

Minimum-weight tree shortcutting for Metric TSP

Vladimir Deineko and Alexander Tiskin

Warwick Business School and Department of Computer Science
University of Warwick

- 1 The Metric Travelling Salesman Problem
- 2 Minimum-weight double tree shortcutting algorithm
- 3 Approximation lower bounds
- 4 Computational experiments
- 5 Conclusions and future work

1 The Metric Travelling Salesman Problem

2 Minimum-weight double tree shortcutting algorithm

3 Approximation lower bounds

4 Computational experiments

5 Conclusions and future work

The Metric Travelling Salesman Problem

The *Travelling Salesman Problem (TSP)*: find a minimum-weight Hamiltonian cycle in a weighted graph

The *Metric TSP*: weights satisfy the triangle inequality (e.g. planar or 3D Euclidean; shortest paths in a graph; $\{1, 2\}$)

Metric TSP:

- NP-hard
- NP-hard to approximate within 1.00456
- polynomial to approximate within 1.5

Planar Euclidean TSP:

- NP-hard
- polynomial to approximate within any $\epsilon > 0$

A simple approximation for Metric TSP:

- start with the minimum spanning tree (MST)
- add edges to MST to make an Eulerian subgraph
- take an Euler tour of the subgraph
- remove repeated nodes from Euler tour (*shortcutting*)

Obtaining an Eulerian subgraph:

- the *double-tree (DT)* algorithm simply doubles every MST edge
 $weight = 2w(MST) \leq 2w(OPT)$
- the *Christofides–Serdyukov (CS)* algorithm adds to MST the minimum-weight full matching M on its odd-degree nodes
 $weight = w(MST) + w(M) \leq w(OPT) + \frac{1}{2}w(OPT) = \frac{3}{2}w(OPT)$

Shortcutting cannot increase the weight, and can be done in any order

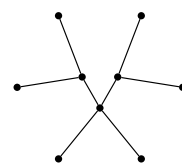
For both the DT and CS methods, we now consider the *relative* optimum problem: find the *minimum-weight shortcutting* of the Eulerian graph

For general Metric TSP, both DT and CS min-weight shortcutting problem is NP-hard (a simple reduction from Hamiltonian cycle)

For planar Euclidean TSP:

- min-weight CS shortcutting is NP-hard [Papadimitriou, Vazirani: 1984]
- min-weight DT shortcutting was long believed to be NP-hard

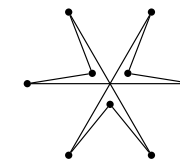
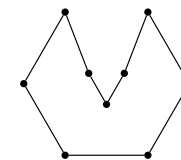
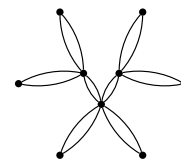
In fact, min-weight DT shortcutting is polynomial, as a special case of *PQ-tree TSP*



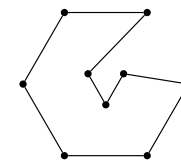
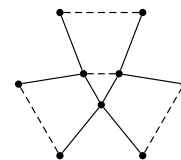
Eulerian graph

Shortcuttings

DT



CS



PQ-tree TSP: time $O(2^d n^3)$, memory $O(2^d n^2)$

[Burkard, Deineko, Woeginger: 1998]

$d \leq 4$ for planar Euclidean TSP

The PQ-tree corresponding to min-weight DT shortcutting has special structure: every non-leaf node has a leaf child

We can exploit this structure to design a more efficient algorithm for min-weight DT shortcutting

(Will describe the algorithm directly on the MST, bypassing the PQ-tree)

The Metric Travelling Salesman Problem

Questions:

- can min-weight DT shortcutting be computed more efficiently than by the general PQ-tree TSP approach?
- does min-weight DT shortcutting approximate the optimal TSP tour any closer than arbitrary DT shortcutting (in the worst case)?
- how well does min-weight DT shortcutting work in practice?

Answers:

- yes: a new algorithm with time $O(4^d n^2)$, memory $O(4^d n)$
 $d \leq 4$ for planar Euclidean TSP
- not in general: a new lower bound of 2 in the shortest path metric planar Euclidean TSP: currently no improvement on the upper bound of 2, but a new lower bound of $\frac{8+\sqrt{3}}{6} \approx 1.622$
- quite well!

