Goal Directed Shortest Path Queries Using Precomputed Cluster Distances[1]

Jens Maue[1] Peter Sanders[2] Domagoj Matijevic[1]

Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany,
[jensmaue,dmatijev]@mpi-inf.mpg.de

Universität Karlsruhe, 76128 Karlsruhe, Germany, sanders@ira.uka.de

27 May 2006

[1]Partially supported by DFG grant SA 933/1-2.

Motivation

Shortest Route between Two Points in a Network

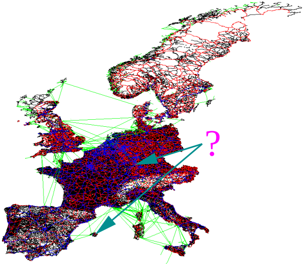
Real-world applications:

- Car navigation systems
- Timetable information services

Dijkstra's algorithm: too slow!

Idea: perform preprocessing to accelerate queries

Precompute APSP: too expensive!



Goals

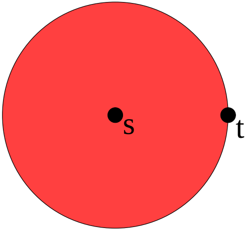
- Exact single source single target shortest path queries
- Fast preprocessing
- Fast queries
- Low extra space requirement
- Flexible trade-off between these objectives

PCD — basic idea:

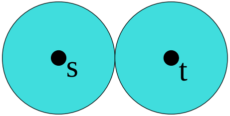
- Partition input graph into k clusters
- Precompute some shortest path distances:
  Distances between all pairs of clusters
- Query: prune nodes far away from shortest path

Related Work

Dijkstra's algorithm [dijkstra:59]

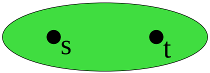


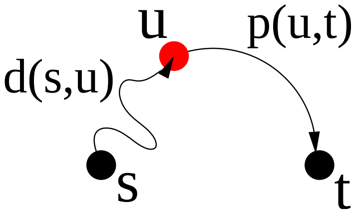
Bidirectional search [pohl:71]



Related Work

Goal-directed search





Select node with smallest $\underline{d}(s,u,t) = d(s,u) + p(u,t)$

- Original A^* [pohl:71]
    - Euclidean distance
    - Bad performance for fastest queries
    - Layout needed
- Landmark-A^* [goldberg:harrelson:05]
    - Preprocess distances to selected landmarks
    - $d(u,t) \geq d(L,t) - d(L,u) = p(u,t)$
    - Superlinear space

Related Work

Graph hierarchy

- Highway hierarchies [sanders:schultes:05]
- Reach-based pruning [gutman:04]
- Seperator-based graph decomposition [köhler:möhring:schilling:05, . . .]

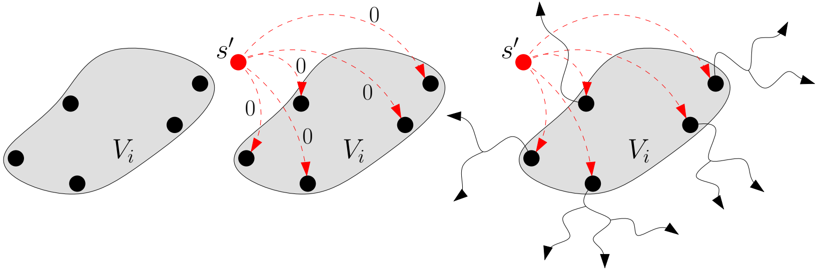
Other heuristics

- Geometric containers [wagner:willhalm:03]
- Edge flags [lauther:04, köhler:möhring:schilling:05]

Combinations [goldberg:kaplan:werneck:06, holzer:schulz:willhalm:04]

