ON THE MAXIMUM SCATTER TSP*

ESTHER M. ARKIN†, YI-JEN CHIANG‡, JOSEPH S. B. MITCHELL§, STEVEN S. SKIENA¶, AND 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.

Key words. traveling salesperson problem, maximum scatter TSP, bottleneck TSP, Hamiltonian cycle/path, approximation algorithms, matching, optimization, Posa conjecture, Dirac's theorem

AMS subject classifications. 68Q25, 68R10, 68U05

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

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

where $d(p_i, p_j)$ denotes the length of edge (p_i, p_j) (*not* the length along the tour/path linking p_i and p_j). The indices in the minimization should be modified appropriately to account for wrap-around effects in cycles, or endpoint effects in paths.

*An extended abstract of an earlier version of this paper appears in *Proc. Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, New Orleans, LA, January, 1997, pages 211–220.

†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 Computer and Information Science, Polytechnic University, 6 MetroTech Center, Brooklyn, NY 11201; yjc@photon.poly.edu. This research was conducted while with Department of Applied Mathematics and Statistics, SUNY Stony Brook. 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 Boeing, 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.

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 [20, 21, 22, 23, 24]. 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 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 6) 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 [17].) 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 [18]. By similar techniques we obtain similar hardness results for the maximum scatter TSP; however, our approximation algorithm is very different from that given in [18], 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 Hassin and Rubinstein [10] that achieves a 5/7 approximation factor. (See also [16], who claimed an earlier approximation factor of 5/7; however, R. Bhatia has recently shown that their algorithm gives a bound of 19/27 instead of the claimed 5/7.) However, neither of these algorithms works for our maximum scatter problem. Also, there exists a polynomial time approximation scheme for geometric (e.g., Euclidean) versions of the MAX TSP [4]. The noncrossing version of geometric MAX TSP is considered in [2]. Very recently, since the original appearance of our own paper [3], Barvinok et al. [5] show that the MAX TSP can be solved exactly in polynomial time, for points in fixed dimension, under a (constant-complexity) polyhedral metric. Most recently, Fekete [8] shows that the Euclidean MAX TSP is NP-hard for the Euclidean metric, in three or more dimensions. His proof also shows that the Euclidean maximum scatter TSP is NP-hard in three or more dimensions.

However, the complexity of the MAX TSP, as well as the maximum scatter TSP and the noncrossing MAX TSP, remains open for points in the Euclidean plane.

The most directly related previous work on the maximum scatter TSP has been done by Penavic [19], 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. See Section 2.
- In the case that distances obey the triangle inequality, we give approximation algorithms (Section 3) 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.
  The methods also extend to give a factor-2 (best factor) approximation for the max-min 2-neighbor TSP, for the cycle version and some cases ($n \neq 3k+1$) of the path version of the problem, by using a very recent result on the *Pósa conjecture* (see Section 3). (These extensions are not practical and are of theoretical interest only.)
- We also give (Section 4) practical 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.
  These methods extend (Section 5) also to yield a constant-factor ($4 \cdot 8^{\lceil m/2 \rceil}$) approximation for the path version of the max-min m-neighbor TSP, for any constant $m > 2$, when n is a multiple of $m+1$.
- We give algorithms (Section 6) 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 (for the path version of points on a circle, we only consider the case in which n is an odd integer or the points are equally spaced).

Notation. We consider a complete graph, $G = (S, E)$ on a set of points S, with the full edge set E. We let $d(p,q)$ denote the distance between point p and point q, and we let $d_e = d(p,q)$ denote the length of edge $e = (p,q)$. We let $diam(R)$ denote the *diameter* of the set $R \subseteq S$; i.e., $diam(R) = \max_{p,q \in R} d(p,q)$. The (open) *disk* $\mathcal{D}_p(r)$ 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 $\mathcal{D}_p(r)$ is the *circle* consisting of points of S that are at distance exactly r from p.

An algorithm is said to yield a *factor α approximation* for a maximization problem if the algorithm is guaranteed to produce a solution whose objective function value is at least $(1/\alpha)$ times the value of an optimal solution.

2. Complexity Considerations: Hardness Results. Our first result demonstrates the importance of restricting our problem to graphs having some structure (e.g., edge lengths obeying triangle inequality).

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.*

Proof. We consider the cycle version of the problem; the same argument applies for the path version as well. Assume to the contrary that such an algorithm exists, and its performance bound is ρ. We show that this algorithm can be used to test whether a graph has a Hamiltonian cycle. Given an arbitrary graph, $G = (S, E)$ construct an instance of maximum scatter TSP (cycle version) on the complete graph on S, with edge lengths $d_e = 1$ if $e \notin E$, and $d_e = 1 + \rho$ if $e \in E$. If G has a Hamiltonian cycle, the maximum scatter TSP has an objective value of $\rho + 1$; thus, the approximation algorithm must yield a tour whose shortest edge length is greater than 1 (i.e., it produces a Hamiltonian cycle in G). If G does not have a Hamiltonian cycle, then the maximum scatter TSP has an objective value of 1, and therefore the approximation algorithm also yields a tour for which at least one edge has length 1. Thus G is Hamiltonian if and only if the alleged approximation algorithm produces a tour of minimum edge length greater than 1. □

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.*

Proof. The proof follows the previous one, except that the weight $\rho+1$ is replaced by weight 2. (Note that the triangle inequality holds in the resulting graph.) □

While the above two results parallel the similar facts known about the bottleneck TSP (BTSP), there is a significant difference in our current knowledge about the BTSP versus the maximum scatter TSP in the case of the geometric versions of these two problems. Specifically, it is easy to see that the BTSP is NP-hard even for points in the plane, using a reduction from Hamiltonian cycle in a grid graph (see [17]): 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 apply, however, to show NP-hardness of the maximum scatter TSP in the plane; to date, this problem remains open. (In one dimension, the problem is solvable in linear time, as we show in Section 6, while in Euclidean 3-space Fekete [8] has recently shown that the problem is NP-hard.)

Next, we 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, the max-min m-neighbor TSP with $m > 1$ is at least as hard as the max-min 1-neighbor TSP, for both the cycle and path versions of the problems.*

Proof. We reduce the problem with $m = 1$ to that with $m > 1$. From the given node set S of the former, we construct another node set S' for the latter. Set S' consists of the original n nodes of S, plus an additional $\lceil \frac{n}{2} \rceil (m-1)$ nodes. Let D and d be the largest and smallest distances between all pairs of nodes in S. We arrange the $\lceil \frac{n}{2} \rceil (m-1)$ additional nodes of S' into $m-1$ clusters $S_1, \ldots, S_{m-1}$, each with $\lceil \frac{n}{2} \rceil$ nodes. Any pair of nodes in the same cluster are within distance less than d, and any pair of nodes that are in different clusters (treating S as a cluster at this time) are separated by a distance larger than D. There must exist an optimal m-neighbor cycle on S' consisting of $\lceil \frac{n}{2} \rceil$ "rounds", where each round contains exactly two nodes of S and one node from each of $S_1, \ldots, S_{m-1}$ (see Fig. 2.1). Observe that for any two nodes in the same cluster (S or S_i), only those in S can possibly be m-neighbors. An optimal 1-neighbor cycle on S can then be obtained from this cycle by skipping the nodes not in S. An optimal m-neighbor path on S' has the same structure except for cutting the cycle open by removing one edge (see Fig. 2.1). Again an optimal 1-neighbor path on S can be obtained in the same way. □

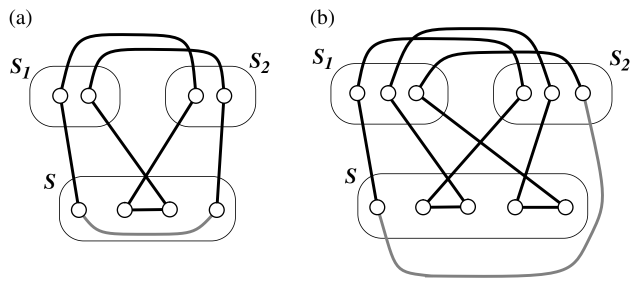

FIG. 2.1. *Examples for the construction in the proof of Theorem 2.3 with $m = 3$: (a) $n = 4$ is even; (b) $n = 5$ is odd. The cycles are given by connecting all (solid and dashed) edges, and the paths are given by removing one edge, e.g., the dashed edge, from the cycles.*

The following is immediate from Theorem 2.3 and Theorem 2.2:

COROLLARY 2.4. *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. Approximating the 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. In the riveting problem, S may be points in the plane, and edges are assigned Euclidean lengths.

We first mention that some of the natural approaches do not work in this case:

(1) The BTSP, in which we must minimize the longest edge (vs. maximizing the shortest edge), has a 2-approximation algorithm [18], which works as follows: First, find a biconnected subgraph of the given graph that minimizes the longest edge; the square of this biconnected graph is Hamiltonian; finally, we trace a Hamiltonian tour in the square of the graph. However, this approach does not yield an approximation for the maximum scatter TSP.

(2) The MAX TSP also has constant factor approximation algorithms (e.g., to get within factor 5/7 of optimal [16]), but again the methods do not yield approximations for the maximum scatter TSP. The reason for this is that the algorithms for MAX TSP generate "long" paths, and then connect them arbitrarily into a cycle, since the added edges will only add to the length of the tour. However, in the scatter problem, adding a very short edge at this stage can drastically reduce the scatter (minimum length edge) obtained, resulting in a tour that is far from optimal.

(3) A natural greedy algorithm (select the next point to be the unvisited one that is furthest away) also fails for the maximum scatter TSP, as there are simple examples showing that it can yield paths or tours that are arbitrarily bad with respect to optimum.

(4) For the geometric version of the maximum scatter TSP in the plane, we have also attempted to use our 1-dimensional (exact) results, by projecting the points onto each of a small number of axes, solving the 1-dimensional

problems on those axes, and then lifting back to two dimensions. However, bad examples exist for this approach as well.

We now present a new approximation method that does yield provably good solutions. We concentrate for now on the cycle version of the problem. We make use of the following theorem due to Dirac [6]: *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.* We employ an algorithmic version of Dirac's proof in our proof of Theorem 3.5 below; to our knowledge, this is the first algorithmic version of Dirac's proof, and it may have interest in its own right.

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 $d_e = d(u,v)$), 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 definition, then, E' includes edge $e = (u,v)$, as well as all other edges of length at least d_e.