Minimum-weight double tree shortcutting algorithm

Root the MST at an arbitrary node r

Dynamic programming on the rooted MST:

- the *upsweep* phase builds a data structure bottom-up, outputs the min-weight DT shortcutting weight
- the *downsweep* phase runs on data structure top-down, outputs the min-weight DT shortcutting edges

Notation:

$d(u, v)$: the metric distance

$C(u) = \{\text{all children of } u\}$ $T(u) = \{\text{all descendants of } u\}$

$T(U) = \uplus_{u \in U} T(u)$, where U a set of siblings, \uplus the disjoint union

Main property of a DT shortcutting: for all u , upon entering $T(u)$ at some point, the tour visits (*sweeps*) the whole $T(u)$ before leaving it

1 The Metric Travelling Salesman Problem

2 Minimum-weight double tree shortcutting algorithm

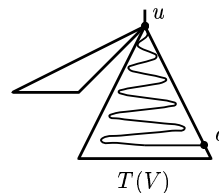
3 Approximation lower bounds

4 Computational experiments

5 Conclusions and future work

Minimum-weight double tree shortcutting algorithm

The upsweep phase



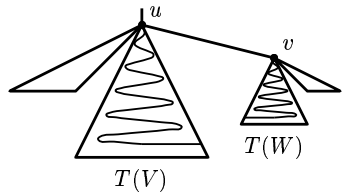
Let $V \subseteq C(u)$, $a \in T(V)$

$D_V^u(a)$: min-weight path $u \rightsquigarrow a$ sweeping $u \uplus T(V)$

The *current* node u to process is any unprocessed node, for which all the children have been processed

Processing node u gives $D_V^u(a)$ for all $V \subseteq C(u)$, $a \in T(V)$

The upsweep phase (contd.)



Processing node u

We have $D_W^v(a)$ for all $v \in C(u)$, $W \subseteq C(v)$, $a \in T(W)$

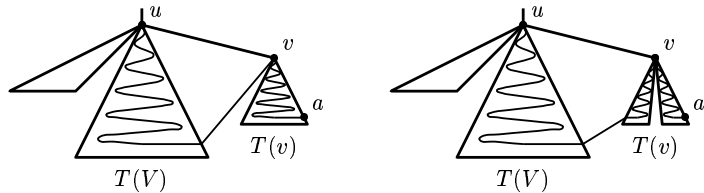
Now build $D_V^u(a)$ for all $V \subseteq C(u)$, $a \in T(V)$, by induction on size of V

Induction base trivial: $V = T(V) = \emptyset$, there is no a so no $D_V^u(a)$

Inductive step: computing $D_{V \uplus v}^u(a)$, where $v \in C(u)$ is such that $a \in T(v)$

Inductive hypothesis: we have $D_V^u(a)$ for all $a \in T(V)$

The upsweep phase, inductive step (contd.)



Note we already have $D_{V \uplus v}^u(v) = D_{V, C(v)}^u(v)$

We are now ready to compute $D_{V \uplus v}^u(a)$ for all $a \in T(v)$

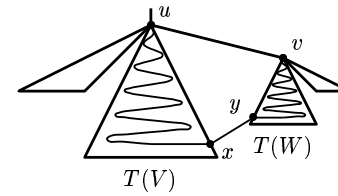
Let $a \in T(C(v))$

$$D_{V \uplus v}^u(a) = \min [D_{V, W}^u(v) + D_W^v(a)]$$

for all partitionings $W \uplus \bar{W} = C(v)$, such that $a \in T(\bar{W})$

Inductive step completed

The upsweep phase, inductive step (contd.)



Let $W \subseteq C(v)$

$D_{V, W}^u(v)$: min-weight path $u \rightsquigarrow v$ sweeping $u \uplus T(V)$ and then $T(W) \uplus v$

$$D_{V, W}^u(v) = \begin{cases} d(u, v) & \text{if } V = \emptyset, W = \emptyset \\ \min [d(u, y) + D_W^v(y)] & \text{if } V = \emptyset, W \neq \emptyset \\ \min [D_V^u(x) + d(x, v)] & \text{if } V \neq \emptyset, W = \emptyset \\ \min [D_V^u(x) + d(x, y) + D_W^v(y)] & \text{if } V \neq \emptyset, W \neq \emptyset \end{cases}$$

for all $x \in T(V)$, $y \in T(W)$

The upsweep phase (contd.)

Processing current node u gives us all $D_V^u(a)$, where $V \subseteq C(u)$, $a \in T(V)$

Eventually, we process root r , obtaining all $D_S^r(a)$ for $S \subseteq C(r)$, $a \in T(S)$

The min-weight DT tour has weight $\min_{a \neq r} [D_{C(r)}^r(a) + d(a, r)]$

Running time:

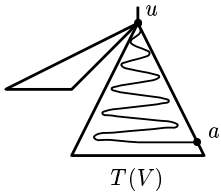
Total number of quadruples u, v, x, y is $\leq n^2$ (since x, y determine u, v)

Total number of triples u, v, a is $\leq n^2$ (since u, a determine v)