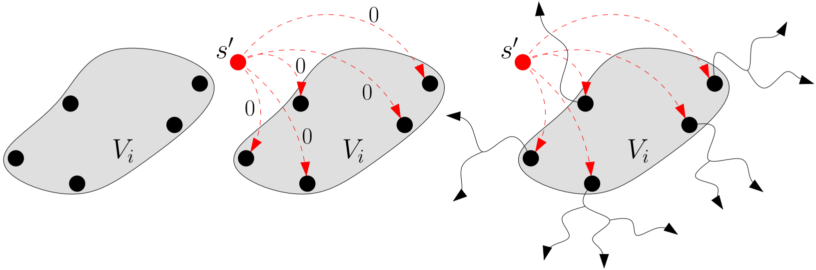
Preprocessing

- Assume graph partitioned into k clusters
- $d(U, V) = \min_{u \in U, v \in V} d(u, v)$
- Compute k^2 distances between clusters



Preprocessing

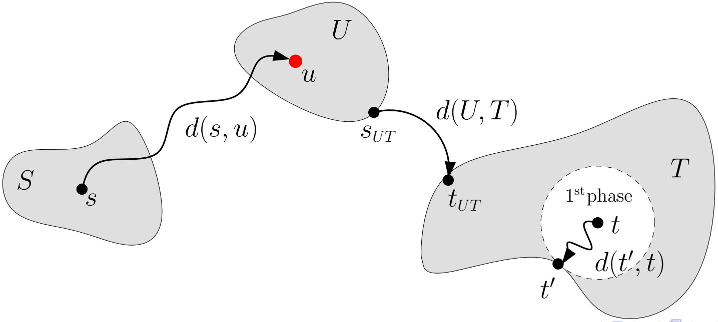
- Assume graph partitioned into k clusters
- $d(U, V) = \min_{u \in U, v \in V} d(u, v)$
- Compute k^2 distances between clusters



- Preprocessing time $\Theta(k \cdot D(n))$
- Additional space $\Theta(B + k^2)$ sublinear

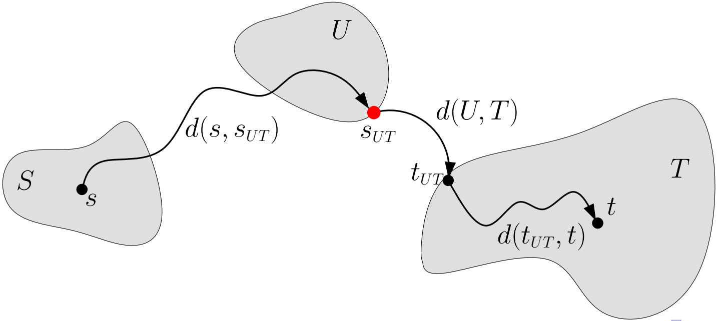
Query

- Modification of (bidirectional) Dijkstra's algorithm
- Repeatedly compute lower bounds $\underline{d}(s, u, t)$



Query

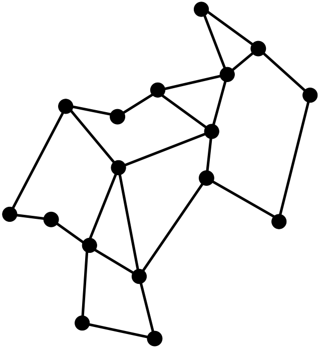
- Modification of (bidirectional) Dijkstra's algorithm
- Repeatedly compute lower bounds $\underline{d}(s, u, t)$
- Maintain upper bound $\hat{d}(s, t)$ on total distance
- Prune node u if $\underline{d}(s, u, t) > \hat{d}(s, t)$
- Robustness: nodes on shortest path are never pruned



Partitioning

k-center clustering:

- k center nodes $c_1, \ldots, c_k$
- assign node to closest center: $u \in V_i \Leftrightarrow d(u, c_i) \leq d(u, c_j)$
- calculate clustering by one shortest paths search



