Good triangulations yield good tours

Adam N. Letchford, Nicholas A. Pearson*

Department of Management Science, Lancaster University, Lancaster LA1 4YW, UK

Available online 4 May 2006

Abstract

Consider the following heuristic for *planar Euclidean* instances of the traveling salesman problem (TSP): select a subset of the edges which induces a planar graph, and solve either the TSP or its graphical relaxation on that graph. In this paper, we give several motivations for considering this heuristic, along with extensive computational results. It turns out that the *Delaunay* and *greedy triangulations* make effective choices for the induced planar graph. Indeed, our experiments show that the resulting tours are on average within 0.1% of optimality.

Scope and purpose

The traveling salesman problem (TSP) is a fundamental and well-known problem in combinatorial optimisation. It has many applications, for example in vehicle routing and machine scheduling. This paper proposes several heuristics methods for the Euclidean TSP, based on the use of triangulations, and gives extensive computational results.
© 2006 Elsevier Ltd. All rights reserved.

Keywords: Delaunay triangulation; Traveling salesman problem

1. Introduction

The *traveling salesman problem* (TSP) is the problem of finding a Hamiltonian circuit (or *tour*) of minimum weight in a complete edge-weighted graph. The TSP is a classic example of a hard combinatorial optimisation problem. For surveys, we refer the reader to the two excellent volumes edited by Lawler et al. [1] and Gutin and Punnen [38]. In this paper, we assume that edge-costs are symmetric, or, equivalently, that the graph is undirected.

A special case of the TSP that has attracted much interest, and which also arises in many practical applications, is obtained when the vertices of the graph correspond to points in the *Euclidean plane*, and the distance between any two vertices is equal to the Euclidean distance between the corresponding points. For brevity, we will call such instances simply *Euclidean*. The Euclidean TSP is a special case of the so-called *metric* TSP, in which the costs obey the triangle inequality. Nevertheless, it is strongly $\mathcal{NP}$-hard [3].

Related to, but distinct from, the Euclidean TSP is the *planar graph* TSP. This is the version of the TSP in which a planar graph $G = (V, E)$ is given, with weights on the edges of E, and one seeks the minimum cost tour which uses only edges in E. This problem too is strongly $\mathcal{NP}$-hard. Indeed, it is $\mathcal{NP}$-hard even to test if a planar graph is Hamiltonian [4].

---

* Corresponding author. Fax: +44 1524 844885.

E-mail addresses: a.n.letchford@lancaster.ac.uk (A.N. Letchford), n.pearson@lancaster.ac.uk (N.A. Pearson).

In this paper, we examine the following natural heuristic for the Euclidean TSP: take a *triangulation* of the vertices in the plane, and then solve the planar graph TSP on the triangulation. As possible candidates for the triangulation, we consider the *Delaunay* and *greedy* triangulations, which are well-known in computational geometry. We give several reasons for considering these triangulations in what follows.

The possibility of using the Delaunay triangulation to generate heuristic tours was in fact already suggested by Reinelt [5] and Stewart [6]. However, here we give a deeper theoretical analysis, and present extensive computational results for several variants of the heuristic. As well as considering the two triangulations mentioned, we also consider the option of solving the so-called *graphical relaxation* of the planar graph TSP, i.e., the relaxation in which vertices are permitted to be visited more than once and edges are permitted to be traversed more than once [7,8]. We also consider the possibility of obtaining even better tours by 'short-cutting' the solution to the graphical relaxation.

The structure of the remainder of the paper is as follows. In Section 2 we review the relevant literature. In Section 3 we describe the various heuristics in more detail and make some theoretical observations. In Section 4 we describe some extensive computational experiments performed on instances from the well-known TSPLIB library. The results confirm that the tours obtained after short-cutting, in particular, are of remarkably high quality. Some concluding remarks are given in Section 5.

2. Literature review and basic concepts

In this section we briefly review the relevant literature, and along the way recall some basic concepts and definitions. In Section 2.1 we cover the various planar variants of the TSP mentioned above. In Section 2.2 we cover the Delaunay and greedy triangulations, and related concepts such as *proximity graphs* and *spanners*.