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 3.5). 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 $S_{out} = S \setminus R$. 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 number of edges out of R is at least $2k+2$ but such number for S_{out} is at most $2k$. We use the same argument to prove the case of a Hamiltonian path. Letting $k = \lceil \frac{n}{2} \rceil$, we have that the total number of edges out of R is at least $2 + 2[(k+1) - 2] = 2k$ but such number for S_{out} is at most $2(k-1)$. □

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

LEMMA 3.4. *$d^* \leq 2d_e$, where d_e is the length of the edge, $e = (u,v)$, which is the first edge* not *to be deleted in our algorithm, as its deletion would result in the degree of v being $< \lceil \frac{n}{2} \rceil$.*

Proof. Let $R = \{u\} \cup N_v$, where N_v denotes the set of nodes w such that $(v,w) \notin E'$. (Note that $v \in N_v$, since we do not allow self-loops, and that $u \notin N_v$.) Thus, R includes all nodes at distance at most d_e from node v, and node v itself. By specification of the algorithm, $|S \setminus N_v| \geq \lceil \frac{n}{2} \rceil$, while $|S \setminus R| < \lceil \frac{n}{2} \rceil$. Thus, $|R| = \lfloor \frac{n}{2} \rfloor + 1$, so, from Corollary 3.3, $d^* \leq diam(R)$. 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 3.5. *Given a complete graph on set S of n points satisfying triangle inequality, there exists an $O(n^2)$-time approximation algorithm for the maximum scatter TSP on S that produces a Hamiltonian tour whose shortest edge is at least $\frac{1}{2}$ times optimal. The approximation factor is the best possible unless P=NP.*

Proof. The approximation factor and its optimality follow from Lemma 3.4 and

Theorem 2.2. 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 (the $\lceil \frac{n}{2} \rceil$-th longest) 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 $\mathcal{G}'$ by deleting all edges of E shorter than e. 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, which are *not* in E'. With each deletion, we modify the existing Hamiltonian cycle, $u = v_1, v_2, \ldots, v_{n-1}, v_n = v$, swapping out edges (u, v) and (v_i, v_{i+1}), and swapping in edges (u, v_{i+1}) and (v_i, v), where $i \in \mathcal{S} \cap \mathcal{T}$ is an index in the intersection of $\mathcal{S} = \{i : (u, v_{i+1}) \in E'\}$ and $\mathcal{T} = \{i : (v_i, v) \in E'\}$. As in Dirac's proof, $\mathcal{S} \cap \mathcal{T}$ must be nonempty, since $|\mathcal{S} \cup \mathcal{T}| < n$ (since $n \notin \mathcal{S}, \mathcal{T}$) and $|\mathcal{S} \cup \mathcal{T}| + |\mathcal{S} \cap \mathcal{T}| = |\mathcal{S}| + |\mathcal{T}| = deg(u) + deg(v) \geq n$. By representing the sets $\mathcal{S}$ and $\mathcal{T}$ with bit vectors of length n each, where bit j indicates if index j belongs to the set, we can find $i \in \mathcal{S} \cap \mathcal{T}$ in $O(n)$ time. Note that the new edges in the resulting Hamiltonian cycle, (u, v_{i+1}) and (v_i, v), are both in E', i.e., we never introduce new edges that need to be deleted. Therefore we need to perform the deletions at most n times, in overall time $O(n^2)$. $\square$

Now we consider the path version of the problem. One simple way is to add a *fake point* Q very far away from all points of S 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. Notice that using Lemma 3.1, we have a corollary for the *path version* similar to Corollary 3.3: 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 3.6. *Given a complete graph on set S of n points satisfying triangle inequality, there exists an $O(n^2)$-time approximation algorithm for the maximum scatter TSP on S that produces a Hamiltonian path whose shortest edge is at least $\frac{1}{2}$ times optimal. The approximation factor is the best possible unless P=NP.*

4. Approximating the Max-Min 2-Neighbor TSP. We turn now to the problem of approximating the max-min 2-neighbor TSP. We begin by pointing out, as observed by Khuller [12], that our method for the max-min 1-neighbor TSP can be directly extended to work for the cycle version of the max-min 2-neighbor TSP, if the following *Pósa conjecture* (which is a generalization of Dirac's theorem) is true: *If every node of an n-node graph has degree $\geq \lceil \frac{2}{3} n \rceil$, then the graph contains the square (i.e., the second power) of a Hamiltonian cycle.* Here the *k-th power* of a cycle C is the graph obtained by joining every pair of nodes that are k-neighbors in C. Very recently, the Pósa conjecture has been proved for a very large n [14]. Also, the proof leads to a polynomial-time algorithm [12, 13]. Thus we have the following result.

THEOREM 4.1. *Given a complete graph on a set S of n points, with edge lengths satisfying the triangle inequality, there exists a polynomial-time approximation algorithm that constructs a Hamiltonian cycle on S whose objective value for the max-min 2-neighbor problem is at least $\frac{1}{2}$ times the optimal. The approximation factor is the best possible, unless P=NP.*

Notice that the node-degree condition in the Pósa conjecture does not work well for us in the path version unless $n = 3k$; see Lemma 3.2. For $n = 3k+2$, we can again add a fake point Q very far away, construct a cycle and then remove Q to obtain a desirable path. But when $n = 3k + 1$, the same trick requires us to add two fake points Q and Q'. In constructing a cycle, we also need to make sure that Q and Q' are next to each other (so that removing Q and Q' does not break the final path), which might not be easy to do. We do not explore this further here.

COROLLARY 4.2. *Given a complete graph on set S of n points satisfying triangle inequality, where $n = 3k$ or $n = 3k+2$, there exists a polynomial-time approximation algorithm that constructs a Hamiltonian path on S whose objective value for the max-min 2-neighbor problem is at least $\frac{1}{2}$ times the optimal. The approximation factor is the best possible unless P=NP.*

Remark 1. Theorem 4.1 and Corollary 4.2 are of theoretical interests only. The current proof of the Pósa conjecture is quite complex, and uses the *Regularity Lemma* that only holds for a very large n [14]. To turn the proof into an algorithm, it is necessary to use an algorithmic version of the Regularity Lemma that runs in polynomial time. Such algorithm is given in [1]. To get an idea about how large n should be, we quote the following sentences from [1, page 474]:

> "We note that the dependence of $Q(\epsilon, t)$ on ϵ and t, as well as the running time of the algorithm on these two parameters is rather horrible; in fact, $\log^* Q$ is a polynomial (of degree about 20) in $1/\epsilon$ and in t."

As described in [1], the Regularity Lemma is valid for $n > Q$, where $Q = Q(\epsilon, t)$ is a function of given (fixed) parameters $\epsilon > 0$ and t (a positive integer). In typical applications, ϵ is small (and thus $1/\epsilon$ is large) and t is large. Even if we take $t = 2$ and $\epsilon = 1$, this means that n has to be larger than Q where $\log^* Q \approx 2^{20}$!

Remark 2. An algorithmic proof of the following *Seymour conjecture* (a generalization of the Pósa conjecture) would also imply a factor-2 (optimal factor) approximation algorithm for the cycle version of the max-min m-neighbor TSP: If every node of an n-node graph has degree $\geq \lceil \frac{m}{m+1} n \rceil$, then the graph contains the m-th power of a Hamiltonian cycle. Currently the conjecture is proved for the case when every node has degree $\geq (\frac{m}{m+1} + \epsilon)n$, for any $\epsilon > 0$, by using the Regularity Lemma [15].

We turn now to describe our own approximation method for the cycle and path versions of the max-min 2-neighbor TSP. Our method is *not* based in any way on the Pósa conjecture. Although the approximation factors are not optimal, the algorithms are simple and practical. Also, unlike Corollary 4.2, the algorithms work for all cases for the path version as well. Furthermore, the techniques can be extended to obtain an approximation for the max-min m-neighbor TSP, for *any* constant $m > 2$ (see Section 5), not just $m = 2$.

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 for the 2-neighbor problem. Our overall strategy consists of the following steps:

1. Construct $\lfloor n/3 \rfloor$ 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 4.3). (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.

We elaborate these steps in the following sections.

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. (If $d(p_i, p_j) = d(p_{i'}, p_{j'})$, where $i < j$ and $i' < j'$, then we consider edge (p_i, p_j) to be shorter than edge $(p_{i'}, p_{j'})$ if (i, j) is lexicographically less than (i', j').)

Let $\mathcal{C}$ denote a smallest circle, having radius denoted by r, centered at one of the n points such that the corresponding (open) disk contains exactly $\lfloor n/3 \rfloor$ points of S (including the center point) interior to the circle. By definition, then, $\mathcal{C}$ passes through exactly one point of S (which is not, of course, counted as one of the $\lfloor n/3 \rfloor$ interior points).

LEMMA 4.3. *The max-min distance in the cycle version of a 2-neighbor TSP is at most* $2r$.

Proof. Immediate from Lemma 3.2. □

Computationally, r and $\mathcal{C}$ can be found as follows. For each point we find as a candidate the $\lfloor \frac{n}{3} \rfloor$-th shortest edge among all edges incident on the point. The shortest edge among these n candidates defines r and the center of $\mathcal{C}$. Clearly, the computation takes $O(n^2)$ time.

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*. The algorithm takes r, $\mathcal{C}$, and S as input, and constructs $n/3$ vertex-disjoint large triangles using n points of S as vertices. It consists of the following three phases.

1. The point-point matching phase. Consider the bipartite graph $G = (V_1, V_2, E)$, where V_1 is the set of points inside the disk defined by $\mathcal{C}$, and V_2 is the set of points outside the disk, including the point on the boundary $\mathcal{C}$. Edge set E consists of "long" edges from V_1 to V_2, between two nodes whose distance is at least r. Refer to Fig. 4.1. Note that $|V_1| = n/3$ and $|V_2| = (2n)/3$. Perform a maximum cardinality matching on G. By Lemma 4.4 (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 4.4. *There is a matching in G, in which all nodes of* V_1 *are matched.*

Proof. We use Hall's Theorem [9], which states: A "complete" matching of V_1 into V_2 exists if and only if for every subset A of V_1, $|A| \le |\Gamma(A)|$, 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

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

since $|A| \le |V_1| = n/3$ and $|V_2| = (2n)/3$. Furthermore, by definition, all of these points are at distance less than r from all points of A. A disk, 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 $H = (W_1, W_2, \mathcal{E})$, in which the nodes W_1 are the edges (segments) e that form the

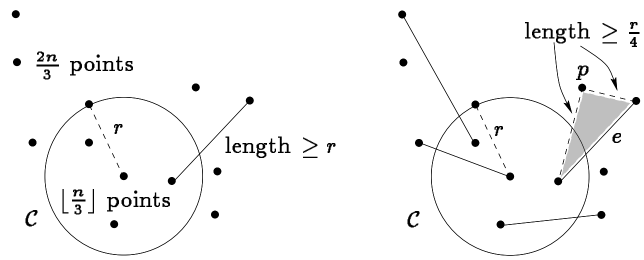

FIG. 4.1. *Left: Point-point matching phase of* Big-Triangles. *Right: Segment-point matching phase.*

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 $r/4$. Refer to Fig. 4.1. 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 "left-over" points not yet in any triangle, and E' be the set of "left-over" 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.

