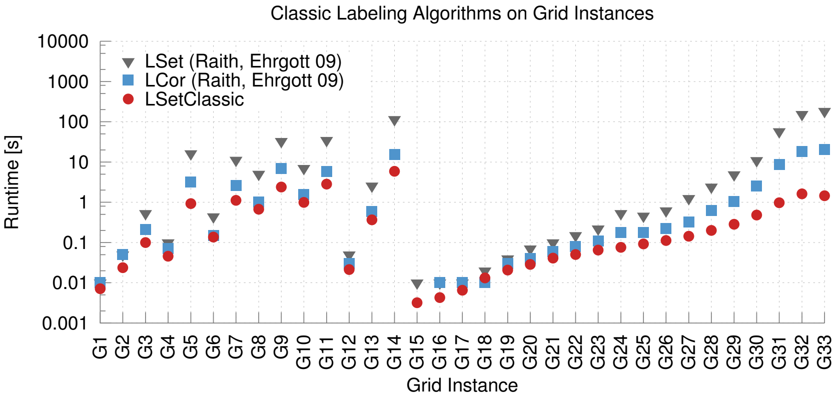


Experimental Results

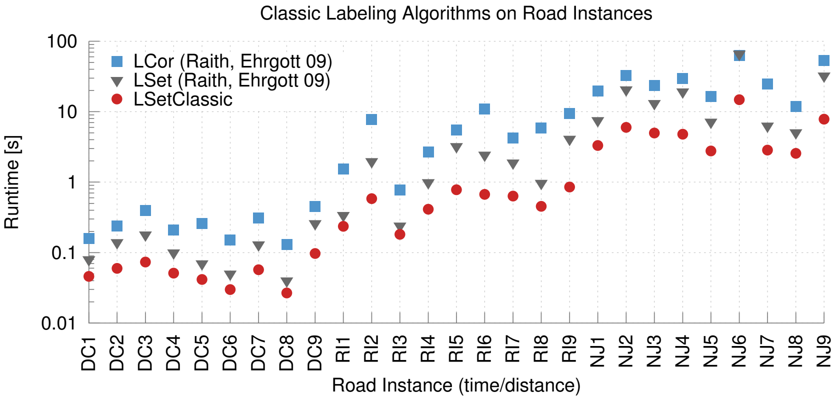


Department of Informatics
Institute of Theoretical Informatics - Algorithms II

Classic Competitor

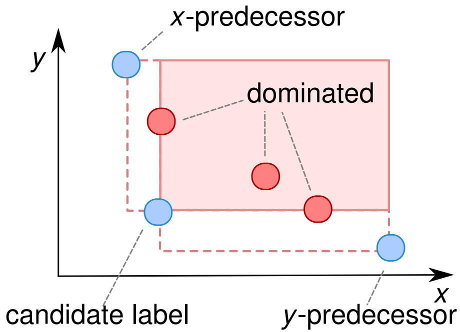


Classic Competitor



Dominance checks

Fast dominance checks on lexicographically sorted label sets:



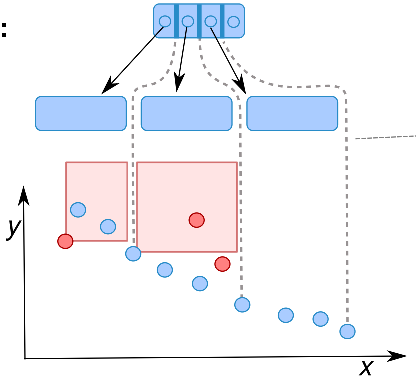
Predecessor queries as **tree traversals**.
Running in $\mathcal{O}(\log N)$ for a label set of size N.

Tuned Dominance checks

Batched dominance check & candidates filtering:

B-tree Label Set:

Logical View:



Router keys are copies of the largest label in each subtree.

Department of Informatics
Institute of Theoretical Informatics - Algorithms II