2.1. Planar variants of the TSP

Although, as mentioned above, the TSP remains strongly $\mathcal{NP}$-hard even when restricted to Euclidean instances, there are some theoretical and empirical evidence that the Euclidean TSP is in some sense a 'relatively easy' special case of the metric TSP:

- The metric TSP is APX-hard (e.g., [9]), which means that no polynomial-time approximation scheme (PTAS) is likely to exist, whereas a PTAS is known for the Euclidean TSP [10].
- The best known time bound for solving the metric TSP exactly is $\mathcal{O}(n^2 2^n)$ [11], whereas for the Euclidean TSP an exact $\mathcal{O}(c^{\sqrt{n}})$ algorithm is known [12].
- Large-scale Euclidean instances, with several thousands of vertices, can often be solved to proven optimality in a reasonable amount of time using the so-called *branch-and-cut* approach (see Naddef [2] for a survey).

We now consider the planar graph TSP and its graphical relaxation. For brevity, we refer to the graphical relaxation as the *planar GTSP*. It is well-known, and easy to show, that a planar GTSP instance defined on an edge-weighted graph G can be transformed to a TSP instance on a complete graph: it suffices to set the cost of traversing from i to j in the complete graph equal to the cost of the shortest path from i to j in G. Note that the resulting TSP instance is metric. Thus, the planar GTSP can be viewed as a special case of the metric TSP. The planar graph TSP, on the other hand, is not regarded as a special case of the metric TSP: although an instance of the planar graph TSP can be transformed to a metric TSP instance using the standard 'big M' method, this transformation is not approximation-preserving.

In any case, there is a sense in which the planar graph TSP and planar GTSP, too, are in some sense 'relatively easy' special cases of the TSP:

- A PTAS is known for the planar GTSP [13].
- For the planar graph TSP and the planar GTSP, exact $\mathcal{O}(c^{\sqrt{n}})$ algorithms are known (e.g., [14]).

Moreover, there are two good reasons to think that the planar graph TSP and planar GTSP will be in practice much easier to solve to optimality via branch-and-cut than the Euclidean TSP:

- Since planar graphs are sparse, one only needs $\mathcal{O}(n)$ variables, compared to $\mathcal{O}(n^2)$ in the case of the Euclidean TSP (see, e.g., [8]). Thus, no *pricing* (column generation) is necessary.
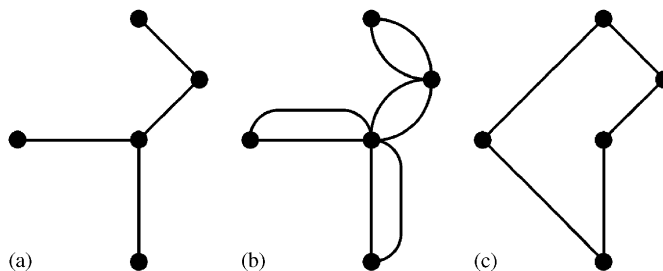
Fig. 1. (a) MST, (b) after doubling of edges, (c) after short-cutting.

- There are exceptionally fast *separation algorithms*, i.e., algorithms for generating useful cutting planes, known for the planar graph TSP and planar GTSP [15–17].

As, in addition, the tours resulting from the heuristics proposed in this paper are of exceptionally high quality, we believe that the heuristics could be of practical as well as theoretical interest.

For what follows, it will also be useful to mention the 'twice-around-the-tree' heuristic for the metric TSP, due to Rosenkrantz et al. [18], and the associated concept of 'short-cutting'. This well-known heuristic involves computing the minimum spanning tree (MST), doubling its edges to make each vertex degree even, and then taking 'short-cuts' to obtain a tour (Fig. 1). The heuristic yields a tour of cost no more than twice the optimum. In general, there may be several ways to take shortcuts. It was shown by Burkard et al. [19] that the optimal set of shortcuts can be computed in polynomial time, but it is not known whether this leads to an improvement in the performance guarantee.

We note in passing that the MST is a trivial example of a planar graph, and the 'doubled' tree is a trivial GTSP tour. Therefore, the double spanning tree heuristic can be regarded as involving the solution of a trivial planar GTSP instance.