Let $e = (v_1, v_2) \in E'$ be an arbitrary left-over segment, and let $C_i = \mathcal{D}_{v_i}(r/2)$ and $c_i = \mathcal{D}_{v_i}(r/4)$, for $i = 1, 2$, be the (open) disks centered at v_i with radii $r/2$ and $r/4$, respectively. Since e is not matched with any point of P in Phase 2, all points in P are within distances less than $r/4$ from at least one endpoint of e, i.e., $P \subset c_1 \cup c_2$. We define $P_i = c_i \cap P$, $i = 1, 2$; we refer to P_i as a *cluster*. Since e has length at least r, $c_1 \cap c_2 = \emptyset$; thus, (P_1, P_2) partition P. Now, the partition (P_1, P_2) has been defined in terms of the particular edge $e \in E'$; however, the choice of e does not matter as far as the partition is concerned:

LEMMA 4.5. *Each choice of $e \in E'$ results in the same partition of P.*

Proof. Suppose that two distinct edges, $e = (v_1, v_2) \in E'$ and $e' = (v_1', v_2') \in E'$, give rise to two distinct partitions (P_1, P_2) and (P_1', P_2'), respectively. Then, there is some pair of points, $p_1, p_2 \in P$, that is split by one partition but not by the other: without loss of generality, assume that $p_1 \in P_1$ and $p_1 \in P_1'$, while $p_2 \in P_2$ and $p_2 \in P_1'$. Then, $d(p_1, p_2) > r/2$, since $d(v_1, p_1) + d(p_1, p_2) + d(p_2, v_2) \geq d(v_1, v_2) > r$ implies that $d(p_1, p_2) > r - d(v_1, p_1) - d(p_2, v_2) \geq r - (r/4) - (r/4) = r/2$. On the other hand, since $p_1, p_2 \in P_1'$, we get that $d(p_1, p_2) \leq d(p_1, v_1') + d(v_1', p_2) \leq r/4 + r/4 = r/2$, a contradiction. □

LEMMA 4.6. *If $P_i \neq \emptyset$, then each segment of E' has exactly one endpoint in C_i.*

Proof. Let $p \in P_i$ and let $e' \in E'$ be any left-over segment. Since p is not matched in Phase 2 with e', we know that $d(p, v') < r/4$ for an endpoint v' of e'. Then, $d(v', v_i) \leq d(v', p) + d(p, v_i) < r/2$, implying that $v' \in C_i$, so e' has at least one endpoint in C_i. Further, e' cannot have both endpoints in C_i, since e' has length greater than r, while C_i has radius $r/2$. □

We call (P_1, P_2) a *1-partition* if some P_i is empty, and a *2-partition* otherwise. For simplicity, unless otherwise stated, the following discussions assume that (P_1, P_2)

is a 2-partition; the case of a 1-partition is easily handled.

For $i = 1, 2$, let $R_i = \mathcal{D}_{v_i}(r)$ be the disk of radius r that is concentric with C_i. We 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 4.7. *Let $\triangle abc$ be a good triangle with respect to P_i whose edge lengths are at least $r/4$. Consider a segment $e = (p, q) \in E'$, with $p \in C_i$ and with q closer to b than to a and c. Then, for any $x \in P_i$, the triangles $\triangle pqa$ and $\triangle xbc$ have edge lengths at least $r/8$.*

Proof. Points a, b and c are each at distance at least $r/2$ from each of x and p, since a, b and c are each at distance at least r from the center, v_i, of R_i, while x and p are each within distance $r/2$ from v_i. Also, recall that e has length at least r. If $d(b, q) \geq r/8$, then $\triangle pqa$ and $\triangle xbc$ have edge lengths at least $r/8$. Otherwise, $d(q, a) \geq r/8$, since $d(q, a) + d(q, b) \geq d(a, b) \geq r/4$, and again $\triangle pqa$ and $\triangle xbc$ have edge lengths at least $r/8$. Refer to Fig. 4.2. $\square$

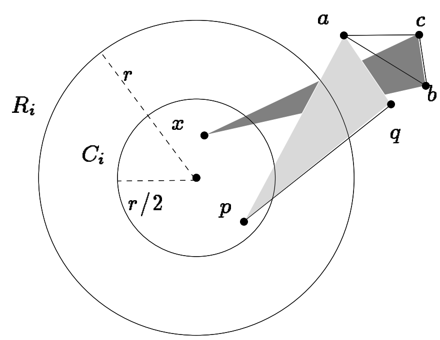

FIG. 4.2. *Swapping: $\triangle abc$ is a good triangle with respect to P_i, having edge lengths at least $r/4$. Then, triangles $\triangle pqa$ and $\triangle xbc$ have edge lengths at least $r/8$.*

Phase 3 proceeds by constructing C_i and R_i, $i = 1, 2$, then performing the swapping operation described in Lemma 4.7 for all points in P_1 (*swapping stage 1*), and then performing the same operation for all points in P_2 (*swapping stage 2*). By Lemma 4.8 (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 4.8. *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 $T_{1,bad}$ 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 $t_1 \geq T_{1,bad}$, since each bad triangle has at least one vertex inside R_1. Recall that R_1 is a disk of radius r; thus, by our choice of r, 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 $n/3 \geq \ell + \ell_1 + t_1 \geq \ell + \ell_1 + T_{1,bad}$. Let $T_{1,good}$ be the number of good triangles with respect to P_1. Then $T_{1,good} = T - T_{1,bad} = (n/3 - \ell) - T_{1,bad} \geq (n/3 - \ell) - (n/3 - \ell - \ell_1) = \ell_1$, i.e., there are enough good triangles for the points in P_1. For swapping stage 2, the situation is the same except that now $\ell = \ell_2$, meaning that we have a 1-partition, 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 4.7. 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 4.7 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 the disk defined by $\mathcal{C}$ contains k points of S; $2k + l$ points of S lie outside the disk (including the one point on its boundary, $\mathcal{C}$). 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 $O(n^{2.5})$ (either by the $O(n^{2.5})$-time algorithm of [11] or by the $O(\sqrt{n}m)$-time algorithm of [7], where $m = O(n^2)$ is the number of edges in the graphs). In Phase 3, we consider an arbitrary edge $e \in E'$ and the disks (C_i and R_i) of radii $r/2$ and r centered at its endpoints, identify the corresponding good/bad triangles, and perform local swappings, in $O(n)$ time per swap. Finally, as analyzed before, radius r and circle $\mathcal{C}$ are determined in $O(n^2)$ time.

THEOREM 4.9. *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 disk 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 4.10. *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 4.11. *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, in the resulting path, if $p \in \{a, b, c, x, y, z\}$ and $q \in \{a, b, c, x, y, z\}$ are 2-neighbors, then $d(p, q) \geq \delta/2$.*

Proof. We need to satisfy the following requirements: (1) the first two nodes of triangle $\triangle xyz$ visited must each be at distance at least $\delta/2$ from c, and (2) the first node of triangle $\triangle xyz$ visited must be at distance at least $\delta/2$ from b. This task can be carried out as follows: (1) among nodes x, y and z, the one that is closest to c (say x) is visited last, and (2) between the remaining two nodes, the one that is closer to b (say y) is visited second to last; this fixes the entire ordering of visiting the nodes. To verify, consider requirement (1) first. If $d(c, x) \geq \delta/2$, then the requirement is trivially satisfied. If $d(c, x) < \delta/2$, then we have $d(c, y) > \delta/2$, since $d(x, y) \geq \delta$

((x, y) is an edge of triangle $\triangle xyz$) and $d(c, y) + d(c, x) \geq d(x, y)$. Similarly, we have $d(c, z) > \delta/2$. This completes the verification of requirement (1). Requirement (2) is satisfied by exactly the same argument. $\square$

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 4.10 to obtain a Hamiltonian path, if n is a multiple of 3. Combining Theorem 4.10 and Theorem 4.9, 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 4.12. *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 $n = 3k + l$, where $l = 1$ or 2. Recall from Section 4.1 that at the completion of Algorithm *Big-Triangles*, we have k triangles and one or two left-over points outside or on the boundary ($\mathcal{C}$) of the disk of radius r. We will combine such left-over points with existing triangles, and then perform the concatenation.

For each left-over point $p \in S$ that is not a vertex of an existing triangle, we claim that we can find an existing triangle $\triangle abc$ (with edge length at least δ $(= r/8)$) such that p is at distance at least r $(> \delta)$ from each of a, b, and c. We call $\triangle abc$ a *good* triangle for p. To prove this claim, note, from the definition of r, that there are at most k points (including p) in the open disk $\mathcal{D}_p(r)$. Thus, there are at most $k - 1$ triangles having at least one vertex in $\mathcal{D}_p(r)$, meaning that there exists some triangle $\triangle abc$ whose vertices are each outside (or on the boundary of) $\mathcal{D}_p(r)$, as claimed. We create a special "triangle" $abcp$.

If $n = 3k + 2$, let p and q denote the 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 a common good triangle, $\triangle abc$. We claim then that $d(p, q) \geq r > \delta$; otherwise, the disk $\mathcal{D}_p(r)$ contains q, as well as at least one vertex of each of the $k - 1$ other large triangles, contradicting the choice of r. Thus we obtain one special "triangle" with five vertices, $abcpq$, with pairwise distances of at least δ.

We now treat the special "triangles" ($abcp$, $abcq$, or $abcpq$) as if they were triangles during the concatenation process. In order 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 the remaining points of the "triangle" (p, q, or both p and q). In order to concatenate a triangle after it, we consider only the last point (p or q) and its two previously visited points as a triangle, and apply the same lemma (Lemma 4.11).

Combining Algorithm *Big-Triangles* with Theorem 4.10, we obtain an algorithm to construct a Hamiltonian path whose max-min 2-neighbor distance is at least $\frac{1}{32}$ times the optimal max-min 2-neighbor distance of a Hamiltonian cycle. We have proved the following lemma:

LEMMA 4.13. *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.