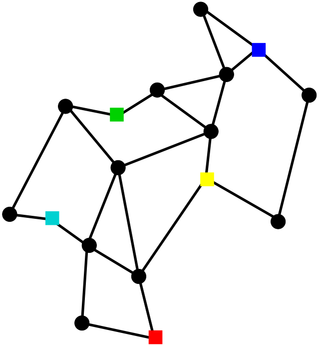
k′-oversampling:

- k'-center clustering, $k' > k$
- remove adverse center
- repeat until k centers left

Partitioning

k-center clustering:

- k center nodes $c_1, \ldots, c_k$
- assign node to closest center: $u \in V_i \Leftrightarrow d(u, c_i) \leq d(u, c_j)$
- calculate clustering by one shortest paths search



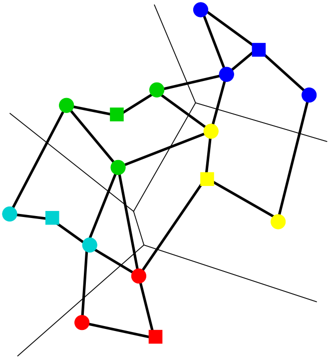
k′-oversampling:

- k'-center clustering, $k' > k$
- remove adverse center
- repeat until k centers left

Partitioning

k-center clustering:

- k center nodes $c_1, \ldots, c_k$
- assign node to closest center: $u \in V_i \Leftrightarrow d(u, c_i) \leq d(u, c_j)$
- calculate clustering by one shortest paths search

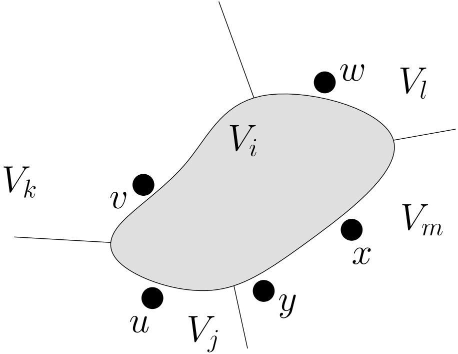


k′-oversampling:

- k'-center clustering, $k' > k$
- remove adverse center
- repeat until k centers left

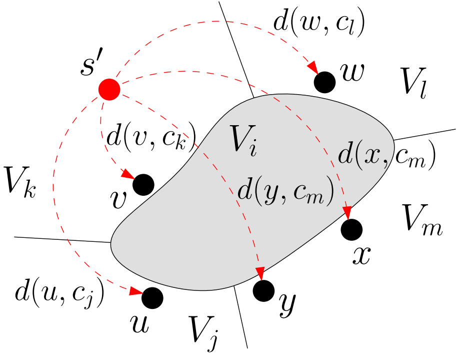
Partitioning

Cluster deletion:



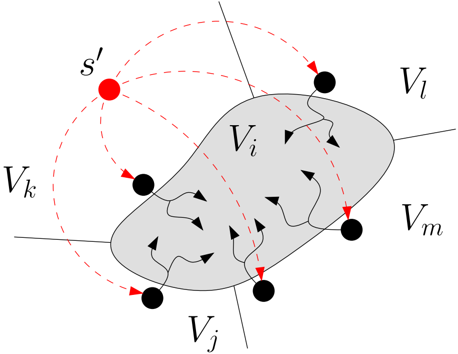
- Redistribute nodes
- Preserves $(k'-1)$-center properties
- Deletion time $D(s(V_i))$

Cluster deletion:



- Redistribute nodes
- Preserves $(k' - 1)$-center properties
- Deletion time $D(s(V_i))$

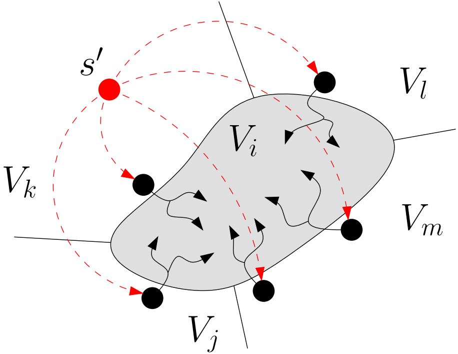
Cluster deletion:



- Redistribute nodes
- Preserves $(k'-1)$-center properties
- Deletion time $D(s(V_i))$

Partitioning

Cluster deletion:



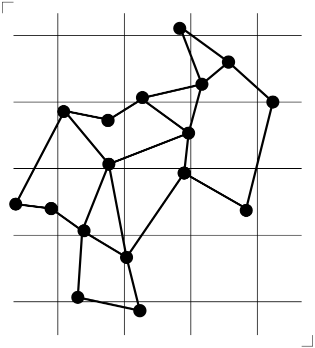
- Redistribute nodes
- Preserves $(k'-1)$-center properties
- Deletion time $D(s(V_i))$

Selection heuristics:

- MinSize: smallest size
- MinRad: smallest radius
- MinSizeRad: alternate between both

Partitioning

- Grid clustering



- Metis
  - Equally-sized clusters
  - Minimize edge cut
  - Non-contiguous clusters possible
- 'Oversampling' $= (k \log k)$-oversampling with MinSize heuristic

Experimental Setup

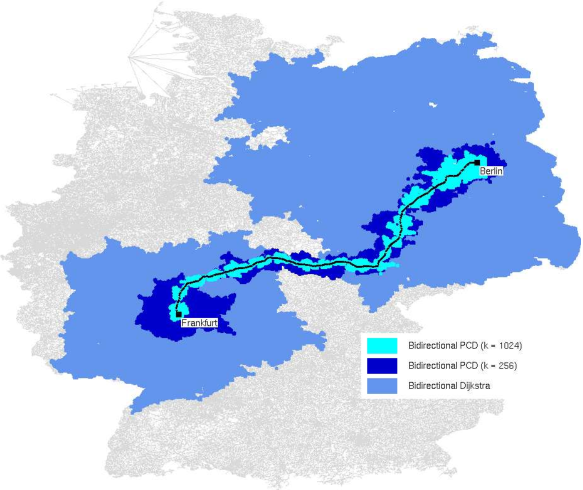
Test instances:

- Real-world road networks
- Western Europe, US
- Travel time edge weights

Speedup:

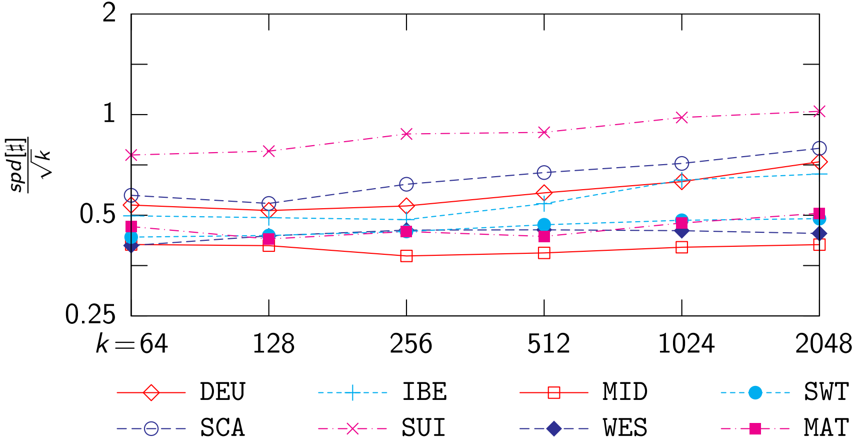
- Ratio of costs Dijkstra vs. PCD
- Average query time
- Average number of settled nodes

Search Space



- Dijkstra: two touching balls around s and t
- PCD: corridor around shortest path