2.2. The Delaunay triangulation and related graphs

Let V be a set of points in the plane. A *triangulation* of V is a planar graph $G = (V, E)$, embedded in the plane in such a way that each edge $\{i, j\} \in E$ is represented by a straight-line, and such that all internal faces are triangles. (The outer face need not be a triangle.) The number of possible triangulations grows exponentially with the cardinality $|V|$.

The *Delaunay triangulation*, and its dual graph, the *Voronoi diagram*, are well-known in computational geometry. To construct the Voronoi diagram, one partitions the plane into convex (polygonal) regions, one for each $i \in V$. A point is assigned to a given $i \in V$ if it is closest to i. The points which are assigned to two vertices in V lie on line-segments, which define the Voronoi diagram [20]. The geometric dual of the Voronoi diagram is the Delaunay triangulation [21]. Rather than go into the formal details, we simply display Fig. 2 and refer the reader to a book on computational geometry such as Okabe et al. [22].

For brevity we say 'DT' rather than 'Delaunay triangulation' in what follows. The DT has a number of remarkable properties. For example:

- It is the triangulation which maximises the smallest angle of all the triangles in the triangulation.
- It contains a number of other well-known 'proximity graphs' as subgraphs. Proximity graphs, well-known in computational geometry, provide a succinct (linear-sized) representation of how 'close' each vertex is to the other vertices. Examples of proximity graphs contained in the DT include the nearest neighbour graph and the MST [23], the relative neighborhood graph [24], and the Gabriel graph [25].
- It is unique (non-degenerate) if there are no four cocircular points, i.e. four points on the same circle.
- It can be computed in $\mathcal{O}(n \log n)$ time, using $\mathcal{O}(n)$ space [23].
- Each triangle of the DT has an empty circumcircle [26], i.e., for each triangle there exists a circle containing the triangle's vertices but not containing any other vertex of the graph.

Another important property, for our purposes, is the fact that the DT is a '2.42-spanner'. This means that the length of the shortest path between any two points in the DT is never larger than 2.42 times the Euclidean distance [27].
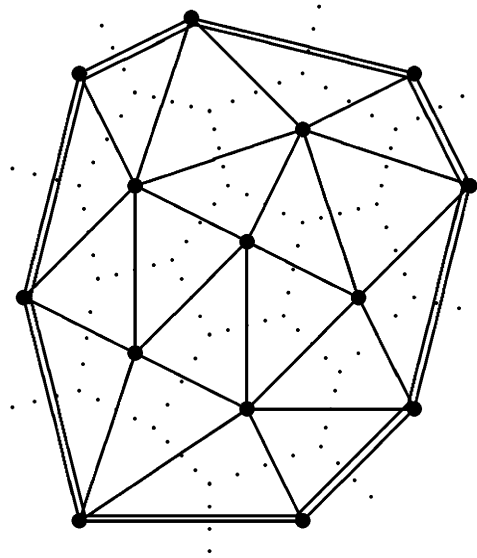
Fig. 2. Delaunay triangulation (solid lines) and Voronoi diagram (dashed lines). Lines on convex hull are doubled.
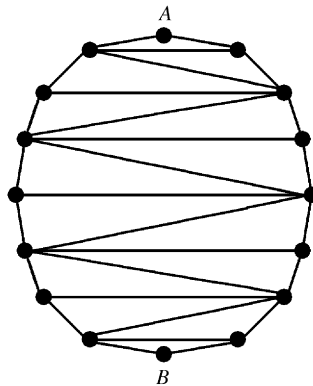

Fig. 3. If points are located on a circle, a ratio approaching $\pi/2$ can occur.

In fact, the worst known family of examples gives a ratio approaching only $\pi/2 \approx 1.57$ [28]. As shown in Fig. 3, this ratio is approached when the points are equally spaced on a circle. If the DT takes the 'zig-zag' form indicated in the figure, the shortest path on the triangulation between the two end-vertices of the 'zig-zag' (labelled A and B in the figure) will be $\pi/2$ times the diameter.

Despite the attractive features of the DT as a 'good' planar subgraph on which the Euclidean TSP can be solved, its most troubling feature is that it may not contain a Hamiltonian cycle [29]. Moreover, the problem of testing whether a DT is Hamiltonian is $\mathcal{NP}$-complete [30]. Nevertheless, Genoud's study [31] indicates that non-Hamiltonian Delaunay triangulations are rare.