In order to construct a Hamiltonian path whose objective function value is at least $\frac{1}{32}$ times that of an optimal *path* (as opposed to an optimal *cycle*, as in the above lemma), considerably more effort is required. The modifications to the algorithm and the analysis are given in the Appendix, where we prove:

THEOREM 4.14. *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.*

Since the Hamiltonian path on S obtained in Lemma 4.13 above has a max-min 2-neighbor distance provably close the the optimal max-min 2-neighbor distance in a *cycle*, we can use it to obtain an approximation algorithm for the cycle version of the problem, by appropriately "closing" the path into a tour:

THEOREM 4.15. *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.*

Proof. Given a list of triangles (with possibly one or two being special triangles) whose minimum edge length is at least $\delta = r/8$, we use the following process to chain the triangles into a cycle such that any two nodes that are 2-neighbors in the cycle are at distance at least $\delta/4$:

We first check if there is any pair of triangles $\triangle abc$ and $\triangle xyz$ such that two vertices of $\triangle abc$, say a and b, are each at distance at least $\delta/4$ from each of x, y and z. If so, chain the triangles starting with $\triangle abc$ (visiting in that order) and end with $\triangle xyz$ (in the order required by its predecessor), by applying Lemma 4.11. The ordering of the other triangles is arbitrary.

We assume from now on that no such pair of triangles exists. We consider an arbitrary triangle $\triangle abc$, and construct disks C_a, C_b and C_c respectively centered at a, b and c, each with radius $\delta/4$. Since the search in Step 1 is not successful, C_a, C_b, and C_c must together contain (in their *interiors*) at least two vertices of each of the remaining triangles (each disk contains at most one vertex of a triangle since each triangle edge has length at least δ). We say that a triangle is *covered* by a pair of disks $\{C_a, C_b\}$ if its two vertices are respectively contained in (the interiors of) C_a and C_b.

First, consider the case in which there are no special triangles.

Case (a): *One pair of disks, say $\{C_b, C_c\}$, covers all triangles.*

Then all triangles are of type xBC, where B and C denote the vertices respectively inside C_b and C_c (all such vertices form *clusters* B and C), and x denotes the third vertex (possibly $x \in A$, where cluster A is defined similarly). It is easy to see that any two points in different clusters are at distance larger than $\delta/2$. Also, any such point x is at distance greater than $\delta/2$ from any point in B (resp. C). (Suppose $\triangle xb'c'$ is one such triangle with $b' \in B$. For any point $b'' \in B$, $d(b', b'') < \delta/2$, but $d(x, b') \geq \delta$, and thus $d(x, b'') \geq d(x, b') - d(b', b'') > \delta/2$.) Construct the Hamiltonian cycle by repeatedly traversing the triangles in the order of xBC, and finally going back to the starting point (denoted by $xBC \rightarrow$ starting point).

Case (b): *Two pairs of disks, say $\{C_b, C_c\}$ and $\{C_a, C_c\}$, are necessary and sufficient to cover all triangles.*

Then there are three types of triangles: ABC, xBC, and yAC, where $x \notin A$ is at distance greater than $\delta/2$ from any point of B (resp. C), and $y \notin B$

is at distance greater than $\delta/2$ from any point of A (resp. C), by the same argument above. Construct the Hamiltonian cycle by exhausting all triangles of the same type and then going to the next type, in the order given as follows: $ABC \to xBC \to AyC \to$ starting point.

Case (c): *All three pairs of disks are needed to cover all triangles.*
Then there are four types of triangles: ABC, xBC, yAC, and zAB, where properties analogous to those in Case (b) hold for x, y, and z. Construct the Hamiltonian cycle as follows:
$ABC \to xBC \to AyC \to ABz \to$ starting point.

Now consider the case in which there are special triangles. If there is one with five vertices, say $uvwpq$, then at least two vertices, say p and q, must each be "far away" from all vertices of $\triangle abc$ (since C_a, C_b and C_c together can contain at most three vertices of $uvwpq$). Thus we can complete the chaining of triangles in Step 1 by starting with $pquvw$ (in that order) and ending with $\triangle abc$ (in the order required by its predecessor). Consider therefore special triangles F with four vertices. If F has less than three vertices in the interiors of C_a, C_b and C_c, then again F has at least two vertices each far away from all vertices of $\triangle abc$ and hence Step 1 can be completed. We thus only need to consider special triangles of type $ABCt$, where t (by the same argument) is at distance greater than $\delta/2$ from any point of A (resp. of B and of C). For the cases (a)–(c) considered above, the Hamiltonian cycle is given as follows (observe that the sequences are unchanged except for starting with the special triangles):

Case (a): $AtBC \to xBC \to$ starting point.

Case (b): $ABtC \to ABC \to xBC \to AyC \to$ starting point.

Case (c): $ABtC \to ABC \to xBC \to AyC \to ABz \to$ starting point.

The approximation factor follows by noting that $\delta/4 = \frac{r}{32}$ and that the optimal objective value is at most $2r$. □

5. Approximating the Max-Min m-Neighbor TSP. In this section, we show how the techniques developed in Sections 4.1–4.2 can be generalized to obtain a constant-factor approximation for the max-min m-neighbor TSP, for any constant $m > 2$. For simplicity, we only consider the *path* version in which *n is a multiple of $m+1$*. The method of "closing" a path into a cycle in Theorem 4.15 and the techniques of handling the case in which n is not a multiple of $m+1$ for the path version (in Appendix A) are not easily extended to the case in which $m > 2$; we pose them as open questions.

Analogous to the techniques for 2-neighbor TSP, we use *large $(m+1)$-cliques* to play the role of large triangles. An $(m+1)$-clique K_{m+1} consists of $m+1$ vertices that are the points in S (thus a triangle is a 3-clique); the *clique edges* of K_{m+1} are implicitly defined by its vertices. We want K_{m+1} to be *large*, meaning that each vertex of K_{m+1} is "far away" from all the other vertices of K_{m+1} (i.e., all clique edges of K_{m+1} are "long"). Again, our overall strategy consists of the following steps:

1. Create $n/(m+1)$ vertex-disjoint *large* $(m+1)$-cliques whose vertices are the points in S.
2. Concatenate these large $(m+1)$-cliques together to create a Hamiltonian path such that the distances between any m-neighbors are large.

We elaborate these steps in the following. First, let r be the radius of a smallest circle, $\mathcal{C}$, centered at one of the n points whose corresponding disk contains exactly $n/(m+1)$ points of S (including the center point), and one additional point on its boundary. From Lemma 3.2, $2r$ is an upper bound for the max-min distance in the path version of an m-neighbor TSP (since n is a multiple of $m+1$). Again, $\mathcal{C}$ and r are easily found in $O(n^2)$ time.

After computing r and $\mathcal{C}$, we use Algorithm *Big-*$(m+1)$*-Cliques* (analogous to Algorithm *Big-Triangles* in Section 4.1), to grow the $n/(m+1)$ points in the *interior* of $\mathcal{C}$ into $n/(m+1)$ vertex-disjoint large $(m+1)$-cliques. The process consists of m phases, where in phase i we grow each i-clique into an $(i+1)$-clique.

Algorithm *Big-*$(m+1)$*-Cliques*. The algorithm consists of m phases, classified into two parts.

Part 1. Complete matching phases. This part consists of $\lfloor m/2 \rfloor$ phases. For each phase $i = 1, 2, \ldots, \lfloor m/2 \rfloor$, perform the following.

Phase i:

Construct a bipartite graph $G = (V_1, V_2, E)$, where the nodes V_1 represent the i-cliques constructed in phase $i-1$ (if $i = 1$ then the nodes V_1 are the points in the disk defined by $\mathcal{C}$), and V_2 are the left-over points. Note that $|V_1| = n/(m+1)$, and $|V_2| = n - i \cdot \frac{n}{m+1} = (m+1-i)\frac{n}{m+1}$. Edge set E consists of "long" edges from V_1 to V_2, between two nodes $K_i \in V_1$ and $p \in V_2$ where p is at distance at least r from *all* vertices of K_i. Perform a maximum cardinality matching on G. By Lemma 5.1, all nodes of V_1 are matched. For each matched pair $K_i \in V_1$ and $p \in V_2$, grow the i-clique K_i into an $(i+1)$-clique by including point p as its additional vertex. Clearly, all clique edges of all resulting $(i+1)$-cliques have lengths at least r.

LEMMA 5.1. *For each phase $i = 1, 2, \ldots, \lfloor m/2 \rfloor$, there exists a matching in G in which all nodes of V_1 are matched.*

Proof. We use an argument similar to the proof of Lemma 4.4. Again, by way of contradiction, we assume that there exists a set $A \subseteq V_1$ for which $|A| > |\Gamma(A)|$, where $\Gamma(A) \subseteq V_2$ is a set of nodes that are adjacent (via edges of G) to at least one node of A. Now we have $|A| \leq |V_1| = n/(m+1)$, $|V_2| = (m+1-i)\frac{n}{m+1}$, and thus $|V_2 \setminus \Gamma(A)| > |V_2| - |A| \geq (m-i)\frac{n}{m+1}$. That is, for each i-clique K_i of A, there are at least $(m-i)\frac{n}{m+1} + 1$ left-over points (in $V_2 \setminus \Gamma(A)$) that are each within distance less than r from *some* vertex of K_i. Consider i (open) disks centered at each of the i vertices of K_i, each with radius r: by the pigeonhole principle, at least one of these disks contains more than $((m-i)\frac{n}{m+1})/i$ points of $V_2 \setminus \Gamma(A)$. Since $i \leq \lfloor m/2 \rfloor$, this disk contains at least $\frac{n}{m+1} + 2$ points, including the center point. This contradicts the original choice of r. $\square$

Part 2. Maximal matching and swapping phases. This part consists of the remaining $\lceil m/2 \rceil$ phases. For each phase $i = \lfloor m/2 \rfloor + 1, \ldots, m$, perform the following.

Phase i:

Let δ be the length of the shortest clique edge among all i-cliques at the end of phase $i-1$ (for $i = \lfloor m/2 \rfloor + 1$, take $\delta = r$). Construct a bipartite graph $G = (V_1, V_2, E)$ in the same way as in Part 1, except that the edge set E is specified as follows: there is an edge between two nodes $K_i \in V_1$ and $p \in V_2$ if and only if p is at distance at least $\delta/4$ from *all* vertices of K_i. Now perform the following steps:

1. Maximal matching step.
   Perform a *maximal* matching (rather than maximum cardinality matching) on G. For each matched pair $K_i \in V_1$ and $p \in V_2$, grow the i-clique K_i into an $(i+1)$-clique by including point p as its additional vertex. If all nodes in V_1 are matched, stop phase i here (all i-cliques in V_1 are grown into $(i+1)$-cliques); otherwise go to the next step.

2. Swapping step.
   Let $V_1' \subseteq V_1$ be the set of the left-over i-cliques of V_1 that are not matched in step 1, and $P \subseteq V_2$ the set of the left-over points that are not yet in any clique. Perform swapping operations of Lemma 5.3 on the i-cliques of V_1'. By Lemma 5.4, all of them can be grown into $(i+1)$-cliques, with the lengths of all clique edges at least $\delta/8$.