For each quadruple/triple, we have 2^d choices of V and 2^d choices of W

Overall upsweep time $O(4^d n^2)$, memory $O(4^d n)$

The downsweep phase



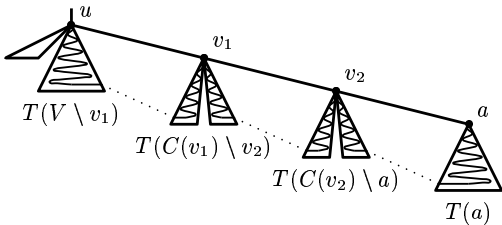
Recall $D_V^u(a)$: min-weight path $u \rightsquigarrow a$ sweeping $u \uplus T(V)$

We solve recursively the following problem: given node u , $V \subseteq C(u)$, $a \in T(V)$, find the edges of $D_V^u(a)$

Recall: the min-weight DT tour has weight $\min_{a \neq r} [D_{C(r)}^r(a) + d(a, r)]$

Therefore, we shall call the recursive procedure for the edges of $D_V^u(a)$ with $u = r$ and the value a that gave the min-weight DT tour

The downsweep phase (contd.)



Optimal choice of bipartitionings $W_i \uplus \overline{W}_i$ can easily be found by dynamic programming, using values $D_{W_i, W_{i+1}}^{v_i}(v_{i+1})$ obtained in the upsweep

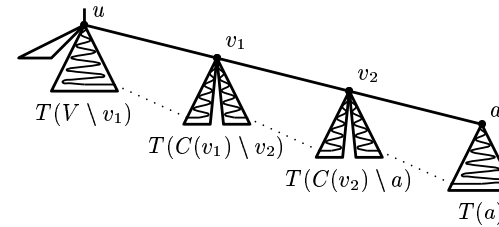
This gives us k disconnected edges of the min-weight DT tour

We now call the procedure recursively for each of $(u, V \setminus v_1)$, (v_1, W_1) , (v_1, \overline{W}_1) , (v_2, W_2) , (v_2, \overline{W}_2) , \dots , $(a, C(a))$

Running time: in each recursion level time $O(4^d k)$

Overall downsweep time and memory $O(4^d n)$

The downsweep phase (contd.)



Consider the MST path $u = v_0 \rightarrow v_1 \rightarrow v_2 \rightarrow \dots \rightarrow v_k = a$

The min-weight DT tour:

- sweeps $u \uplus T(V \setminus v_1)$
- for each i , sweeps $v_i \uplus T(W_i)$ and $v_i \uplus T(\overline{W}_i)$, for some bipartitioning $W_i \uplus \overline{W}_i = C(v_i) \setminus v_{i+1}$
- sweeps $T(a)$

Summary so far: a min-weight DT shortcutting algorithm in time $O(4^d n^2)$, memory $O(4^d n)$

Improves on the general PQ-tree TSP approach by exploiting the special structure of the PQ-tree: every non-leaf node has a leaf child

Related: a PQ-tree TSP *path* algorithm for a *binary balanced* PQ-tree in time $O(n^2 \log n)$, memory $O(n)$ [Brandes: 2007]

1 The Metric Travelling Salesman Problem

2 Minimum-weight double tree shortcutting algorithm

3 Approximation lower bounds

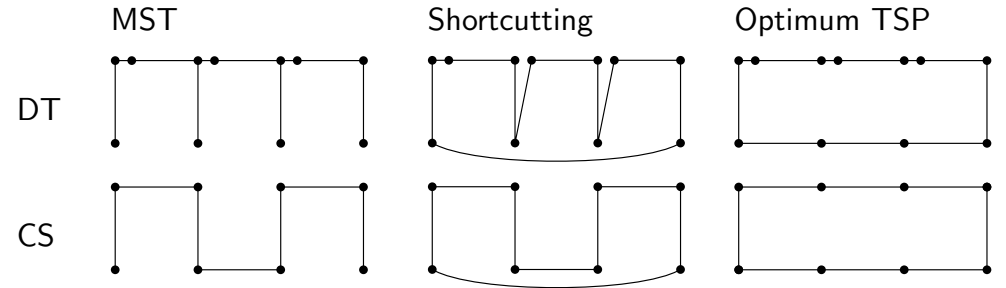
4 Computational experiments

5 Conclusions and future work

Approximation lower bounds

Recall that DT shortcutting gives a 2-approximation, and CS shortcutting gives a 1.5-approximation for Metric TSP

Ratios 2 and 1.5 are sharp for DT and CS for planar Euclidean TSP

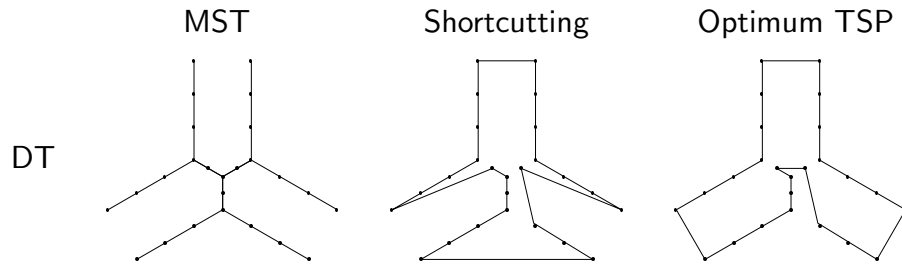


In both these instances, min-weight shortcutting gives ratio 1.5, so

- ratio 1.5 is sharp for min-weight CS shortcutting
- can we improve on ratio 2 for min-weight DT shortcutting?