Another triangulation which has received much attention is the so-called *greedy triangulation*, which we will refer to as 'GT'. It is the triangulation obtained by adding nonintersecting edges, in non-decreasing order, to the point set until no further edges can be added. Less is known about the GT than about the DT. In fact, to the best of the authors' knowledge, no research has been done to determine whether the GT is even Hamiltonian. Nevertheless several important properties are known, such as the following:

- It is unique, provided that no pair of edges are equal in length.
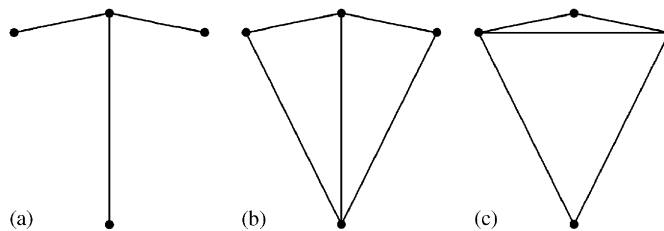- It can be computed from the DT in linear time [32].

Fig. 4. (a) MST, (b) DT, (c) GT.

- It does not necessarily contain the MST as a subgraph. Fig. 4 shows a simple counterexample on four vertices, which was provided to us by Levcopoulos [33].
- It is a *t*-spanner for some constant *t* [34], but no good upper bound on *t* is known.

Both the DT and GT eliminate 'long' edges to some extent, however, neither the DT nor the GT yields the minimum weight triangulation (MWT) (see [35]). Nevertheless, the DT and GT are on average close to the MWT [36]. Interestingly, Manacher and Zobrist [35] have shown that the total edge length of the DT can be significantly larger than the total edge length of the GT for the same point set. Therefore, the GT seems to be the better of the two in approximating the MWT.

It is worth noting that there are several important differences between the DT, GT and MWT (see [34]). For our purposes, the most significant difference is that both the DT and GT can be computed in $\mathcal{O}(n \log n)$ time [23], whereas for general point sets there is no known polynomial-time algorithm to compute the MWT. This underlies the selection of the DT and GT, rather than the MWT, in this paper.

3. The heuristics and their analysis

As mentioned in the introduction, for each of the two triangulations, the DT and the GT, we explore three ways of generating tours:

1. *Solve the planar graph TSP on the triangulation*: This means finding the best tour using only edges in the triangulation. The disadvantage here, at least in theory, is that the triangulation may not be Hamiltonian. In this case, the heuristic will fail even to find a tour.

2. *Solve the planar GTSP on the triangulation*: This resolves the Hamiltonicity problem, and, perhaps more importantly, considerably expands the set of feasible solutions. As we will see, solutions of much lower cost can be obtained in this way.

3. *Shortcut the planar GTSP solution*: In the majority of cases, the optimal planar GTSP tour visits some vertices more than once. As in the 'twice-around-the-tree' heuristic mentioned above, we can convert it into a Hamiltonian circuit by 'short-cutting'. Short-cutting can be performed easily in linear time and causes the cost of the tour to decrease (in theory at least).

This yields six variants of the heuristic in total.

A natural question which arises is, what is the performance guarantee of the six heuristics (i.e., the worst-case ratio between the cost of the heuristic tour and the optimal tour)? Since the DT can fail to be Hamiltonian, and the same seems likely to be true of the GT, the variants based on the planar graph TSP have no performance guarantee.

For the variants based on the solution of the planar GTSP (possibly with short-cuts) on the DT, we can say something better. Since the DT contains the MST as a subgraph, these heuristics yield tours which are at least as good as the one obtained from the twice-around-the-tree heuristic. Thus, they have a performance guarantee of at most 2. Moreover, if the DT is indeed a $\pi/2$-spanner, as conjectured by several authors, then the heuristics will automatically have a performance guarantee of $\pi/2$. In fact, we would like to make the following conjecture:

Conjecture 1. *The heuristics based on solving the planar GTSP on the DT have a performance guarantee strictly better than* $\pi/2$.

The motivation for this conjecture is as follows: when these heuristics are applied to the 'bad' instances of the form illustrated in Fig. 3, they yield the optimal TSP tour. Thus, the known 'bad' examples for the DT are in fact 'good' examples for these particular TSP heuristics.

