To appear in Proc. ACM-SIAM Symposium on Discrete Algorithms (1997)

On the Maximum Scatter TSP

(Extended Abstract*)

Esther M. Arkin† Yi-Jen Chiang‡ Joseph S. B. Mitchell§
Steven S. Skiena¶ Tae-Cheon Yang‖

Abstract

We study the problem of computing a Hamiltonian tour (cycle) or path on a set of points in order to maximize the minimum edge length in the tour or path. This "maximum scatter" TSP is closely related to the bottleneck TSP, and is motivated by applications in manufacturing (e.g., sequencing of rivet operations) and medical imaging. In this paper, we give the first algorithmic study of these problems, including complexity results, approximation algorithms, and exact algorithms for special cases. In an attempt to model more accurately the real problems that arise in practice, we also generalize the basic problem to consider a more general measure of "scatter" in which points on a tour or path should be far not only from their immediate predecessor and successor, but also from other near-neighbors along the tour or path.

1 Introduction

In this paper we study a new class of traveling salesman problem (TSP) variants that is based on the objective of finding, in an edge-weighted complete graph $G = (S, E)$, a tour (cycle) or a path that is most "scattered." Specifically, the goal is to *maximize* the length of a shortest edge in the tour/path; i.e., to have each point be far from the points that are visited just before or just after it in the tour/path. We refer to this problem as the *maximum scatter TSP*, or the *max-min 1-neighbor TSP*.

More generally, we consider the *max-min m-neighbor TSP* in which the goal is to maximize the minimum distance between any point and all of its "m-neighbors" along the tour/path. An *m-neighbor* of p is a point that is at most m points away from p in the tour/path. The problem, then, is to find a tour/path, $(p_1, \ldots, p_n)$, on the n points of S in order to maximize

$$\min_{j \in \{i-m,\ldots,i-1,i+1,\ldots,i+m\}} d(p_i, p_j),$$

where $d(p_i, p_j)$ denotes the length of edge (p_i, p_j). (The indices in the minimization should be modified appropriately to account for wrap-around effects in cycles, or endpoint effects in paths.)

Motivation. This *maximum scatter TSP* problem arises in some manufacturing processes where it is important to have substantial separation (distance) between consecutive (or *nearly* consecutive) operations on a workpiece. We first encountered the problem in discussions with Boeing, where the problem was that of sequencing the riveting operations when fastening sheets of metal together [9]. To prevent nonuniform deformation of the sheet metal plates that are being joined, it is important to sequence the riveting process so that the distance between one rivet and the next one is large; i.e., the riveting operations should be *scattered*. In operations that involve heating the workpiece, it may be important not just to have each point well separated from its immediate predecessor and successor, but also from its m-neighbors, in order to allow for cooling time in the vicinity of each operation.

The maximum scatter TSP also arises in some medical imaging applications, as has been recently studied by Penavic (private communication). When imaging physiological functions using a Dynamic Spatial Reconstructor (DSR), the radiation sources are placed along the top half of a circular ring, with sensors placed directly opposite, in the bottom half of the ring. The "firing sequence" determines the order in which the sources, and their partnered sensors, are activated, usually in a periodic pattern. The sensors

*The full paper is available at the web site **http://ams.sunysb.edu/~yjc/yjc.html**

†Department of Applied Mathematics and Statistics, SUNY Stony Brook, NY 11794-3600; estie@ams.sunysb.edu. Supported in part by NSF Grant CCR-9504192.

‡Department of Applied Mathematics and Statistics, SUNY Stony Brook, NY 11794-3600; yjc@ams.sunysb.edu. Supported in part by NSF Grant DMS-9312098.

§Department of Applied Mathematics and Statistics, SUNY Stony Brook, NY 11794-3600; jsbm@ams.sunysb.edu. Supported in part by Hughes Research Laboratories and NSF Grant CCR-9504192.

¶Department of Computer Science, SUNY Stony Brook, NY 11794-4400; skiena@cs.sunysb.edu. Supported in part by NSF Grant CCR-9109289 and ONR Award 400x116yip01.

‖Dept. of Computer Science and Statistics, Kyungsung University, 110, DaeYeon-dong, Nam-gu, Pusan, 608-736, KOREA; tcyang@csd.kyungsung.ac.kr. Supported in part by Kyungsung University Research Grants in 1996.

collect intensity data of the energy that passes through the patient, who is placed in the center of the ring. When source i is activated, some amount of scattering occurs, so it is important not to activate sensors of nearby sources (e.g., $i+1, i-1, i+2, \ldots$) soon after i is activated. This motivated Penavic to study firing sequence orderings for some specific geometries of DSR hardware, and motivates our study (in Section 5) of some 1-dimensional versions of the maximum scatter problem.

Related Work. A closely related problem is the *bottleneck Traveling Salesman Problem* (BTSP) in which the goal is to *minimize* the length of a *longest* edge in a Hamiltonian cycle. (See [7].) The BTSP is known to be NP-complete, and no constant factor approximation algorithm can exist, unless P=NP. Assuming the edge lengths satisfy the triangle inequality, there exists an approximation algorithm to produce a tour whose longest edge has length that is at most twice the optimal, and this is best possible [8]. By similar techniques we obtain similar hardness results for the maximum scatter TSP; however, our approximation algorithm is very different from that given in [8], and neither algorithm works for the other problem.

Another variant of the TSP that is potentially related to our work is the MAX TSP, in which the goal is to find a tour whose length is as long as possible. Several approximation algorithms exist for the MAX TSP (without any assumptions on the edge lengths), including a naive algorithm that achieves a 2/3 approximation factor, and a recent algorithm of [6] that achieves a 5/7 approximation factor. However, neither of these algorithms works for our maximum scatter problem. Also, there exists a polynomial time approximation scheme for the Euclidean version of the MAX TSP [2]. Nevertheless, as with our maximum scatter TSP for points in the plane (and Euclidean distances), the complexity of the Euclidean version of the MAX TSP is open [1].

The most directly related work on the maximum scatter TSP has been done by Penavic, who studied the optimal firing sequences for a DSR application in which all sources are equally spaced. Our one-dimensional results generalize this previous work.

Summary of Results.

- We show that the maximum scatter TSP is, in general, NP-complete and that no constant-factor approximation algorithm exists, unless P=NP. We do not yet know the complexity of the *geometric* maximum scatter TSP problem for points in the plane (and Euclidean distances); we conjecture that it is NP-hard.
- In the case that distances obey the triangle inequality, we give approximation algorithms guaranteed to get within factor 2 of optimal for the maximum scatter TSP, for both the *tour* (cycle) and *path* versions of the problem. Further, we prove that this factor is best possible.
- We also give constant-factor approximation results on the max-min 2-neighbor TSP in the case that distances obey the triangle inequality, for both the cycle (factor 64) and path (factor 32) versions of the problem. Our approach utilizes novel methods of applying matching techniques, together with metric properties of the distance matrix, and may be of independent interest.