Approximation lower bounds

A tough instance for min-weight DT shortcutting



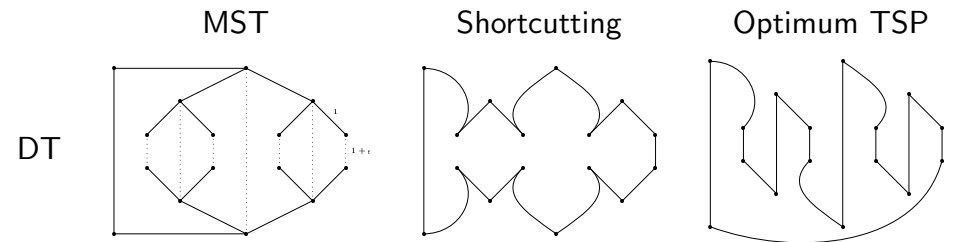
Approximation ratio:

- $\frac{8+\sqrt{3}}{6} \approx 1.622$ in the planar Euclidean metric
- $5/3 \approx 1.666$ in the planar hexagonal Minkowski metric

Problem: where between 2 and 1.622 is the true worst-case min-weight DT approximation ratio for planar Euclidean TSP?

Approximation lower bounds

A worst-case instance for min-weight DT shortcutting in the graph shortest path metric



Approximation ratio 2

- 1 The Metric Travelling Salesman Problem
- 2 Minimum-weight double tree shortcutting algorithm
- 3 Approximation lower bounds
- 4 Computational experiments**
- 5 Conclusions and future work

Previous computational experiments:

- comparison of 37 tour-constructing algorithms, including naive DT shortcutting, on 24 Euclidean instances from TSPLIB: DT showed very poor quality [Reinelt:1994]
- re-run of Reinelt's experiment, including min-weight DT shortcutting by PQ-tree TSP: DT showed very good quality, but poor running time [Deineko]

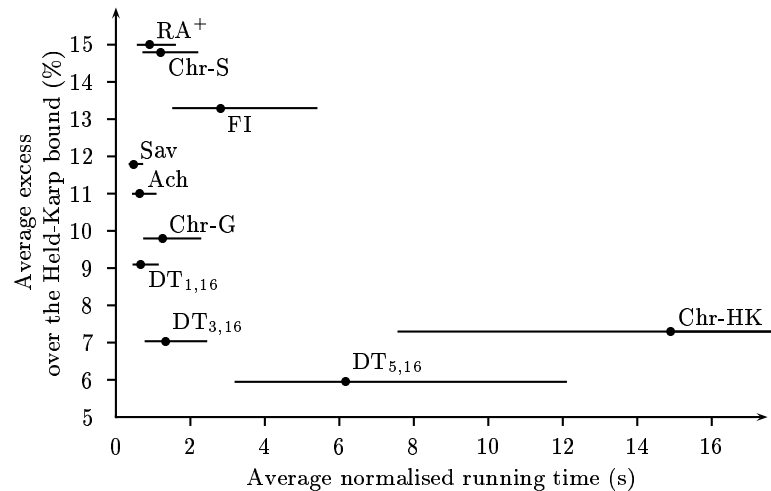
We implemented the new min-weight DT algorithm, including some additional heuristics:

- bounding subtree search depth: speedup, no significant loss of quality
- transforming MST to increase node degree: improves quality

Computational experiments

$DT_{D,k}$: min-weight DT shortcutting, target degree D , search depth $\leq k$

$DT_{1,\infty}$: the default min-weight DT shortcutting algorithm



- 1 The Metric Travelling Salesman Problem
- 2 Minimum-weight double tree shortcutting algorithm
- 3 Approximation lower bounds
- 4 Computational experiments**
- 5 Conclusions and future work

An interesting new algorithm for Metric TSP, with practical relevance

Further work:

- alternative methods of obtaining the initial Eulerian graph (e.g. Delaunay triangulations)
- more detailed exploration of possible optimisations
- closing the approximation ratio gap between 2 and 1.622!
- exploring connections with FPT theory
- exploring conditions on distance matrices that would guarantee that min-weight TSP shortcutting gives the optimum TSP tour