Finally, for the heuristics based on the solution of the planar GTSP on the GT, it follows from the result of Das and Joseph [34] that they have a constant performance guarantee. However, the precise value of this constant is unknown.

4. Computational experiments

4.1. Test problems

Rather than construct random instances of our own to compare the performance of the six heuristics for the Euclidean TSP, we decided to use instances taken from TSPLIB [37]. TSPLIB is a library of TSP instances, mostly taken from real applications, which are widely regarded as representative. TSPSLIB contains over 70 two-dimensional Euclidean instances. We selected those with $n < 1500$, leading to 58 instances in total.

In fact, the use of TSPLIB instances revealed an interesting phenomenon. As mentioned above, short-cutting a GTSP tour should lead to a decrease in the total cost. At least, it should lead to no increase. In our computational experiments, however, we noticed that this was not always true. This anomaly turned out to be caused by the TSPLIB standard for rounding Euclidean distances to the nearest integer. The resulting edge costs do not always obey the triangle inequality. Suppose, for example, that the Euclidean distance from u to v is 6.6, whereas w is at a distance of 3.4 from both u and v. If the path $u-w-v$ appears in the GTSP tour, the cost contribution will be $\lfloor 3.4 \rfloor + \lfloor 3.4 \rfloor = 6$. If, after short-cutting, we obtain the edge $u-v$, the cost contribution will be $\lceil 6.6 \rceil = 7$. Thus, on rare occasions, short-cutting a tour can lead to a small *increase* in length. We actually observed this phenomenon in a few cases.

4.2. Methodology

To perform our experiments, we used the software package CONCORDE, which is available on the web. CONCORDE is a state-of-the-art computer code for the symmetric TSP written by Applegate, Bixby, Chvátal and Cook, and has been used to solve the largest known TSP to optimality, 24,978 cities in Sweden. In the two-dimensional case, CONCORDE takes as input the Cartesian coordinates of the vertices, calculates Euclidean distances for the complete graph and solves the corresponding TSP. It also contains code for some related network optimisation problems, such as the Lin–Kernighan TSP heuristic, several edge generation routines, and algorithms for solving fractional two-matching problems. One of these edge generation routines was used to obtain a set of Delaunay triangulations. We created our own code for the greedy triangulations.

Although CONCORDE is extremely fast, it has one important limitation: it is designed to deal with instances in which the graph is complete. Thus, we had to devise a way of restricting it to using only edges in the given triangulation for each of our heuristics. We accomplished this, in a rather ad hoc manner, as follows:

Finding the best tour using only edges in the triangulation: This required three modifications to the CONCORDE code: start with only the edges in the triangulation, switch off the column generation algorithms and switch off the initial tour-finding heuristic(s). CONCORDE then required a starting tour, i.e., an arbitrary tour using only edges in the triangulation.

To obtain the starting tour, CONCORDE was used to solve the TSP in a graph in which every edge not in the triangulation was given a weight of 1 and every triangulation edge a weight of 0. The idea is to test if the triangulation is Hamiltonian and, if so, to produce a starting tour. Each of our test instances was found to be Hamiltonian.

Solving the GTSP on the triangulation: Since CONCORDE is not designed to solve GTSP instances, it was necessary to convert the GTSP instance into a complete (and metric) TSP instance. This involved computing all-pairs shortest paths in the triangulation and using the costs of these paths in place of the original Euclidean costs. Therefore, we coded a simple routine which takes an edge-weighted triangulation as input and uses Dijkstra's shortest path algorithm to calculate the distance between each pair of vertices. The tour found by CONCORDE can then easily be converted back into a GTSP tour on the triangulation.