Suppose that the swapping step is performed in phase i. Let $K_i \in V_1'$ be some left-over i-clique whose vertices are $u_1, \ldots, u_i$, and let $C_j = \mathcal{D}_{u_j}(\delta/2)$ be the disk centered at u_j with radius $\delta/2$, for each $j = 1, \ldots, i$.

LEMMA 5.2. *Independent of the choice of the left-over i-clique K_i, the disks C_j induce a partition of the left-over points P into at most i non-empty clusters $P_1, \ldots, P_h$, $h \leq i$. For each $j = 1, \ldots, h$, all points of P_j are enclosed in the interior of C_j (re-numbering the subscripts of the vertices of K_i might be necessary), and each left-over i-clique in V_1' (including K_i) has exactly one vertex in the interior of C_j.*

Proof. This is a direct generalization of Lemma 4.5 and Lemma 4.6. □

In phase i, let $R_j = \mathcal{D}_{u_j}(r)$ be the disk that has the same center, u_j, as C_j and has radius r, for each $j = 1, \ldots, h$. We call a clique (either an i-clique or an $(i+1)$-clique) *good* for P_j if the clique has no vertex in the *interior* of R_j, and *bad* otherwise. By Lemma 5.2, each left-over i-clique in V_1' is *bad for all clusters* $P_1, \ldots, P_h$.

LEMMA 5.3. *Let K_{i+1} be a good $(i+1)$-clique for cluster P_j. Then for any left-over i-clique $K_i \in V_1'$ and a left-over point $x \in P_j$, we can construct two new $(i+1)$-cliques from K_{i+1}, K_i, and x such that one new $(i+1)$-clique is bad for all $P_1, \ldots, P_h$ and has clique edge lengths at least $\delta/8$, and the other has clique edge lengths at least $\delta/4$.*

Proof. Let $\delta' = \delta/4$. Note that all clique edges in K_{i+1} and in K_i have lengths at least δ'. We claim that there is always a vertex v^* of K_{i+1} that is at distance at least $\delta'/2 = \delta/8$ from all vertices of K_i. To see this, consider the vertices $u_1, \ldots, u_i$ of K_i, one by one. Initially, all $i+1$ vertices of K_{i+1} are candidates for v^*. Now, u_1 can be "too close" (within distance less than $\delta'/2$) to at most one vertex of K_{i+1}: if there is a vertex v_ℓ of K_{i+1} with $d(u_1, v_\ell) < \delta'/2$, then since all clique edges in K_{i+1} have lengths at least δ', we have $d(v, v_\ell) \geq \delta'$ for each vertex $v \neq v_\ell$ of K_{i+1}, and thus $d(u_1, v) \geq d(v, v_\ell) - d(u_1, v_\ell) > \delta'/2$. Therefore u_1 can filter out at most one candidate for v^*. Now consider u_2 and the remaining i candidates. Again, u_2 can filter out at most one candidate. Repeating this process, there are i constraints (from the vertices of K_i) in total, but we have $i+1$ candidates initially. Therefore such v^* must always exist. This completes the proof of the claim.

Now we construct two new $(i+1)$-cliques from K_{i+1}, K_i and x as follows. First, we grow K_i into an $(i+1)$-clique by including v^* as its additional vertex. Clearly, this new clique has edge lengths at least $\delta'/2 = \delta/8$. Also, since K_i is originally bad for all clusters $P_1, \ldots, P_h$ by Lemma 5.2, this new $(i+1)$-clique continues to be bad for all clusters. Secondly, we construct the other $(i+1)$-clique by swapping out vertex v^* from K_{i+1} and swapping in point $x \in P_j$ as a new vertex. The fact that this clique still has edge lengths at least $\delta' = \delta/4$ follows from the observation that K_{i+1} is good for P_j (hence all vertices of K_{i+1} are at distance at least r from the center of R_j, C_j), that x is within distance less than $\delta/2$ from the center of C_j (by Lemma 5.2), and the triangle inequality. □

LEMMA 5.4. *Suppose that the swapping step is performed in phase i. Then there are always enough good $(i+1)$-cliques for P_j, $j = 1, \ldots, h$, such that all left-over i-cliques can be grown into $(i+1)$-cliques by the swapping operations of Lemma 5.3,* *with final clique edge lengths at least $\delta/8$.*

Proof. This is similar to Lemma 4.8. The swapping step goes through at most h stages, where for each stage $j = 1, \ldots, h$ we perform the swapping operations of Lemma 5.3 for points in P_j, until either all points in P_j are used up (in which case we move to the next stage), or all left-over i-cliques are grown into $(i+1)$-cliques (in which case we stop the entire swapping step). Now consider any such stage j. Right before stage j, suppose there are T $(i+1)$-cliques (and hence $\frac{n}{m+1} - T$ left-over i-cliques), and ℓ left-over points that are not yet in any clique. Let $|P_j| = \ell_j$, then $\Sigma_j \ell_j = \ell$. Also, ℓ is at least $\frac{n}{m+1} - T$, the number of left-over i-cliques. We claim that all points of P_j have enough good $(i+1)$-cliques to swap with. To show this, suppose among the T $(i+1)$-cliques, $T_{j,good}$ of them are good for P_j and $T_{j,bad}$ of them are bad for P_j; $T = T_{j,good} + T_{j,bad}$. Consider all points of S in the interior of R_j: these include ℓ_j points from P_j, $\frac{n}{m+1} - T$ vertices each from a left-over i-clique (by Lemma 5.2), and t_j vertices from bad $(i+1)$-cliques, where $t_j \geq T_{j,bad}$. Since the interior of R_j contains at most $n/(m+1)$ points, we have $n/(m+1) \geq \ell_j + (\frac{n}{m+1} - T) + t_j \geq \ell_j + (\frac{n}{m+1} - T) + T_{j,bad}$. Then we have $T_{j,good} = T - T_{j,bad} \geq \ell_j$, completing the proof of the claim. Since the number of the left-over i-cliques is no more than the number of the left-over points (all of which in turn have enough good $(i+1)$-cliques to swap with), all the left-over i-cliques can eventually be grown into $(i+1)$-cliques. Also, from Lemma 5.3, after each swapping operation, the only new $(i+1)$-clique that can possibly have edge lengths less than $\delta/4$ (but at least $\delta/8$) is *bad for all* $P_1, \ldots, P_h$, and hence it will not be used for swapping hereafter in phase i. Therefore at the end of phase i all left-over i-cliques can be grown into $(i+1)$-cliques with clique edge lengths at least□

From Lemma 5.4, we know that in the worst case, the length of the shortest clique edge is reduced by a factor of 8 (from δ to $\delta/8$) in each phase i of Part 2. Therefore the final $(m+1)$-cliques have edge lengths at least $r(1/8)^{\lceil m/2 \rceil}$.

THEOREM 5.5. *Given a constant $m > 2$ and a complete graph on set S of n points satisfying triangle inequality, where n is a multiple of $m+1$, there exists an algorithm that constructs in $O(n^{2.5})$ time $n/(m+1)$ vertex-disjoint $(m+1)$-cliques from points in S such that the smallest edge length of the cliques is at least $r(1/8)^{\lceil m/2 \rceil}$, where r is the radius of a smallest (open) disk centered at a point of S, containing exactly $n/(m+1)$ points of S, and one additional point on its boundary.*

Now we are ready to concatenate the $(m+1)$-cliques into a Hamiltonian path.

THEOREM 5.6. *Given a list of $(m+1)$-cliques whose minimum edge length is δ, the cliques can be chained into a path such that any two nodes that are m-neighbors in the path are at distance at least $\delta/2$ from each other.*

Proof. This is the same as Theorem 4.10, except that we use an immediate generalization of Lemma 4.11 to chain the $(m+1)$-cliques. □

From Theorems 5.5 and 5.6, and the fact that $2r$ is an upper bound for the optimal objective value, we conclude with the following theorem.

THEOREM 5.7. *Given a constant $m > 2$ and a complete graph on set S of n points satisfying triangle inequality, where n is a multiple of $m+1$, there exists an algorithm that constructs in $O(n^{2.5})$ time a Hamiltonian path on S whose objective value for the max-min m-neighbor problem is at least $\frac{1}{4}(\frac{1}{8})^{\lceil m/2 \rceil}$ times the optimal.*

6. Exact Algorithms for Points on a Line or on a Circle. 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 (for the path version of points on a circle, we only consider the case in which

n is an odd integer or the points are equally spaced). First, we concentrate on the cycle version of points on a line; we discuss the remaining problems and versions in Appendix B.

In the following, the n points of S lie on a line, labeled $1, 2, \ldots, n$ from left to right. We distinguish between two cases, depending on whether n is odd or even. Interestingly, the latter case is 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 *constructed* and output (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 constructed using a purely combinatorial method in $O(n)$ time.

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

6.1. 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 3.3, 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 $p = 1, 2, \ldots, k$, and the "left" edges connect node $(k+q)$ to node q, for $q = 1, 2, \ldots, k+1$.

It is easy to verify that the above specification gives a Hamiltonian cycle, and that each "right" edge spans $k+2$ nodes and each "left" edge spans $k+1$ nodes. Therefore this Hamiltonian cycle is optimal.

THEOREM 6.1. *For the cycle version of the maximum scatter TSP where the n points lie on a line, the distances are Euclidean, and n is an odd integer, there exists an algorithm that specifies an optimal Hamiltonian cycle in $O(1)$ time, and constructs such a cycle in $O(n)$ time.*

6.2. 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 6.2. *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. Let $S_{in} = I_1 \cap I_2$ and $|S_{in}| = \ell > 0$. Also, let $S_{out} = S \setminus (I_1 \cup I_2)$. Then since $|I_1 \cup I_2| = 2k - \ell$, we have $|S_{out}| = \ell$. Suppose on the contrary that there exists a Hamiltonian cycle C that avoids having an edge joining two nodes either both in I_1 or both in I_2. Then consider the ℓ nodes in S_{in}: they can only be connected to the ℓ nodes in S_{out}. Now there are 2ℓ edges of C going out from S_{in}, but the total degree of the nodes of S_{out} in C is 2ℓ, therefore all of the nodes of S_{out} can only be connected to the nodes of S_{in} also. This implies that the nodes in $S_{in} \cup S_{out}$ are disconnected from the other nodes of S, and thus C is not a Hamiltonian cycle, a contradiction. ∎

COROLLARY 6.3. *Any shortest edge in any optimal Hamiltonian cycle on S can span at most k nodes.*

Proof. By Lemma 3.1, any such edge e can span at most $k+1$ nodes. Now suppose e spans $k+1$ nodes. Let I_1 be the k-interval consisting of the leftmost k nodes spanned by e, and I_2 be the k-interval of the rightmost k nodes spanned by e. Clearly I_1 and I_2 overlap, and the optimal cycle containing e does not have any edge joining either two nodes of I_1 or two nodes of I_2, contradicting Lemma 6.2. Thus e can span at most k nodes. □

COROLLARY 6.4. *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 $d^* \leq d_{e_2}$ if I_2 overlaps with I_1, and $d^* \leq d_{e_3}$ otherwise.*

Proof. If I_1 and I_2 do not overlap, then they must be the leftmost and rightmost k-intervals on S and $I_1 \cup I_2 = S$. Thus I_3 must overlap with I_1 or I_2. The stated upper bounds follow from Lemma 6.2 and Corollary 6.3. □

Assume now that we are given the shortest k-interval, I_1, on S. We give an algorithm that specifies an optimal Hamiltonian cycle C^* by distinguishing the following two cases:

Case 1: *I_1 is the leftmost k-interval.* Place $I_1 = \{1, 2, \ldots, k\}$ and $I_2 = \{k+1, \ldots, 2k\}$ on a *schematic cycle* as shown in Fig. 6.1(a)(b). Produce edges $e_1 = (1, k+1)$ and $e_2 = (k, 2k)$.