  Our methods extend also to yield constant-factor approximations for the max-min m-neighbor TSP, for constant $m > 2$; these results are deferred to the full paper.
- We give algorithms that find the optimal solution to the max-min 1-neighbor TSP exactly, in some special cases. In particular, we obtain linear-time (optimal) exact algorithms for the case of points on a line, or on a circle, with Euclidean distances, for both the cycle and path versions of the problems.

Notation. We consider a complete graph, $G = (S, E)$ on a set of sites S, with the full edge set E. We let $d(p, q)$ denote the distance between point p and point q, and we let $\lambda(e) = d(p, q)$ denote the length of edge $e = (p, q)$. The *circle* $\mathcal{C}$ of radius r, centered at $p \in S$ is defined to be the set of points in S that are at distance less than r from p. The *boundary* of $\mathcal{C}$ is defined to be those points of S that are at distance exactly r from p.

2 Hardness Results

We state the following two results, whose proofs, omitted here, are similar to those for the BTSP.

THEOREM 2.1. *There can exist no polynomial-time, constant-performance-bound approximation algorithm for an arbitrary instance of the maximum scatter TSP (cycle or path version), unless P=NP.*

THEOREM 2.2. *There can exist no polynomial-time algorithm for the maximum scatter TSP (cycle or path version) in a graph satisfying the triangle inequality with a performance bound smaller than 2, unless P=NP.*

While the above two results parallel the similar facts known about the bottleneck TSP (BTSP), a significant difference between BTSP and maximum scatter TSP can be seen when considering the geometric versions, in

the plane. It can be shown that the BTSP is NP-hard even for points in the plane, using a reduction from Hamiltonian cycle in a grid graph (given in [7]). A tour whose maximum edge length is 1 exists in the complete graph whose edge lengths are given by the Euclidean distances if and only if the grid graph is Hamiltonian. This approach does not work, though, to show hardness of the maximum scatter TSP in the plane; to date, this problem remains open. (In one dimension, the problem is linear-time solvable, as we show in Section 5.)

It is not hard to show that the max-min m-neighbor TSP is at least as hard as the max-min 1-neighbor (scatter) TSP:

THEOREM 2.3. *In a graph satisfying the triangle inequality, for both the cycle and path versions the max-min m-neighbor TSP with $m > 1$ is at least as hard as the max-min 1-neighbor TSP.*

The following is immediate from Theorem 2.3 and Theorem 2.2:

COROLLARY 2.1. *There can exist no polynomial-time algorithm for the max-min m-neighbor TSP (cycle or path version) in a graph satisfying the triangle inequality with a performance bound smaller than 2, unless P=NP.*

3 Max-Min 1-Neighbor TSP

In this section, we consider the maximum scatter (max-min 1-neighbor) TSP, for both the cycle and path versions, on the complete graph $\mathcal{G} = (S, E)$ whose edges are weighted with nonnegative lengths that satisfy the triangle inequality.

We first mention that some of the natural approaches do not work in this case. In particular, the approximation methods for BTSP and MAX TSP both fail for max-min 1-neighbor TSP, as does a greedy approach (select the next site to be the unvisited one that is furthest away).

Now we consider the cycle version of the problem. We make use of the following theorem due to Dirac [3]: If every node of a graph having n nodes ($n \geq 3$) has degree $\geq \lceil \frac{n}{2} \rceil$, then the graph has a Hamiltonian cycle.

Our algorithm is very simple: We consider the edges E in order of increasing length. We delete edges, one by one, starting with the shortest one, and stopping when the deletion of an edge, say $e = (u, v)$ (of length $d_e = \lambda(e) = d(u, v)$), would cause a node (say, v) to have degree *less* than $\lceil \frac{n}{2} \rceil$. Let $\mathcal{G}' = (S, E')$ be the graph that remains.

By Dirac's Theorem, we know that $\mathcal{G}'$ has a Hamiltonian cycle. We can find such a cycle (call it C) in $O(n^2)$ time (see Theorem 3.1). A shortest edge of C has length at least d_e.

Let d^* be the length of a shortest edge of an optimal tour.

LEMMA 3.1. *Let $R \subset S$ be a subset of nodes with $|R| > \lfloor \frac{n}{2} \rfloor$ (resp., $|R| > \lceil \frac{n}{2} \rceil$). Then, in any Hamiltonian cycle (resp., path) on S, there must exist an edge joining two nodes of R.*

Proof. Let $k = \lfloor \frac{n}{2} \rfloor$ and $S_{out} = S - R$. To avoid having any edge joining two nodes of R in a Hamiltonian cycle, all nodes in R must be connected only to nodes in S_{out}, which is impossible since the total degree of the nodes in R is at least $2k + 2$ but that of the nodes in S_{out} is at most $2k$. □

We later also use the following immediate generalization of Lemma 3.1:

LEMMA 3.2. *For any integer $m \geq 1$, let $R \subset S$ be a subset of nodes with $|R| > \lfloor \frac{n}{m+1} \rfloor$ (resp., $|R| > \lceil \frac{n}{m+1} \rceil$). Then, in any Hamiltonian cycle (resp., path) on S, there must exist two nodes of R that are m-neighbors.*

COROLLARY 3.1. *Let $R \subset S$ be a subset of nodes with $|R| = \lfloor \frac{n}{2} \rfloor + 1$. Then, $d^* \leq diam(R)$.*

LEMMA 3.3. $d^* \leq 2d_e$

Proof. Let $R = \{u\} \cup N_v$, where N_v denotes the set of nodes w such that $(v, w) \notin E'$. (By definition, also $v \in N_v$.) By specification of the algorithm, $|S \backslash N_v| \geq \lceil \frac{n}{2} \rceil$, while $|S \backslash R| < \lceil \frac{n}{2} \rceil$. Thus, $|R| = \lfloor \frac{n}{2} \rfloor + 1$, so, from Corollary 3.1, $d^* \leq diam(R)$. But, since all nodes of R are at distance $\leq d_e$ from v, we know, from the triangle inequality, that $diam(R) \leq 2d_e$. Thus, $d^* \leq 2d_e$. □

THEOREM 3.1. *There exists an $O(n^2)$-time approximation algorithm for the maximum scatter TSP that produces a Hamiltonian tour whose shortest edge is at least $\frac{1}{2}$ times optimal.*