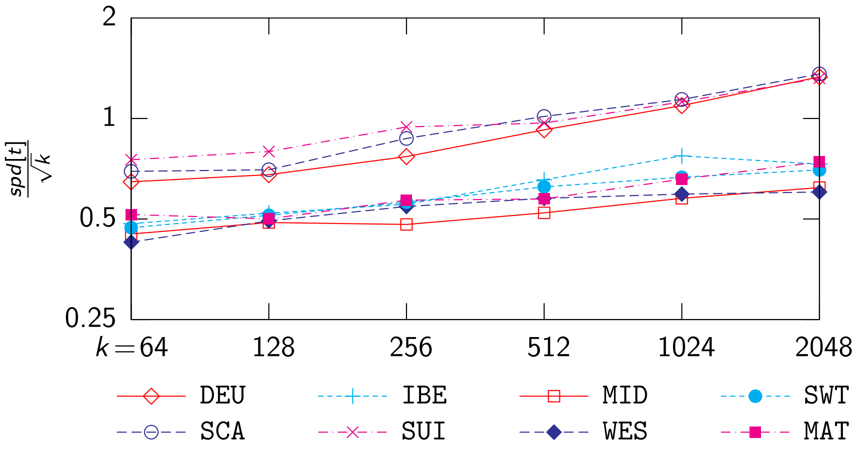
Speedup (Settled Nodes)



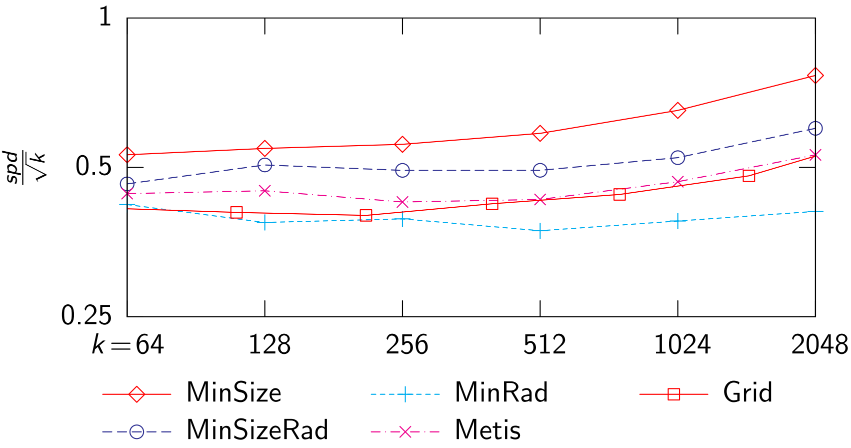
Speedup

Graph	k	$\frac{B+k^2}{n}$	prep. [min]	Bidirectional PCD Query t [ms]	*spd*	settled [$\sharp$]	*spd*
DEU	2^4	< 0.01	2.6	2114	2.5	1 028 720	2.4
	2^5	< 0.01	6.5	1631	3.3	833 545	3.1
	2^6	0.01	11.1	971	5.2	553 863	4.3
	2^7	0.01	19.1	622	7.7	404 121	5.8
	2^8	0.03	35.0	422	12.3	295 525	8.5
	2^9	0.08	68.6	242	20.9	188 239	13.2
	2^{10}	0.26	123.0	157	35.0	127 604	20.2
	2^{11}	0.99	246.9	105	60.3	82 404	32.7
	2^{12}	3.88	558.2	62	**114.9**	50 417	57.4

Speedup (Query Time)



Partitioning Methods



Conclusion

- Interpolation between APSP and plain Dijkstra via k
- Flexible trade-off between preprocessing cost and speedup
- Sublinear additional space

- Combines with any partitiong method
- No need of graph layout

- High speedups independent of graph size
- Provide goal-direction instead using A^*

Future Work

- Objective function for clustering
- Other partitioning methods
- Existing k-center approximations
- Provide goal-direction to...
  - Highway hierarchies
  - Reach-based pruning (instead of landmark-A^*)
- Alternative to A^* useful for other applications?