Start visiting node 1, and then repeatedly visit the k-th node in the schematic cycle counted *counterclockwise* from the current node (the current node is counted as the first node), i.e., visiting nodes $k+2, 3, k+4, \ldots$, until reaching an endpoint node of e_2. Note that this visits every other node in I_1 and in I_2 in *increasing* order (nodes $1, 3, \ldots$ in I_1 and nodes $k+2, k+4, \ldots$ in I_2). If k is an even integer (see Fig. 6.1(a)), then node $2k \in I_2$ is finally reached; if k is odd (see Fig. 6.1(b)), then node $k \in I_1$ is finally reached. In either case, follow e_2 to visit the other endpoint node of e_2. Now, complete the other half of C^* by a symmetric method: repeatedly visit the k-th node in the schematic cycle, this time counted *clockwise* from the current node, visiting the remaining nodes in I_1 and in I_2 in *decreasing* order, until reaching node $k+1$. Finally, follow e_1 to go back to node 1 (see Fig. 6.1(a)(b)).

It is easy to verify that C^* is a Hamiltonian cycle. Also, all edges in C^* other than e_1 and e_2 span k nodes on one side and $k+2$ nodes on the other side of the schematic cycle, and e_1 and e_2 span $k+1$ nodes on both sides. In addition, all edges in C^* avoid joining two nodes that are both in I_1 or both in I_2. (I_1 is the shortest k-interval, and I_2 may happen to be the second shortest k-interval.) Therefore, C^* achieves the upper bound given in Corollary 6.4 and is an optimal Hamiltonian cycle.

Case 2: *I_1 is not the leftmost k-interval.* First, re-label the nodes of S, by labeling the nodes of I_1 from 1 to k, and continuing the labeling in a wrap-around fashion (see Fig. 6.1(c)). Then, use the new labels and apply the same procedure of Case 1 to produce C^*. Since the wrap-around effect has already been taken into account in Case 1 (via the schematic cycle), it is clear that C^* is an optimal Hamiltonian cycle.

Clearly, by considering each of the $k+1$ possible k-intervals we can, in $O(n)$ time, identify the length d^*. Then, we can apply the above algorithm to construct an optimal Hamiltonian cycle in additional $O(n)$ time.

THEOREM 6.5. *For the cycle version of the maximum scatter TSP where the n points lie on a line, the distances are Euclidean, and n is an even integer, there exists an algorithm that computes optimal Hamiltonian cycle in $O(n)$ time.*

In Appendix B we consider extensions of these results to the cycle version of points on a circle (Theorem B.2), and the path version of points on a line (Theorem B.4) or

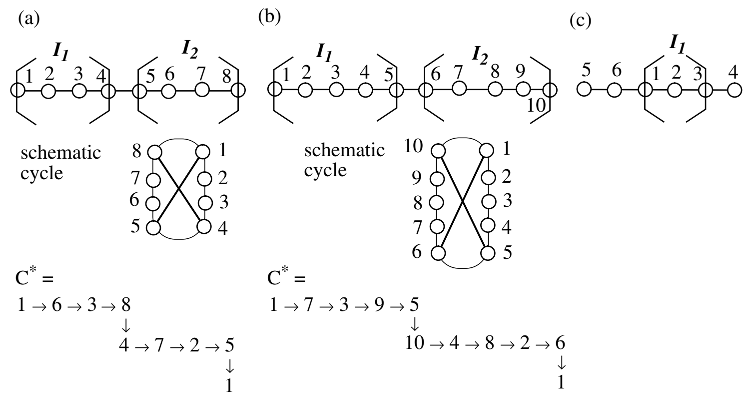

FIG. 6.1. *Examples of C^* specified by the algorithm: (a) $n = 8$ and $k = 4$ is even; (b) $n = 10$ and $k = 5$ is odd; (c) wrap-around re-labeling when I_1 is not the leftmost k-interval.*

on a circle (Corollary B.5).

7. Conclusion. We conclude with the following open problems:

- What is the complexity of the geometric maximum scatter TSP for points in the Euclidean plane? We conjecture that it is NP-hard. The recent result of Fekete [8] shows that the problem is indeed NP-hard in Euclidean spaces of dimension three or greater.
- Can better bounds for the max-min m-neighbor TSP for any $m \geq 1$ be obtained in the geometric case?
- For points on a line or on a circle, can we solve the max-min m-neighbor TSP exactly for $m > 1$? We conjecture that there exists a polynomial-time algorithm, perhaps similar to the one we presented for the case $m = 1$.
- Can we solve the maximum scatter TSP exactly for points lying on the boundary of an arbitrary convex body?

Acknowledgments. We thank Samir Khuller for stimulating discussions and comments, which lead to Theorem 4.1 and an improvement on the running time of Theorem 3.5. 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, R. A. Duke, H. Lefmann, V. Rödl, and R. Yuster. The algorithmic aspects of the regularity lemma. In *Proc. 33rd Annu. IEEE Sympos. Found. Comput. Sci. (FOCS 92)*, pages 473–481, 1992.

[2] 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.

[3] E. M. Arkin, Y.-J. Chiang, J. S. B. Mitchell, S. S. Skiena, and T. Yang. On the maximum scatter TSP. In *Proc. 8th ACM-SIAM Sympos. Discrete Algorithms*, pages 211–220, 1997.

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

[5] A. Barvinok, D. S. Johnson, G. J. Woeginger, and R. Woodroofe. The maximun traveling salesman problem under polyhedral norms. In *Sixth Conference on Integer Programming and Combinatorial Optimization*, page to appear, Rice University, Houston, TX, June 1998.
[6] G. A. Dirac. Some theorems on abstract graphs. *Proc. London Math. Soc.*, 2:69–81, 1952.
[7] S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM J. Comput.*, 4:507–518, 1975.
[8] S. P. Fekete. Simplicity and hardness of the maximum traveling salesman problem under geometric distances. Manuscript (submitted), Mathematisches Institut, Universität zu Köln, 1998.
[9] P. Hall. On representatives of subsets. *J. London Math. Soc.*, 10:26–30, 1935.
[10] R. Hassin and S. Rubinstein. An approximation algorithm for the maximum traveling salesman problem. Manuscript, submitted. Tel Aviv University, 1997.
[11] J.E. Hopcroft and R.M. Karp. A $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2:225–231, 1973.
[12] S. Khuller. Personal communication, 1996.
[13] J. Komlós. Personal communication, 1996.
[14] J. Komlós, G. N. Sárközy, and E. Szemerédi. On the square of a Hamiltonian cycle in dense graphs. *Random Structures and Algorithms*, 9(1 and 2):193–211, 1996.
[15] J. Komlós, G. N. Sárközy, and E. Szemerédi. On the Pósa-Seymour conjecture. *J. Graph Theory*, to appear.
[16] 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.
[17] 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.
[18] R.G. Parker and R.L. Rardin. Guaranteed performance heuristics for the bottleneck traveling salesman problem. *Oper. Res. Letters*, 2(6):269–272, 1984.
[19] K. Penavic. Optimal firing sequences for CAT scans. Manuscript, Department of Applied Mathematics, SUNY Stony Brook, NY, 1994.
[20] F. Scholz. Coordination hole tolerance stacking. Technical Report BCSTECH-93-048, Boeing Computer Services, November, 1993.
[21] F. Scholz. Coordination hole tolerance stacking panel/panel/stringer join. Technical Report BCSTECH-94-017, Boeing Computer Services, May, 1994.
[22] F. Scholz. Cylinder section tolerance stacking issues. Technical Report BCSTECH-94-018, Boeing Computer Services, May, 1994.
[23] F. Scholz. Tolerance stack analysis methods a critical review. Technical Report BCSTECH-95-021, Boeing Computer Services, November, 1995.
[24] F. Scholz. Tolerance stack analysis methods. Technical Report BCSTECH-95-030, Boeing Computer Services, December, 1995.

Appendix A. Max-Min 2-Neighbor Hamiltonian Path: n 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 4.12). Now we give algorithms achieving the same factor for the remaining cases, when $n = 3k + l$, for $l = 1, 2$. 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 A.1. *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 4.3.