Proof. The approximation factor follows from Lemma 3.3. The $O(n^2)$ running time can be accomplished as follows. First, in overall time $O(n^2)$, we find for each node the median weight edge among those edges incident on it. By taking the minimum of these n medians, we find the length of the critical edge, e, and therefore can easily identify the edges E' of the graph $\mathcal{G}'$. Our algorithm now resembles a constructive version of Dirac's proof. We start with a Hamiltonian cycle in the complete graph; this is trivially found. Then, we delete edges (u, v), one by one, that are *not* in E'. With each deletion, we modify the existing Hamiltonian cycle, $u = v_1, v_2, \ldots, v_{n-1}, v_n = v, u$, swapping out edges (u, v) and (v_i, v_{i+1}), and swapping in edges (u, v_{i+1}) and (v_i, v), where $i \in S \cap T$ is an index in the intersection of $S = \{i : (u, v_{i+1}) \in E'\}$ and $T = \{i : (v_i, v) \in E'\}$. (As in Dirac's proof, $S \cap T$ must be nonempty, since $|S \cup T| < n$ (since $n \notin S, T$) and

$|S \cup T| + |S \cap T| = |S| + |T| = deg(u) + deg(v) \geq n$.) By representing the sets S and T with bit vectors, all of the bookkeeping can be done in overall time $O(n^2)$. □

Now we consider the path version of the problem. One simple way is to add one *fake point* Q very far away from all points of S that does not affect the max-min edge length at all, and then solve the cycle problem. When we get the answer that is a cycle, we simply cut the cycle open at Q to get a path, and then remove Q. Notice that using Lemma 3.1, we have a corollary for the *path version* similar to Corollary 3.1: Any subset R of S with $|R| = \lceil \frac{n}{2} \rceil + 1$ has $d^* \leq diam(R)$, where d^* is now the optimal max-min edge length of Hamiltonian paths. When we apply our cycle algorithm, the problem size is $n' = n + 1$, for which we take the size of R to be $\lfloor \frac{n'}{2} \rfloor + 1 = \lceil \frac{n}{2} \rceil + 1$, which is exactly the desired size of R for the *path* version.

COROLLARY 3.2. *There exists an $O(n^2)$-time approximation algorithm for the maximum scatter TSP that produces a Hamiltonian path whose shortest edge is at least $\frac{1}{2}$ times optimal.*

4 Max-Min 2-Neighbor TSP

In this section, we consider the cycle and path versions of the max-min 2-neighbor TSP, in which we not only want edges of the tour/path to be long, but we also want each point to be "far" from its 2-neighbors along the tour/path.

Our main results are constant-factor approximation algorithms, which utilize novel methods of applying matching techniques, together with metric properties of the distance matrix, to search for "large" triangles (cycles of three nodes) that can then be concatenated to form provably good tours and paths. Our overall strategy consists of the following steps:

1. Create $\lfloor n/3 \rfloor$ vertex-disjoint *large* triangles whose vertices are the points in S. Here, by "large" we mean that all three sides of the triangles have lengths at least a constant fraction of the upper bound, $2r$ (see Lemma 4.1).

   (Of course, if n is not a multiple of 3, there are one or two points not part of any triangle, but this complication can be handled, as we see later.)

2. Concatenate these large triangles together in order to create a Hamiltonian path or cycle such that the distances between any 2-neighbors are large.

4.1 Constructing Large Triangles. We describe how to construct large triangles. We assume without loss of generality that there are no two distances between points that are exactly the same. This is not restrictive since we can perturb the edge lengths if there is a tie.

Let r be the radius of a smallest circle, $\mathcal{C}$, centered at one of the n points that contains exactly $\lfloor n/3 \rfloor$ points in its interior (including the center point), and one additional point on its boundary. ($\mathcal{C}$ and r are easily found in $O(n^2)$ time.)

LEMMA 4.1. *The max-min distance in the cycle version of a 2-neighbor TSP is at most $2r$.*

Proof. Immediate from Lemma 3.2. □

For simplicity of exposition, we now assume that n is a multiple of 3. We will handle the other cases later. After computing r and $\mathcal{C}$, we use Algorithm *Big-Triangles* to construct $n/3$ vertex-disjoint large triangles, using n points of S as vertices, as follows.

Algorithm. *Big-Triangles*

1. The point-point matching phase

Create a bipartite graph $G = (V_1, V_2, E)$, where the nodes V_1 are all of the points inside the circle $\mathcal{C}$, and V_2 are all of the points outside the circle, including the point on the boundary. Edge set E consists of "long" edges from V_1 to V_2, between two nodes whose distance is at least r. Note that $|V_1| = n/3$ and $|V_2| = (2n)/3$. Perform a maximum cardinality matching on G. By Lemma 4.2 (below), all nodes of V_1 are matched. For each matched pair, create a segment e by connecting the two points. Clearly, each such e is of length at least r.

LEMMA 4.2. *There is a matching in G, in which all nodes of V_1 are matched.*

Proof. We use Hall's Theorem [5], which states: A "complete" matching of V_1 into V_2 exists if and only if for every subset A of V_1, $|A| \leq |\Gamma(A)|$, where $\Gamma(A)$ is a set of those vertices of V_2 which are adjacent to at least one vertex in A.

We assume, by way of contradiction, that there exists a set $A \subseteq V_1$ for which $|A| > |\Gamma(A)|$. Consider now the nodes of $V_2 \setminus \Gamma(A)$. The cardinality of this set satisfies

$$|V_2 \setminus \Gamma(A)| = |V_2| - |\Gamma(A)| > |V_2| - |A| \geq n/3,$$

since $|A| \leq |V_1| = n/3$ and $|V_2| = (2n)/3$. Furthermore, by definition, all of these points are at distance less than r from all points of A. A circle, centered at an arbitrary point of A, of radius r contains this center point as well as all of the points of $V_2 \setminus \Gamma(A)$, i.e., at least $n/3 + 2$ points in total, in its *interior*, contradicting the original choice of r. □

2. The segment-point matching phase

We now create a bipartite graph $H = (W_1, W_2, \mathcal{E})$, where the nodes W_1 are the edges (segments) e that form the point-point matching created in Phase 1, and the nodes W_2 are the points of V_2 that are *not* matched in Phase 1. Note that $|W_1| = |W_2| = n/3$. The edge set $\mathcal{E}$ is specified as follows: there is an edge between node $e \in W_1$ and point $p \in W_2$ if and only if p is separated from *both* endpoints of the segment e by a distance at least $r/4$. Perform a *maximal* (rather than maximum cardinality) matching on H. Now, for each matched pair, e and p, create a triangle by connecting p with the two endpoints of e. If the matching is a perfect matching, stop (there are $n/3$ triangles whose edge lengths are at least $r/4$); otherwise go to the next phase.