Shortcutting the GTSP tour: Due to the way in which we compute the GTSP tour using CONCORDE, there is no need to perform short-cutting *explicitly*. Indeed, it suffices to take the tour output from CONCORDE when solving the

Table 1
Number of instances solved to optimality, maximum and mean percentage gap for both heuristics on the DT and GT

	No. optimal	Maximum	Mean
Delaunay triangulation:			
Planar graph TSP	18	3.30	0.28
Planar GTSP	18	0.73	0.13
GTSP with shortcuts	19	0.73	0.10
Greedy triangulation:			
Planar graph TSP	11	3.70	0.32
Planar GTSP	11	0.79	0.21
GTSP with shortcuts	12	0.67	0.18

Table 2
Euclidean TSPLIB instances: percentage gap heuristic results

Name	Delaunay triangulation			Greedy triangulation		
	TSP	GTSP	GTSP with s-cuts	TSP	GTSP	GTSP with s-cuts
eil51	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
berlin52	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
st70	**0.00**	**0.00**	**0.00**	0.15	0.15	0.15
eil76	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
pr76	0.48	0.48	0.07	0.87	0.41	0.20
rat99	**0.00**	**0.00**	**0.00**	0.08	0.08	0.08
kroA100	0.05	0.05	0.05	0.59	0.59	0.17
kroB100	**0.00**	**0.00**	**0.00**	0.05	0.05	0.05
kroC100	**0.00**	**0.00**	**0.00**	0.02	0.02	0.02
kroD100	0.06	0.06	0.06	0.51	0.51	0.51
kroE100	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
rd100	0.28	0.28	0.28	0.28	0.28	0.28
eil101	0.48	0.16	**0.00**	**0.00**	**0.00**	**0.00**
lin105	0.58	0.30	0.16	0.33	0.16	0.16
pr107	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
pr124	0.86	0.73	0.73	0.97	0.79	0.37
bier127	0.52	0.38	0.16	0.52	0.37	0.22
ch130	0.03	0.03	0.03	0.05	0.05	0.05
pr136	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
pr144	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
ch150	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
kroA150	**0.00**	**0.00**	**0.00**	0.10	0.10	0.10
kroB150	0.03	0.03	0.03	0.23	0.23	0.23
pr152	0.35	0.23	0.23	0.55	0.55	0.55
u159	**0.00**	**0.00**	**0.00**	0.05	0.05	0.05
rat195	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
d198	0.10	0.10	0.10	0.30	0.30	0.27
kroA200	0.19	0.07	0.03	0.04	0.04	0.04
kroB200	0.05	0.05	0.05	0.08	0.08	0.08

second heuristic, and recompute its cost using original Euclidean distances in place of shortest-path distances. This may not be the optimal way to short-cut, but it is extremely fast and trivial to implement.

We want to stress that CONCORDE is not at all optimised for sparse graphs, and, as a result, we focus solely on the bounds obtained by our heuristics and not on their comparative running times. Nevertheless, we discuss the potential for specially tailored branch-and-cut code and improved running times in Section 5.

Table 3
Euclidean TSPLIB instances: percentage gap heuristic results

Name	Delaunay triangulation			Greedy triangulation		
	TSP	GTSP	GTSP with s-cuts	TSP	GTSP	GTSP with s-cuts
ts225	3.30	0.65	0.53	3.70	0.65	0.62
tsp225	**0.00**	**0.00**	**0.00**	**0.00**	0.08	0.08
pr226	1.70	0.05	0.05	0.67	0.67	0.67
gil262	0.34	0.25	0.25	0.38	0.38	0.29
pr264	0.11	0.11	0.11	0.23	0.17	0.17
a279	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
pr299	0.68	0.32	0.27	0.25	0.25	0.25
lin318	0.54	0.29	0.18	0.49	0.28	0.25
rd400	0.01	0.01	0.01	0.11	0.10	0.03
fl417	0.50	0.29	0.18	0.54	0.54	0.54
pr439	0.28	0.10	0.04	0.07	0.07	0.07
pcb442	0.05	0.05	0.05	0.04	0.04	0.04
d493	0.03	0.03	0.03	0.94	0.62	0.56
u574	0.34	0.20	0.12	0.56	0.43	0.34
rat575	0.10	0.10	0.10	0.12	0.12	0.12
p654	0.23	0.23	0.23	0.36	0.36	0.36
d657	0.19	0.13	0.13	0.14	0.14	0.14
u724	0.08	0.08	0.08	0.22	0.17	0.18
rat783	**0.00**	**0.00**	**0.00**	0.05	0.05	0.05
pr1002	0.09	0.09	0.05	0.21	0.21	0.19
u1060	0.25	0.10	0.07	0.22	0.18	0.13
vm1084	0.17	0.09	0.04	0.21	0.16	0.11
pcb1173	0.12	0.03	0.01	0.10	0.09	0.08
d1291	0.31	0.16	0.14	0.51	0.30	0.20
rl1304	1.45	0.53	0.39	1.05	0.56	0.48
rl1323	1.08	0.61	0.38	0.93	0.68	0.55
nrw1379	0.05	0.02	0.01	0.03	0.03	0.03
fl1400	0.08	0.06	0.05	0.35	0.33	0.31
u1432	0.05	0.03	0.03	0.04	0.01	**0.00**