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 4.10 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 A.2. *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 4.4, 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 4.8, 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 4.5 and Lemma 4.6 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 A.3. *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. Let $e = (v_1, v_2)$ be a left-over segment in E', where v_i is within distances less than $r/4$ from all points in P_i for $i = 1, 2$ (see Lemma 4.5). Recall that e is constructed in Phase 1 and that one of its endpoints, say v_1, is inside $\mathcal{C}$ and the other endpoint v_2 is outside or on the boundary of $\mathcal{C}$. Let R_i and C_i, $i = 1, 2$ as defined in the description of Phase 3 be centered at v_i. Recall that the radiuses of R_i and C_i are r and $r/2$, respectively; also, a triangle is good for P_i if all three vertices are outside or on the boundary of R_i, and bad otherwise. Now $d(v_1, a) < r$, since v_1 is inside $\mathcal{C}$ and a is the center of $\mathcal{C}$, and thus a is inside R_1. This means that $\triangle aQQ'$ is *bad* for P_1 and is not needed for swapping the points of P_1. We first perform swappings on P_1 (swapping stage 1), with $\triangle aQQ'$ untouched. Now we perform swappings on P_2 (swapping stage 2). Observe that v_2 is outside or on the boundary of $\mathcal{C}$, so $d(a, v_2) \geq r$ and thus $\triangle aQQ'$ is *good* for P_2. Suppose in swapping stage 2 we want to swap $\triangle aQQ'$ with a point $x \in P_2$ and a left-over segment $e' = (p, q)$, where p is closer to P_1 and q is closer to P_2 (and again by Lemma 4.5 p is within distances less than $r/4$ from all points of P_1, and similarly for q and points of P_2). We claim that we can construct triangles $\triangle xQQ'$ and $\triangle apq$ so that their edge lengths are large enough. (Note that Q and Q' stay in the same triangle.) Clearly, $\triangle xQQ'$ satisfies the condition. To prove the claim, observe that q is inside C_2 and thus $d(q, v_2) < r/2$. But since $d(a, v_2) \geq r$, we have $d(a, q) \geq d(a, v_2) - d(q, v_2) > r/2$. Also, by the construction in Phase 1, any left-over point $t \in P_1$ (before swapping stage 1) is *outside or on the boundary of* $\mathcal{C}$, i.e., $d(a, t) \geq r$. But as already mentioned, $d(p, t) < r/4$ by Lemma 4.5. Therefore $d(a, p) \geq d(a, t) - d(p, t) > (3r)/4$. This shows that all edge lengths in $\triangle apq$ are also large enough, which completes the proof of the claim. Now the cardinality of P_2 and the number of good triangles for P_2 are both decreased by 1, so there are enough good triangles remaining to swap with the rest of the points in P_2 (as shown in Lemma 4.8), with $\triangle xQQ'$ untouched — actually it is a bad triangle for P_2 now. This shows that the conditions of the lemma are satisfied. ◻

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 A.4. *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 of a left-over segment such that v_1 is within distance less than $r/4$ from all points of P_1 (such endpoint must exist for every segment; see Lemma 4.5 and Lemma 4.6). 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 4.7) still works with this new definition of good/bad triangles, since any point not in the interior of R_1' is at distance greater than $(5r)/8 - r/2 = r/8$ from each point inside C_1. Also, the two new triangles produced by a swapping operation are both *bad* for P_1, and thus there are no further shrinkings on their edge lengths. In the following, good/bad triangles are all defined by using R'_1. 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 A.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.*

Proof. The proof is a refinement of the proof of Lemma 4.8. At the end of Phase 2, suppose there are T triangles, $T_{1,bad}$ bad triangles for P_1, t_1 bad triangle vertices inside R'_1, and ℓ left-over segments. Then we have $|P_1| = \ell$, $T + \ell = n'/3$, and $T_{1,bad} \leq t_1$. If R'_1 has less than $n'/3$ points in the interior, then $T_{1,bad} + \ell + \ell \leq t_1 + 2\ell < n'/3$; if some bad triangle has more than one vertex inside R'_1, then $T_{1,bad} < t_1$, i.e., $T_{1,bad} + \ell + \ell < t_1 + 2\ell \leq n'/3$. In both cases, we have $T_{1,bad} \leq n'/3 - 2\ell - 1$. The number $T_{1,good}$ of good triangles for P_1 is then given by $T_{1,good} = T - T_{1,bad} \geq (n'/3 - \ell) - (n'/3 - 2\ell - 1) = \ell + 1 = |P_1| + 1$. □

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 A.2. *We can perform the desired swappings on P_1.*

Proof. We perform the task by considering the cases below.

Case (a) s is an endpoint of some left-over segment $e = (s,q)$.
Since q is outside or on the boundary of $\mathcal{C}$, we have $d(a,q) \geq r$. Also, we already have $d(a,s) \geq r/8$. We swap $\triangle aQQ'$ with segment e and point $x \in P_1$ and produce two triangles $\triangle asq$ and $\triangle xQQ'$. Notice that $\triangle xQQ'$, with Q and Q' staying in the same triangle, is bad for P_1 and will not be touched from now on. Also, both $|P_1|$ and $T_{1,good}$, the number of good triangles for P_1, are decreased by 1 and the swapping operations can be completed as before.

Case (b) no such s exists as an endpoint of a left-over segment (i.e., each left-over segment has one endpoint within distance less than $r/8$ from a) and s is a vertex of some triangle $\triangle suv$.
Note that by the construction in Phases 1 and 2, $\triangle suv$ has exactly one vertex inside $\mathcal{C}$; since s is inside $\mathcal{C}$, we have $d(a,u) \geq r$ and $d(a,v) \geq r$. We want to swap $\triangle aQQ'$ with a point $x \in P_1$ and a left-over segment (p,q), where p is inside $\mathcal{C}$ (and thus $d(p,a) < r/8$) and q is inside C_1.

1. $\triangle suv$ is good for P_1.
   We produce triangles $\triangle xQQ', \triangle asv$, and $\triangle upq$. To verify, observe that $d(a,s) \geq r/8$ and $d(a,v) \geq r$, so $\triangle asv$ is fine. As for $\triangle upq$, we have $d(u,p) > (7r)/8$, since $d(u,a) \geq r$ and $d(p,a) < r/8$. Also, u is outside or on the boundary of R'_1 (since $\triangle suv$ is good), so $d(u,d) \geq (5r)/8$. Recall that d is the center of R'_1 and that C_1 is also centered at d with radius $r/2$. Now q is inside C_1, so $d(q,d) < r/2$. Therefore $d(u,q) \geq d(u,d) - d(q,d) > r/8$,

which completes the validity of $\triangle upq$. Note that $\triangle asv$ is good, $\triangle upq$ is bad, and $\triangle xQQ'$ is bad and will not be touched from now on. Again, both $|P_1|$ and $T_{1,good}$ are decreased by 1 and the desired conditions are satisfied.

2. $\triangle suv$ is bad for P_1.
Recall that each bad triangle has exactly one vertex inside R_1'. If s is inside R_1', then both u and v are outside or on the boundary of R_1'; otherwise, one of u and v, say v, is inside R_1' (and s and u are outside or on the boundary of R_1'). In both cases, we again produce triangles $\triangle xQQ'$, $\triangle asv$, and $\triangle upq$. The validities of these triangles are established by the same arguments as before. Now $\triangle asv$ is *bad* (the other two triangles are bad as well), but originally $\triangle suv$ was bad also. Again $|P_1|$ and $T_{1,good}$ are both decreased by 1 and the desired conditions are satisfied.

$\square$

(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 A.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 $(7r)/8$.*

Proof. Recall that $d(a,d) \geq r$ ($\triangle aQQ'$ has no vertex inside R_1 in the first place). Now any point inside R_1' is within distance less than $(5r)/8$ from d, so any point inside R_1' is at distance at greater than $r - (5r)/8 = (3r)/8$ from a. But any point inside $\mathcal{C}$ is within distance less than $r/8$ from a. Therefore all $n'/3$ points inside R_1' are outside or on the boundary of $\mathcal{C}$ (and thus at distance at least r from a). This shows the disjointness of the two clusters. Let y and z be two points inside R_1' and inside $\mathcal{C}$, respectively. Then $d(y,z) > (7r)/8$, since $d(y,a) > r$ and $d(z,a) < r/8$. $\square$

CLAIM A.4. *We can perform the desired swappings on P_1.*

Proof. Recall that d is an endpoint of some left-over segment $e = (p,d)$, where p is inside $\mathcal{C}$ (and thus $d(p,a) < r/8$). We perform the task by considering the cases below.

Case (a) t is a point in P_1.
We produce $\triangle tpd$, which is valid since $d(t,d) \geq r/8$ (the condition of Case (II.2)) and $d(t,p) > (7r)/8$ (t and p are in different clusters). Now $|P_1|$ is decreased by 1 ($|P_1| = \ell - 1$) but $T_{1,good}$ is still ℓ. Thus we can complete the swappings on P_1 *without using* $\triangle aQQ'$.

Case (b) t is an endpoint of a left-over segment $e' = (p',t)$, where p' is inside $\mathcal{C}$.
Let x be any point of P_1. We produce $\triangle tpd$ and segment (p',x). Note that $\triangle tpd$ is valid as shown in Case (a), and segment (p',x) has length larger than $(7r)/8$ since p' and x are in different clusters. Again $|P_1|$ is decreased to $\ell - 1$ but $T_{1,good}$ is still ℓ, so we can complete the desired swappings on P_1.

Case (c) t is a vertex of a (bad) triangle $\triangle tuv$ for P_1.
Since each bad triangle has exactly one vertex inside R_1', u and v are outside or on the boundary of R_1', so $d(u,d) \geq (5r)/8$ and $d(v,d) \geq (5r)/8$. Let x be any point in P_1. Since x and (p,d) are left-over point and segment, by the construction of Phase 2, $d(x,d) < r/4$. Therefore $d(u,x) > (5r)/8 - r/4 = (3r)/8$; similarly, $d(v,x) > (3r)/8$. Thus we can produce triangles $\triangle uvx$ and $\triangle tpd$ (whose validity is established as before). Again $|P_1|$ is decreased to $\ell - 1$ but $T_{1,good}$ is still ℓ, so we can complete 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 A.5. *We can construct an optimal max-min 2-neighbor Hamiltonian path on S.*

Proof. As in Case (II.2), the interiors of $\mathcal{C}$ and of R_1' contain two *disjoint* clusters S_1 and S_2, respectively, each with $n'/3$ points (i.e., $|S_1| = |S_2| = n'/3 = k+1$), and the distance between any two points in different clusters is larger than $(7r)/8$. Let $S' = (S \setminus S_1) \setminus S_2$, then $|S'| = k-1$. We disregard the fake points Q and Q' from now on. Any point $p' \in S'$ is outside or on the boundary of $\mathcal{C}$, and thus p' is at distance greater than $r - r/8 = (7r)/8$ from any point of S_1. Similarly, p' is outside or on the boundary of R_1', and thus p' is at distance greater than $(5r)/8 - r/8 = r/2$ from any point of S_2. Note that $diam(S_1) < (r/8) \cdot 2 = r/4$, and similarly $daim(S_2) < r/4$. Without loss of generality, assume that $diam(S_1) < diam(S_2)$. To avoid having any pair of 2-neighbors both in S_1, the Hamiltonian path starts from a point in S_1, and visits two points outside S_1 (one in S_2 and one in S'), then repeats the process. After repeating the process $(k-1)$ times, S_1 and S_2 both have two points left over, and S' is empty. The only possible way to avoid having any pair of 2-neighbors both in S_1 is to visit the k-th point of S_1, then the last two points of S_2, and finally the last point of S_1. This means that there must be a pair of points both in S_2 that are 2-neighbors, i.e., $diam(S_2)$ is an upper bound for d^*, the max-min 2-neighbor distance of an optimal path. We can achieve this upper bound, by constructing the path $(S_1S_2S')^{k-1}S_1S_2S_2S_1$, where each occurrence of S_1 means visiting a point in S_1, and similarly for the occurrences of S_2 and S', and the last two points in S_2 to be visited are the pair in S_2 defining $diam(S_2)$. Clearly, all other 2-neighbors in this path have distances larger than $r/2$, which is larger than $diam(S_2)$, and the minimum 2-neighbor distance is $diam(S_2)$, the upper bound of d^*. Therefore $d^* = diam(S_2)$ and the path is optimal. □

This completes the proof of Theorem A.4. □

This completes the proof of Theorem 4.14.