3. The swapping phase

Let P be the set of points not yet in any triangle, and E' the set of segments (created in Phase 1) connecting points of S that are not yet in any triangle. Note that $|P| = |E'| = n/3 - T$, where T is the size of the matching in Phase 2. Define $c(p)$ to be the circle centered at point $p \in P$ with radius $r/4$.

LEMMA 4.3. *There is a unique partition of P into two subsets P_1 and P_2 (i.e., $P_1 \cup P_2 = P$ and $P_1 \cap P_2 = \emptyset$), where one of them may be empty, such that for each P_i, $i = 1, 2$, there is a circle C_i of radius $r/2$ that encloses all points in P_i. Moreover, denote $\bigcap_{p \in P_i} c(p)$ by $ker(P_i)$. Then $ker(P_i)$ is nonempty (if P_i is nonempty) and is enclosed in C_i, and each segment in E' has one endpoint in $ker(P_1) \in C_1$ and the other endpoint in $ker(P_2) \in C_2$.*

Proof. Consider an arbitrary segment $e = (v_1, v_2) \in E'$. Define P_1 (resp., P_2) to be the set of those $p \in P$ such that v_1 (resp., v_2) is inside (but not on the boundary of) $c(p)$. We claim that P_1 and P_2 form a partition of P. First, by the definition of P and E', for all points $p \in P$, either v_1 or v_2 must be inside $c(p)$, or otherwise e would have been matched in Phase 2. Also, $c(p)$ cannot enclose both v_1 and v_2, since $c(p)$ is of radius $r/4$ and the length of e is at least r. This completes the proof of the claim.

Clearly, for $i = 1, 2$, if P_i is nonempty, $v_i \in ker(P_i) \neq \emptyset$. Also, any point q_i in $ker(P_i)$ is within a distance less than $r/4$ to all points in P_i, and thus the circle c_i centered at q_i with radius $r/4$ encloses all points in P_i. Clearly, another circle C_i centered at q_i with radius $r/2$ encloses all points in P_i as well as the entire $ker(P_i)$, since the diameter of $ker(P_i)$ is less than $r/2$. Note the difference between c_i and C_i.

We have shown that each segment $e \in E'$ defines a partition of P. Now we show that the partitions defined by all such segments are actually the same. We call (P_1, P_2) a *1-partition* if some P_i is empty, and a *2-partition* otherwise. Clearly, a 1-partition is unique, namely $P_1 = P$. Suppose $e \in E'$ defines a partition (P_1, P_2) and there is a different partition (P_1', P_2') of P defined by another segment $e' \in E'$. We consider two cases: (1) one of the two partitions, say (P_1', P_2'), is a 1-partition, and (2) both are 2-partitions. In both cases, (P_1, P_2) is a 2-partition. Without loss of generality, assume p_1 is in both P_1 and P_1', and some p_2, originally in P_2, is now in P_1'. This means that there exist points p_2 inside circle c_2 and p_1 inside circle c_1 such that the interiors of $c(p_1)$ and of $c(p_2)$ intersect, i.e., $d(p_1, p_2) < r/2$. This is a contradiction, because letting c_i centered at w_i, $i = 1, 2$, where w_1 and w_2 define the longest distance between a point in $ker(P_1)$ and a point in $ker(P_2)$, the centers of c_1 and of c_2 are separated by a distance $d(w_1, w_2) \geq \lambda(e) \geq r$, but each c_i is of radius $r/4$, and thus the closest distance between c_1 and c_2 is at least $r/2$. Having established the uniqueness of the partition, it is immediate to see that if (P_1, P_2) is a 1-partition, then *each* segment $e \in E'$ has exactly one endpoint in $ker(P_1)$; otherwise, *each* e has one endpoint in $ker(P_1)$ and the other endpoint in $ker(P_2)$. □

For simplicity, unless otherwise stated, the following discussions assume that (P_1, P_2) is a 2-partition. For $i = 1, 2$, let C_i be centered at an arbitrary segment endpoint $q_i \in ker(P_i)$, where the segment is in E'. Recall that C_i is of radius $r/2$. Define circle R_i to be the one centered at q_i with radius r. Call a triangle $\triangle abc$ constructed from Phase 2 *good* with respect to P_i if a, b and c are all outside or on the boundary of R_i, and *bad* otherwise.

LEMMA 4.4. *Let $\triangle abc$ be a good triangle with respect to P_i, $i = 1$ or 2, whose edge length is at least $r/4$, denoted by δ. Then we can pick $\triangle abc$ together with a point $x \in P_i$ and a segment $e = (p, q) \in E'$ to construct two triangles using points x, p, q, a, b, c as vertices, so that the shortest edge length is at least $\delta/2 = r/8$.*

Proof. By Lemma 4.3, x and one endpoint of e, say p, are both inside C_i. Since a, b and c are all outside R_i, they are all away from each of x and p by distances at least $r/2$, since the shortest distance between the boundaries of C_i and of R_i is $r/2$. Also, recall that $d(p, q) \geq r$. Without loss of generality, let b be the vertex of $\triangle abc$ closest to q. If $d(b, q) \geq \delta/2$, then $\triangle pqa$ and $\triangle xbc$ satisfy the conditions of the lemma. Otherwise, $d(q, a) \geq \delta/2$, since $d(q, a) + d(q, b) \geq d(a, b) \geq \delta$. Then again we can construct $\triangle pqa$ and $\triangle xbc$. □

Phase 3 proceeds by finding C_i and R_i, $i = 1, 2$, then perform the swapping operation described in Lemma 4.4 for all points in P_1 (*swapping stage 1*), and

then perform the same operation for all points in P_2 (*swapping stage 2*). By Lemma 4.5 (given below), there are always enough good triangles to complete the swapping operations, so that all points are now vertices of $n/3$ triangles. In addition, the smallest edge length of all these triangles is at least $r/8$.