4.3. Results

All experiments were performed on Lancaster University's High Performance Computing facility which consists of a set of dual-processor Sun-Blade workstations, each having between 1 and 8 GB of memory.

Table 1 shows percentage gap results for each of the heuristics and the number of instances solved to optimality. The percentage gap is calculated as the difference between the heuristic tour bound and the optimal tour bound as a percentage of the optimal tour bound. The DT results indicated that planar graph TSP tours (respectively, GTSP tours with shortcuts) were within 3.3% (respectively, 0.73%) of optimal in the worst case, but were on average 0.28% and 0.1% larger than optimal. The GT heuristics had less impressive results. The shortcut heuristic solved 12 instances to optimality, and had an average percentage gap of 0.18%.

Percentage gap results on all 58 instances for the DT and GT heuristics are given in Tables 2 and 3. Instances which were solved to optimality are shown in bold.

The running times, however, were less conclusive. Although we have not reported them here in detail, we can mention that the times taken by CONCORDE to find the heuristic tours were typically only slightly smaller than the times taken to solve the original TSP instances themselves. This is unsurprising, given that CONCORDE is not designed to exploit sparsity, nor is it designed to solve GTSP instances.

5. Conclusion

The results in Table 1 show that the planar GTSP heuristics, in particular, perform extremely well. In our opinion this fact, together with the fast separation algorithms presented in [15–17], provides a strong motivation for devising

and coding a branch-and-cut algorithm specifically designed for solving planar TSP and GTSP instances. Such an algorithm may be the subject of a future paper.

A final comment: although the DT and GT are *internally* triangulated, they are not usually *full* triangulations (unless only three points lie on the convex hull of the points, which almost never occurs in practice). In theory, then, extra edges could be added to the DT and GT while preserving planarity. These additional edges could lead to even better performance of the heuristics.