LEMMA 4.5. *There are always enough good triangles to complete the swapping operations on all points in P_1 and all points in P_2. Also, after completion of Algorithm* Big-Triangles, *the smallest edge length of all $n/3$ triangles is at least $r/8$.*

Proof. We first consider swapping stage 1. At the end of Phase 2, suppose there are T triangles, ℓ unmatched segments (and points), $|P_1| = \ell_1$, $|P_2| = \ell_2$, then we have $T + \ell = n/3$, and $\ell_1 + \ell_2 = \ell$. Let t_1 be the number of bad triangles with respect to P_1. If t_1' is the total number of vertices of these triangles that are inside R_1, then $t_1' \geq t_1$, since each bad triangle has at least one vertex inside R_1. Observe that R_1 is a circle of radius r. By the definition of r, the *interior* of circle R_1 contains at most $n/3$ points; these points include ℓ endpoints from unmatched segments, ℓ_1 points of P_1, and t_1' vertices from bad triangles. Therefore we have $n/3 \geq \ell + \ell_1 + t_1' \geq \ell + \ell_1 + t_1$. Let $T_{1,good}$ be the number of good triangles with respect to P_1. Then $T_{1,good} = T - t_1 = (n/3 - \ell) - t_1 \geq (n/3 - \ell) - (n/3 - \ell - \ell_1) = \ell_1$, i.e., there are enough good triangles for the points in P_1. For swapping state 2, the situation is the same except that now $\ell = \ell_2$; this is the same as having a 1-partition to start with, and the same counting argument applies. Finally, the smallest edge length of triangles at the end of Phase 2 is at least $r/4 = \delta$. During Phase 3, let $\triangle abc$ be a good triangle for P_1, and in swapping stage 1 we swap $\triangle abc$ with a segment $e = (p, q) \in E'$ and a point $x \in P_1$ to produce $\triangle pqa$ and $\triangle xbc$, as shown in the proof of Lemma 4.4. Notice that $\triangle pqa$, with edge lengths at least $\delta/2$, is now *bad* for P_2 (and also for P_1) and will not be used for swapping in swapping stage 2. Also, $\triangle xbc$ has edge lengths at least δ rather than $\delta/2$. Applying Lemma 4.4 again in swapping stage 2, the smallest edge length of the final triangles is still at least $\delta/2 = r/8$. □

Now, for n not being a multiple of 3, i.e., $n = 3k + l$, $l = 1$ or 2, we perform exactly the same operations. Note that circle $\mathcal{C}$ now contains k points in its *interior*, and $2k + l$ points are outside $\mathcal{C}$, including the one on the boundary. Applying Algorithm *Big-Triangles* then gives k large triangles, using up all k points in the interior of $\mathcal{C}$, with one or two points left over outside or on the boundary of $\mathcal{C}$.

To analyze the complexity, we note that Phase 1 and Phase 2 are just maximum cardinality matching and maximal matching for bipartite graphs, which can be done in time $O(\sqrt{n}m) = O(n^{2.5})$ [4] (where $m = O(n^2)$ is the number of edges in the graphs). In Phase 3, we construct C_i and R_i in $O(1)$ time, identify good/bad triangles and perform local swappings, in $O(n)$ time. Finally, as analyzed before, radius r and circle $\mathcal{C}$ are computed in $O(n^2)$ time.

THEOREM 4.1. *Given a complete graph on set S of n points satisfying triangle inequality, there exists an algorithm that constructs in $O(n^{2.5})$ time $\lfloor n/3 \rfloor$ triangles from points in S such that the smallest edge length of the triangles is at least $r/8$, where r is the radius of a smallest circle centered at a point of S, containing exactly $\lfloor n/3 \rfloor$ points in its interior, and one additional point on its boundary.*

4.2 Concatenating the Triangles. We first show how to concatenate the triangles into a Hamiltonian path, and then show how to make a Hamiltonian cycle.

THEOREM 4.2. *Given a list of triangles $\Delta_1, \Delta_2, \ldots, \Delta_k$ whose minimum edge length is δ, the triangles can be chained into a path such that any two nodes that are 2-neighbors in the path are at distance at least $\delta/2$ from each other.*

In fact, the order in which the triangles are placed in the path is arbitrary, so that any permutation on the triangles is possible. This will be used to complete a cycle.

To prove the theorem, we can arrange the order in which the vertices of a triangle is visited, according to the following lemma:

LEMMA 4.6. *We are given a triangle $\triangle abc$ whose nodes are visited in that order by a path, and a triangle $\triangle xyz$ to be added after $\triangle abc$, such that all (6) edges in both triangles are of length at least δ. Then the order of the nodes of triangle $\triangle xyz$ can be chosen such that all 2-neighbors in the path are at distance at least $\delta/2$ from each other.*

Proof. Without loss of generality, let x be the node of triangle $\triangle xyz$ closest to a node in triangle $\triangle abc$. If this distance is at least $\delta/2$ then any order of visitation of the nodes of xyz will satisfy the conditions of the lemma. We consider the remaining case, in which this distance is less than $\delta/2$ which we separate into three cases (that are very similar, but not quite symmetric):

1. a is the closest node to x, $d(a, x) < \delta/2$.
2. b is the closest node to x, $d(b, x) < \delta/2$.
3. c is the closest node to x, $d(c, x) < \delta/2$.

Case (1): In this case we claim that $d(b, x) \geq \delta/2$ and $d(c, x) \geq \delta/2$. Assume otherwise, if $d(b, x) < \delta/2$ then $d(a, b) \leq d(a, x) + d(b, x) < \delta$ a contradiction. Also,

we must have that either $d(c,y) \geq \delta/2$ or $d(c,z) \geq \delta/2$, (or possibly both) otherwise $d(y,z) \leq d(c,y) + d(c,z) < \delta$ is again a contradiction. If $d(c,y) \geq \delta/2$, then we arrange the triangles in order $abcxyz$, and by the previous discussion we have that $d(b,x), d(c,x), d(c,y) \geq \delta/2$. Otherwise we arrange the triangles in order $abcxzy$, and obtain that $d(b,x), d(c,x), d(c,z) \geq \delta/2$.

Cases (2) and (3) follow similar reasoning and are omitted here. □

The proof of the theorem is now immediate. Start the path with any triangle, in any order of the vertices, and keep adding new triangles to the path, one at a time, according to the lemma.

Now we can directly apply Theorem 4.2 to obtain a Hamiltonian path, if n is a multiple of 3. Combining Theorem 4.2 and Theorem 4.1, the shortest distance among all 2-neighbors in the path is at least $r/16$. Recall that, from Lemma 3.2 and the definition of r given in Section 4.1, the objective value of an optimal Hamiltonian path for max-min 2-neighbor problem is at most $2r$.

THEOREM 4.3. *Given a complete graph on set S of n points satisfying triangle inequality, where n is a multiple of 3, there exists an algorithm that constructs in $O(n^{2.5})$ time a Hamiltonian path on S whose objective value for the max-min 2-neighbor problem is at least $\frac{1}{32}$ times the optimal.*

Now consider the case in which n is *not* a multiple of 3. Let $k = \lfloor n/3 \rfloor$, and $n = 3k + l$, $l = 1$ or 2. Recall from Section 4.1 that at the completion of Algorithm *Big-Triangles*, we have k triangles and 1 or 2 left-over points outside or on the boundary of circle $\mathcal{C}$ with radius r. We will combine such left-over points with existing triangles, and then perform the concatenation.

For each left-over point p not being a vertex of the triangles, we claim that we can find an existing triangle $\triangle abc$ (with edge length at least δ $(= r/8)$) such that p is away from all a, b, and c by distances at least r $(> \delta)$. We call $\triangle abc$ a *good* triangle for p.

Consider the *interior* of circle C centered at p with radius r: there are at most k points in the interior, including p. Thus at most $k-1$ triangles can have some of its vertices in the interior of C, meaning that there is always some triangle $\triangle abc$ whose vertices are all outside (possibly on the boundary of) C. Therefore points a, b and c are all away from p by distances at least r. We create a special "triangle" $abcp$.

Now consider the case where $n = 3k + 2$. Let p and q be two left-over points. We know that each has a "good" triangle. If each has a distinct good triangle we obtain two special "triangles" each with four vertices. Otherwise, p and q have only one common good triangle, $\triangle abc$. We claim that in this case $d(p,q) \geq r > \delta$, otherwise a circle centered at p of radius r contains at least one vertex of each of the $k-1$ other large triangles, as well as q, contradicting the choice of r. Thus we obtain one special "triangle" with five vertices, $abcpq$, pairwise far away by distance at least δ.

We treat $abcp$ (resp. $abcpq$) as a special triangle in the triangle concatenating process: to concatenate it with its predecessor triangle, we treat it as triangle $\triangle abc$. The predecessor will fix a specific ordering of points a, b, and c to be visited. We then visit p (and q). To concatenate a triangle after it, consider only p (q) and its two previously visited points as a triangle, and apply the same lemma (Lemma 4.6).

LEMMA 4.7. *Given a complete graph on set S of n points satisfying triangle inequality, there exists an algorithm that constructs in $O(n^{2.5})$ time a Hamiltonian path on S whose max-min 2-neighbor distance is at least $\frac{1}{32}$ times the optimal max-min 2-neighbor distance of a Hamiltonian cycle.*

Finally, we show how to "close" the path into a tour (see the full paper for the proof):

THEOREM 4.4. *Given a complete graph on set S of n points satisfying triangle inequality, there exists an algorithm that constructs in $O(n^{2.5})$ time a Hamiltonian cycle on S whose max-min 2-neighbor distance is at least $\frac{1}{64}$ times the optimal.*

4.3 Paths When n is not a Multiple of 3. We already gave an algorithm to construct a Hamiltonian path whose objective value for the max-min 2-neighbor problem is at least $\frac{1}{32}$ times the optimal, when n is a multiple of 3 (Theorem 4.3). Now we give algorithms achieving the same factor for the remaining cases, where $n = 3k + l$, $l = 1, 2$. Surprisingly, the case of $n = 3k + 2$ can be easily solved, but the case of $n = 3k + 1$ is much more difficult, and we overcome the difficulty by refining our Algorithm *Big-Triangles* and utilizing metric properties of the distance matrix. Also worth noting is that in the last case of our algorithm for $n = 3k+1$, we actually produce an *optimal* Hamiltonian path.

One major difference between the cycle and path versions, when n is not a multiple of 3, is on the upper bounds for the optimal objective values. Let r be the radius of a smallest circle, $\mathcal{C}$, centered at one of the n points that contains exactly $\lceil n/3 \rceil = k + 1$ points in its interior (including the center point), and one additional point on its boundary.

LEMMA 4.8. *The max-min distance in the path version of a 2-neighbor TSP is at most $2r$.*

Proof. Immediate from Lemma 3.2. □

Notice that $\mathcal{C}$ now contains $k+1$ points in its *interior*, as opposed to just k points in the interior in the cycle version given in Lemma 4.1.

When $n = 3k+2$, we add one *fake point* Q far away from all n points in S, and then apply Algorithm *Big-Triangles* on this new set of $n' = 3k+3$ points. The circle $\mathcal{C}$ is the smallest one containing $n'/3 = k+1$ points in its interior and one additional point on the boundary, as desired. Suppose $\triangle Qyz$ is the (big) triangle containing Q produced by Algorithm *Big-Triangles*. Then in applying Theorem 4.2 to chain the $k+1$ triangles into a Hamiltonian path, we start with $\triangle Qyz$, visiting the vertices in that order, and chain the remaining triangles in an arbitrary order. Finally, we remove Q and obtain a Hamiltonian path on S with approximation factor $\frac{1}{32}$.

When $n = 3k+1$, we apply the same trick, adding two fake points Q and Q' far away from all n points in S as well as from each other, and apply our algorithm on this new set of $n' = 3k+3$ points. Again circle $\mathcal{C}$ and its radius r are as desired. If we can make sure that Q and Q' are in the same triangle, say $\triangle QQ'z$, among the $k+1$ triangles constructed by Algorithm *Big-Triangles*, then we can again chain the triangles by visiting Q, Q' and z first (in that order), followed by the remaining triangles, and finally remove Q and Q' from the resulting path. This will give a desired Hamiltonian path on S.

In the rest of this section, we show how to modify Algorithm *Big-Triangles* so that it constructs $k+1$ triangles whose shortest edge length is at least $r/8$, such that Q and Q' are in the same triangle. There is one case, however, in which this is not achieved, but instead we are able to produce an optimal path for this case.

Let $a \in S$ be the center point of $\mathcal{C}$. First, we want Algorithm *Big-Triangles* to produce $\triangle QQ'a$ at the end of Phase 2. This can be done as follows: In Phase 1 we match $a \in V_1$ with $Q \in V_2$ but leave $Q' \in V_2$ unmatched, and in Phase 2 we match segment (Q, a) with point Q' first and then complete the matching. We have to show that there is still a complete matching in Phase 1.

LEMMA 4.9. *In Phase 1 of Algorithm* Big-Triangles*, there is a matching in G in which all nodes of V_1 are matched, with $a \in V_1$ and $Q \in V_2$ being a matched pair and $Q' \in V_2$ being left unmatched.*

Proof. Let $V_1' = V_1 \setminus \{a\}$ and $V_2' = V_2 \setminus \{Q, Q'\}$. We want to match all nodes in V_1' with nodes in V_2'. We use the same proof of Lemma 4.2, except that now $|A| \le |V_1'| = n'/3 - 1$, and $|V_2'| = (2n')/3 - 2$. Thus $|V_2' \setminus \Gamma(A)| > |V_2'| - |A| \ge n'/3 - 1$, i.e., $|V_2' \setminus \Gamma(A)| \ge n'/3$. Again, any circle centered at an arbitrary point of A with radius r contains at least $n'/3 + 1 = k+2$ points in its *interior*, contradicting the original choice of r. □

In Phase 3, we have to make sure that Q and Q' still stay in the same triangle after swapping operations. A natural attempt to achieve this is to try to show, by modifying the argument of Lemma 4.5, that there are still enough good triangles to perform the swappings, even *without using* $\triangle QQ'a$ during the swappings. Unfortunately this method does not work, and thus we have to employ additional techniques to perform careful swappings.