References

[1] Lawler EL, Lenstra JK, Rinnooy-Kan AHG, Schmoys DB. The traveling salesman problem. New York: Wiley; 1985.
[2] Naddef D. Polyhedral theory and branch-and-cut algorithms for the symmetric TSP. In: Gutin G, Punnen A, editors. The Traveling Salesman Problem and Its Variations. Kluwer Academic Publishers. 2002.
[3] Garey MR, Graham RL, Johnson DS. Some NP-complete geometric problems. In: Proceedings of the eighth ACM STOC. 1976. p. 10–22.
[4] Garey MR, Johnson DS, Tarjan RE. The planar Hamiltonian circuit problem is NP-complete. SIAM Journal on Computing 1976;5:704–14.
[5] Reinelt G. Fast heuristics for large geometric traveling salesman problems. ORSA Journal on Computing 1992;4:206–17.
[6] Stewart WR. Euclidean traveling salesman problems and Voronoi diagrams. Working paper, School of Business Administration, College of William and Mary. Williamsburg, VA23185; 1997.
[7] Cornuéjols G, Fonlupt J, Naddef D. The traveling salesman on a graph and some related integer polyhedra. Mathematical Programming 1985;33:1–27.
[8] Fleischmann B. A cutting plane procedure for the traveling salesman problem on a road network. European Journal of Operational Research 1985;21:307–17.
[9] Papadimitriou CH, Yannakakis M. The traveling salesman problem with distances one and two. Mathematics of Operations Research 1993;18:1–11.
[10] Arora S. Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. Journal of the ACM 1998;45:753–82.
[11] Held M, Karp R. A dynamic programming approach to sequencing problems. SIAM Journal 1962;10:196–209.
[12] Smith WD. Finding the optimum N-city traveling salesman tour in the Euclidean plane in subexponential time and polynomial space. Unpublished manuscript; 1988.
[13] Arora S, Grigni M, Karger D, Klein P, Woloszyn A. A polynomial-time approximation scheme for weighted planar graph TSP. In: Proceedings of the ninth annual ACM-SIAM SODA. 1998. p. 33–41.
[14] Deineko VG, Klinz B, Woeginger GJ. Exact algorithms for the Hamiltonian cycle problem in planar graphs. Operations Research Letters 2006;34:269–74.
[15] Fleischer L, Tardos E. Separating maximally violated comb inequalities in planar graphs. Mathematics of Operations Research 1999;24:130–48.
[16] Letchford AN. Separating a superclass of comb inequalities in planar graphs. Mathematics of Operations Research 2000;25:443–54.
[17] Letchford AN, Pearson NA. Exploiting planarity in separation routines for the symmetric traveling salesman problem. Working paper, Department of Management Science, Lancaster University; 2005, submitted for publication.
[18] Rosenkrantz DJ, Stearns RE, Lewis PM. An analysis of several heuristics for the traveling salesman problem. SIAM Journal on Computing 1977;6:563–81.
[19] Burkard RE, Deineko VG, Woeginger GJ. The travelling salesman and the PQ-tree. Mathematics of Operations Research 1998;23:613–23.
[20] Voronoi G. Nouvelles applications des paramétres continus á la théorie des formes quadratiques, Deuxiéme memoire: Recherche sur les parallélloédres primitifs. Journal für Reine und Angewandte Mathematik 1908;3:198–287.
[21] Delaunay B. Sur la sphére vide, Izvestia Akademia Nauk. SSSr, VII Seria. Otdelenie Matemtischeskii i Estestvennyka Nauk 1934;7:793–800.
[22] Okabe A, Boots B, Sugihara K, Chiu SN. Spatial tessellations: concepts and applications of Voronoi diagrams. Chichester: Wiley; 1992.
[23] Preparata FP, Shamos MI. Computational geometry: an introduction. New York: Springer; 1985.
[24] Toussaint GT. The relative neighborhood graph of a finite planar set. Pattern Recognition 1980;12:261–8.
[25] Gabriel KR, Sokal RR. A new statistical approach to geographic variation analysis. Systematic Zoology 1969;18:259–78.
[26] Baker TJ. Three dimensional mesh generation by triangulation of arbitrary point sets. In: Proceedings of the AIAA eighth computational fluid dynamics conference. 1987.
[27] Keil JM, Gutwin CA. The Delaunay triangulation closely approximates the complete graph. Discrete & Computational Geometry 1992;7:13–28.
[28] Chew LP. There are graphs almost as good as the complete graph. Journal of Computer and System Sciences 1989;39:205–19.
[29] Dillencourt MB. A non-Hamiltonian Delaunay triangulation. Information Processing Letters 1987;25:149–51.
[30] Dillencourt MB. Finding Hamiltonian cycles in Delaunay triangulations is NP-complete. In: Proceedings of the fourth conference on computational geometry. Canada; 1994.
[31] Genoud T. Etude du caractère hamiltonien de Delaunays aléatoires. Working Paper. Ecole Polytechnique Federale de Lausanne, Switzerland; 1989.
[32] Levcopoulos C, Krznaric D. The greedy triangulation can be computed from the Delaunay triangulation in linear time. Computational Geometry, Theory and Applications 1999;14:197–220.
[33] Levcopoulos C. Private communication; December 2005.
[34] Das G, Joseph D. Which triangulations approximate the complete graph? In: Proceedings of the international symposium on optimal algorithms. Lecture notes in computer science, vol. 401. Berlin: Springer; 1989. p. 168–83.

[35] Manacher GK, Zobrist AL. Neither the greedy nor the Delaunay triangulation of a planar point set approximates the optimal triangulation. Information Processing Letters 1979;9:31–4.

[36] Lingas A. The greedy and Delaunay triangulations are not bad in the average case. Information Processing Letters 1986;22:25–31.

[37] Reinelt G. TSPLIB—a traveling salesman problem library. ORSA Journal on Computing 1991;3:376–85.

[38] Gutin G, Punnen A. The traveling salesman problem and its variations. Dordrecht: Kluwer Academic Publishers; 2002.