Recall from Lemma 4.3 that at the beginning of Phase 3, the set P of left-over points is partitioned into (P_1, P_2) by the set E' of left-over segments, where (P_1, P_2) is either a 2-partition or a 1-partition. Also, the triangles constructed at the end of Phase 2 have edge lengths at least $r/4$.

LEMMA 4.10. *If (P_1, P_2) is a 2-partition, then we can complete all swapping operations on P_1 and on P_2 in Phase 3 of Algorithm* Big-Triangles*, constructing $k+1$ triangles with edge lengths at least $r/8$, such that Q and Q' stay in the same triangle.*

Proof. See the full paper. □

A nice property for (P_1, P_2) being a 2-partition is that the points of P_1 and P_2 are all outside or on the boundary of $\mathcal{C}$, and they also "confine" the endpoints of left-over segments, so that the endpoints of the left-over segments are far enough from a. When (P_1, P_2) is only a 1-partition, we do not have this nice property, and a more delicate technique is needed.

THEOREM 4.5. *If (P_1, P_2) is a 1-partition with $P_1 = P$, then we can either complete all swapping operations on P_1 in Phase 3 of Algorithm* Big-Triangles*, constructing $k+1$ triangles with edge lengths at least $r/8$, such that Q and Q' stay in the same triangle, or otherwise we can construct an optimal max-min 2-neighbor Hamiltonian path on S.*

Proof. Let R_1 and C_1 be centered at any endpoint $v_1 \in ker(P_1)$ of a left-over segment. If there is such v_1 inside $\mathcal{C}$, then $d(a, v_1) < r$. The conditions of the lemma are satisfied in this case, since letting R_1 and C_1 be centered at this v_1 means that a is inside R_1, i.e., $\triangle aQQ'$ is bad for P_1 and thus is not touched in the swapping operations on P_1. So let us assume that all such endpoints v_1 are outside or on the boundary of $\mathcal{C}$ and thus $\triangle aQQ'$ is good for P_1. Let R_1 and C_1 be centered at any arbitrary such v_1, and call this center d. Let R_1' be a circle centered also at d with radius $(5r)/8$. Now we *redefine* a triangle to be good/bad for P_1 using R_1', namely, a triangle is *good* for P_1 if no vertex is inside R_1', and *bad* otherwise. The swapping lemma (Lemma 4.4) still works with this new definition of good/bad triangles, since the shortest distance between the boundaries of R_1' and of C_1 is at least $(5r)/8 - r/2 =$

$r/8$. In the following, good/bad triangles are all defined by using R'_1. We omit here the highly technical proofs of the claims; see the full paper. We consider two cases:

Case (I). There are less than $n'/3$ points inside R'_1 or some bad triangle has more than one vertex inside R'_1.

CLAIM 4.1. *There are at least $|P_1|+1$ good triangles for P_1 and thus $\triangle aQQ'$ can be left untouched in the swapping operations on P_1, i.e., we can perform the desired swappings on P_1.*

Case (II). Otherwise, R'_1 has exactly $n'/3$ points in its *interior* and each bad triangle for P_1 has exactly one vertex inside R'_1.
This means that there are exactly $\ell = |P_1|$ good triangles, *including* $\triangle aQQ'$. There are three subcases.

(II.1) There exists some point s inside $\mathcal{C}$, $s \neq a$, such that $d(s,a) \geq r/8$.

CLAIM 4.2. *We can perform the desired swappings on P_1.*

(II.2) All $n'/3$ points inside $\mathcal{C}$ (including a) are within distances less than $r/8$ from a, but there exists some point t inside R'_1, $t \neq d$, such that $d(t,d) \geq r/8$.

CLAIM 4.3. *The interiors of $\mathcal{C}$ and of R'_1 contain two disjoint clusters, each with $n'/3$ points, and the distance between any two points in different clusters is larger than $(7r)/8$.*

CLAIM 4.4. *We can perform the desired swappings on P_1.*

(II.3) All $n'/3$ points inside $\mathcal{C}$ (including a) are within distance $< r/8$ from a and all $n'/3$ points inside R'_1 (including d) are within distance $< r/8$ from d.

CLAIM 4.5. *We can construct an optimal max-min 2-neighbor Hamiltonian path on S.*

This completes the proof of Theorem 4.5. □

THEOREM 4.6. *Given a complete graph on set S of n points satisfying triangle inequality, there exists an algorithm that constructs in $O(n^{2.5})$ time a Hamiltonian path on S whose objective value for the max-min 2-neighbor problem is at least $\frac{1}{32}$ times the optimal.*

5 Exact Algorithms: Points on Lines/Circles

We consider now the special cases in which S is a set of n points on a line or on a circle, and distances are Euclidean. In these cases, we are able to obtain linear-time (optimal) algorithms to solve exactly the maximum scatter TSP, for both the cycle and path versions. First, we concentrate on the cycle version of points on a line; we discuss the remaining problems and versions in Section 5.3.

In the following, the n points of S lie on a line, labeled $1, 2, \cdots, n$ from left to right. We distinguish between two cases, depending on whether n is odd or even. Interestingly, the latter case is considerably more complex than the former case. In fact, when n is odd, an optimal cycle can be specified in $O(1)$ time, i.e., without knowing the *positions* of the nodes, a fixed specification based only on the *ordering* of the nodes is guaranteed to be an optimal cycle. Such an optimal cycle can be actually constructed (with the length d^* of a shortest edge of the cycle identified) in $O(n)$ time. When n is even, we show that in linear time we can identify d^*, and after that an optimal cycle can be specified in $O(1)$ time and actually constructed in $O(n)$ time.

We define the following terminology. A subset I of S containing m consecutive nodes is called an *m-interval*. An edge (u,v) is said to *span* the m nodes in the m-interval whose endpoint nodes are u and v (thus each edge spans at least two nodes). An edge is called an *m-edge* if it spans m nodes. We $k = \lfloor \frac{n}{2} \rfloor$ throughout this section. We defer proofs to the full paper.

5.1 Cycle Version: Points on a Line (n Odd)

. We have $n = 2k+1$ in this case. Recall that d^* is the length of a shortest edge of an optimal cycle. By Corollary 3.1, the length of the shortest $(k+1)$-interval is an upper bound for d^*. Now we show that this upper bound is achievable by the following specification of a Hamiltonian cycle:

> Starting from node 1, the "right" edges connect node p to node $[(k+1)+p]$, for $p = 1, 2, \cdots, k$, and the "left" edges connect node $(k+q)$ to node q, for $q = 1, 2, \cdots, k+1$.

It is easy to verify that the above specification gives a Hamiltonian cycle, and that each edge spans at least $k+1$ nodes. Therefore this Hamiltonian cycle is optimal.

THEOREM 5.1. *For 1-dimensional maximum scatter TSP where the problem size n is an odd integer, there exists an algorithm that specifies an optimal cycle in $O(1)$ time, and constructs such cycle in $O(n)$ time.*

5.2 Cycle Version: Points on a Line (n Even)

. We have $n = 2k$ in this case. Without loss of generality, we assume that all k-intervals on S have distinct lengths. This assumption is clearly non-restrictive, since we can always break a tie by viewing the interval whose position is to the left of the other as having a shorter length. We say that two intervals I_1 and I_2 *overlap* if $I_1 \cap I_2 \neq \emptyset$.

LEMMA 5.1. *Let I_1 and I_2 be two overlapping k-intervals. Then in any Hamiltonian cycle on S that avoids having an edge joining two nodes of I_1, there must exist an edge joining two nodes of I_2.*

Proof. See the full paper. □

COROLLARY 5.1. *Any shortest edge in any optimal Hamiltonian cycle on S can span at most k nodes.*

Proof. See the full paper. □

COROLLARY 5.2. *Let I_1, I_2 and I_3 be, respectively, the shortest, the second-shortest, and the third-shortest k-intervals on S, and let e_i be the edge connecting the two endpoint nodes of I_i, $\forall i = 1, 2, 3$. Then $d^* \leq \lambda(e_2)$ if I_2 overlaps with I_1, and $d^* \leq \lambda(e_3)$ otherwise.*

Proof. See the full paper. □

THEOREM 5.2. *Given the shortest k-interval I_1 on S, there exists an algorithm specifying a Hamiltonian cycle C^* that has no edge joining two nodes of I_1 and each of whose edge spans at least k nodes. In addition, if I_1 is the leftmost (rightmost) k-interval, then C^* has no edge joining two nodes of the rightmost (leftmost) k-interval. The length of the shortest edge in C^* achieves the upper bound given in Corollary 5.2 and thus C^* is an optimal cycle.*

Proof. See the full paper. □

Clearly, after scanning all $k+1$ of the k-intervals in $O(n)$ time, we can identify the length d^* of a shortest edge of an optimal cycle, and then apply Theorem 5.2 to specify and construct such a cycle in additional $O(1)$ and $O(n)$ times, respectively.

THEOREM 5.3. *For 1-dimensional maximum scatter TSP where the problem size n is an even integer, there exists an algorithm that computes the shortest length of an optimal cycle in $O(n)$ time, and specifies and constructs an optimal cycle in additional $O(1)$ and $O(n)$ times, respectively.*

5.3 Other Extensions

Now we briefly mention our results of other extensions; the details are given in the full paper. First, The above methods for the cycle version of points on a line can be extended in a fairly straightforward way to yield optimal-time exact algorithms for the cycle version of points on a circle. Now, consider the path version of points on a line. As we did at the end of Section 3, we add one *fake point* Q very far away from all points of S, say at position $-\infty$ so that it does not affect the max-min edge length at all, and then solve the cycle problem. When we get the answer that is a cycle, we simply cut the cycle open at Q to get a path, and then remove Q. When $n = 2k$, adding Q makes the new problem an easy case for cycle ($n' = 2k+1$ is odd), and the upper bound for the path version (every set R of cardinality $\lceil \frac{n}{2} \rceil + 1 = k+1$ must have an edge inside R) is easily achieved by the algorithm (the edges in the path are $(k+1)$- or $(k+2)$-edges). When $n = 2k+1$, adding Q makes the new problem a more complicated case for cycle ($n' = 2k+2$ is even). In this case, a tighter upper bound of the optimal path is needed. In fact, we can show the following lemma similar to Lemma 5.1:

LEMMA 5.2. *Suppose $n = 2k+1$, and let I_1 and I_2 be two $(k+1)$-intervals. Clearly, I_1 and I_2 must overlap. Then in any Hamiltonian path that avoids having any edge joining two nodes of I_1, there must exist an edge joining two nodes of I_2.*

This says that the length of the *second smallest* $(k+1)$-*interval* gives the upper bound for d^*, the optimal objective value. Applying the above algorithm achieves this upper bound, and hence is optimal.

The path version of points on a circle is discussed in the full paper.

Acknowledgements

We thank Samir Khuller for stimulating discussions and comments, which lead to an improvement on the running time of Theorem 3.1. We thank Fritz Scholz and Karel Zikan at Boeing for originally bringing the riveting problem to our attention and for discussions on the practical aspects of riveting in aircraft assembly. We thank Kresha Penavic for discussions on the applications of maximum scatter TSP in medical imaging. We also thank Vera Sacristan and Amitabh Varshney for useful discussions in the early stages of this research.

References

[1] N. Alon, S. Rajagopalan, and S. Suri. Long non-crossing configurations in the plane. In *Proc. 9th Annu. ACM Sympos. Comput. Geom.*, pages 257–263, 1993.

[2] A. I. Barvinok. Two algorithmic results for the traveling salesman problem. *Math. Oper. Res.*, 21:65–84, 1996.

[3] G. A. Dirac. Some theorems on abstract graphs. *Proc. London Math. Soc.*, 2:69–81, 1952.

[4] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.*, 4:507–518, 1975.

[5] P. Hall. On representatives of subsets. *J. London Math. Soc.*, 10:26–30, 1935.

[6] S.R. Kosaraju, J.K. Park, and C. Stein. Long tours and short superstrings. In *Proc. 35th Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 94)*, 1994.

[7] E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, and D.B. Shmoys (eds.). *The traveling salesman problem: A guided tour of combinatorial optimization.* Wiley, New York, 1985.

[8] R.G. Parker and R.L. Rardin. Guaranteed performance heuristics for the bottleneck traveling salesman problem. *Oper. Res. Letters*, 2(6):269–272, 1984.

[9] F. Scholz. Coordination hole tolerance stacking. Technical Report BCSTECH-93-048, Boeing Computer Services, November, 1993. (See also reports 94-017, 94-018, 95-021, and 95-030, by F. Scholz.)