Appendix B. Other Exact Algorithms for Points on a Line or a Circle.

B.1. Cycle Version: Points on a Circle. We define an *ℓ-interval* as before, and the *arc* of an ℓ-interval I to be the arc containing the ℓ nodes of I, with endpoints being the endpoint nodes of I. Also, we define the *length* of I to be the length of the longest edge among all edges joining two nodes of I. Observe that in the case of points on a line, the edge joining the two endpoints of I always gives the length of I; this is no longer true for the case of points on a circle — unless the angle subtended by the arc of I is no more than π.

LEMMA B.1. *Let S be a set of n nodes lying on a circle, and d^* be the length of the shortest edge of an optimal Hamiltonian cycle on S. If $n = 2k+1$ is odd, then for any $(k+1)$-interval $I = \{u, \ldots, v\}$ whose arc subtends an angle larger than π, the length of edge (u, v) is larger than d^*. Moreover, the shortest and the second shortest $(k+1)$-intervals are both shorter than edge (u, v) and must each subtend an angle no more than π. Similarly, if $n = 2k$ is even, then for any k-interval $I = \{u, \ldots, v\}$ whose arc subtends an angle larger than π, the length of edge (u, v) is larger than d^*. Moreover, the shortest, the second shortest, and the third shortest k-intervals are all shorter than edge (u, v) and must each subtend an angle no more than π.*

Proof. Consider the case in which n is odd (see Fig. B.1(a)). Observe that $I' = \{u, s, \ldots, t\}$ is also a $(k+1)$-interval, whose arc subtends an angle no more than π. This means that edge (u, t) is the longest edge with both endpoints in I'. By Corollary 3.3, the length of (u, t) is an upper bound for d^*. But since (u, v) is longer than (u, t), the length of edge (u, v) is larger than d^*. In fact, (u, v) is longer than *both* (u, t) and (s, v) by the same argument, where (s, v) defines the length of another $(k+1)$-interval $I'' = \{s, \ldots, t, v\}$ with subtended angle no more than π.

Now consider the case in which n is even (see Fig. B.1(b)). Again, $I' = \{s, p, \ldots, t\}$, $I'' = \{p, \ldots, t, v\}$ and $I''' = \{u, s, \ldots, q\}$ are three k-intervals each of whose arcs subtends an angle no more than π, and edge (u, v) is longer than all three edges joining both endpoints of I', of I'', and of I'''. This means that the length of (u, v) is larger than the length of the third shortest k-interval. The lemma then follows from Corollary 6.4. □

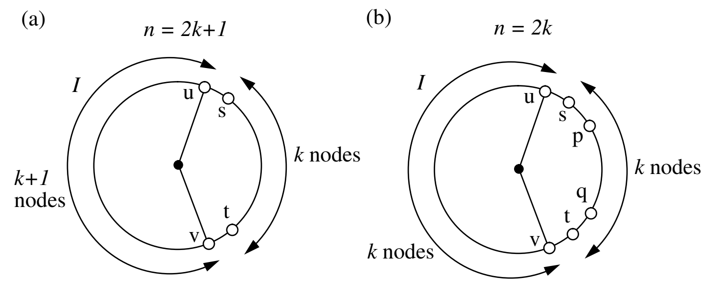

FIG. B.1. *Proof of Lemma B.1: (a) n is odd; (b) n is even.*

We solve the problem by the following algorithm. When $n = 2k+1$, we apply the algorithm of Section 6.1 to construct a Hamiltonian cycle C^*. When $n = 2k$, we first scan all n possible k-intervals to identify the shortest k-interval, I_1. By Lemma B.1, I_1 is found by taking the shortest edge among all edges joining the two endpoint nodes of all k-intervals. Then we label all nodes of S from 1 to $2k$ clockwise along the circle, with the nodes of I_1 labeled 1 to k. Finally, we use the new labels and apply the algorithm of to construct a Hamiltonian cycle C^*.

We claim that the resulting C^* is optimal. First, the wrap-around effect is readily taken care of: when $n = 2k+1$, each edge on C^* spans $k+2$ nodes on one side and also spans $k+1$ nodes on the other side; when $n = 2k$, each edge either spans $k+1$ nodes on both sides (edges e_1 and e_2 given in the algorithm of Section 6.2, or spans k nodes on one side and $k+2$ nodes on the other side (the remaining edges). Secondly, let I be an $(k+1)$-interval (resp. k-interval) when $n = 2k+1$ (resp. $n = 2k$). The issue that an edge spanning all $k+1$ nodes (resp. all k nodes) of I may not be the longest edge within I is resolved by Lemma B.1. Hence C^* is an optimal Hamiltonian cycle.

THEOREM B.2. *Let S be a set of n points lying on a circle and the distances are Euclidean. Then there exists an algorithm that solves the cycle version of the maximum scatter TSP on S exactly in optimal $O(n)$ time.*

B.2. Path Version: Points on a Line. Now, consider the path version of points on a line. As we did at the end of Section 3, we add a *fake point* Q very far away from all points of S, say at position ∞ so that it does not affect the max-min edge length, and 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. We call this method *fake-point algorithm* for future reference. We denote by C^* the intermediate cycle constructed, and P^* the final path obtained.

When $n = 2k$, adding Q makes the construction of C^* an easy case ($n' = n + 1$ is odd), for which we apply the algorithm of Section 6.1. The upper bound for the path version (from Lemma 3.1: every set R of cardinality $\lceil \frac{n}{2} \rceil + 1 = k + 1$ must have an edge inside R) is easily achieved (the edges in the resulting path P^* are $(k+1)$- or $(k+2)$-edges), and hence P^* is optimal. When $n = 2k + 1$, adding Q makes the construction of C^* a more complicated case ($n' = 2k + 2$ is even). In this case, a tighter upper bound for an optimal path is needed. In fact, we can show the following lemma similar to Lemma 6.2:

LEMMA B.3. *Suppose $n = 2k + 1$, and let I_1 and I_2 be two* distinct *$(k+1)$-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.*

Proof. Let $S_{in} = I_1 \cap I_2$, and $S_{out} = S \setminus (I_1 \cup I_2)$. If $|S_{in}| = \ell$ ($0 < \ell < k + 1$, since I_1 and I_2 overlap and also they are distinct), then $|I_1 \cup I_2| = 2k + 2 - \ell$, and thus $|S_{out}| = \ell - 1$. In a Hamiltonian path on S, to avoid having any edge joining two nodes both in I_1 or both in I_2, every node in S_{in} must connect only to nodes in S_{out}. Since $|S_{in}| = |S_{out}| + 1$, this means that the path must alternate between the nodes of S_{in} and the nodes of S_{out}, starting and ending at some nodes both in S_{in}, *without* visiting the nodes outside $S_{in} \cup S_{out}$ (such left-over nodes exist since $|S \setminus (S_{in} \cup S_{out})| = 2k + 1 - (\ell + \ell - 1) > 0$), a contradiction. □

This says that the length of the *second shortest $(k+1)$-interval* gives an upper bound for d^*, the optimal objective value. We claim that using the fake-point algorithm, where we apply the algorithm of Section 6.2 to construct cycle C^* on $n' = 2k+2$ points, achieves this upper bound. Recall from Section 6.2 that each edge of C^* spans either $k' = k + 1$ and $k' + 2$ nodes on two sides of the schematic cycle (type 1 edge), or $k' + 1$ nodes on both sides (type 2 edge). When we remove Q to obtain P^*, each edge spans one less node on one side of the schematic cycle. The only possibility for an edge of P^* to span less than $k + 1$ nodes is when the k' nodes (on the schematic cycle) spanned by some type 1 edge include Q. But this means that these k' nodes are counted by wrapping around the line, and thus such an edge actually spans $k' + 2$ nodes on the line. Therefore P^* has each edge spanning at least $k + 1$ nodes (while avoiding both endpoints in the shortest $(k+1)$-interval), and thus is optimal.

We can also use the following algorithm to solve the problem directly (we thus call the method *direct algorithm*):

> **When $n = 2k$:**
> Connect node p to node $(p + k)$, for all $p = 1, 2, \ldots, k$; also connect node $[(k + 1) + q]$ to node q, for all $q = 1, 2, \ldots, (k - 1)$.
> **When $n = 2k + 1$:**
> Identify the shortest $(k+1)$-interval I_1. Re-label the nodes of S in a wrap-around fashion, with the nodes of I_1 labeled $1, \ldots, (k+1)$. Put the nodes in a schematic cycle as in Fig. B.2. Use the new labels to construct the following path: start visiting node 1, and repeatedly visit the $(k + 1)$-st node counted *counterclockwise* from the current node (which is counted as the first node), until reaching node $k + 1$ and stop. The resulting path is $1, (k+2), 2, (k+3), 3, \ldots, (k+1)$ (see Fig. B.2).

It is easy to see that when $n = 2k$, the constructed path has each edge spanning

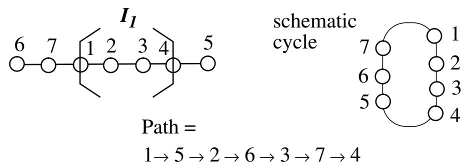

FIG. B.2. *Example of an optimal Hamiltonian path specified by the* direct algorithm *when n is odd ($n = 7$).*

$k+1$ or $k+2$ nodes (in fact, the path is the same as the one constructed by the fake-point algorithm), and when $n = 2k+1$, the constructed path has each edge spanning respectively $k+1$ and $k+2$ nodes on two sides of the schematic cycle while avoiding both endpoints in I_1. Therefore the path is optimal.

THEOREM B.4. *Let S be a set of n points lying on a line and the distances are Euclidean. Then there exists an algorithm that solves the path version of the maximum scatter TSP on S exactly in optimal $O(n)$ time.*

B.3. Path Version: Points on a Circle. Because of the wrap-around effect, the fake-point algorithm does not work: it is not possible to place Q so that Q is far away from *all* points of S. When $n = 2k+1$, the above direct algorithm readily takes care of the wrap-around effect. Also, since the length of the second shortest $(k+1)$-interval is (an upper bound for) d^*, the issue that an edge spanning all $k+1$ nodes of a $(k+1)$-interval I may not be the longest edge within I is resolved by Lemma B.1. Therefore we can solve the problem by scanning all n possible $(k+1)$-intervals to identify the shortest one (whose length is given by the edge joining its two endpoints), and applying the direct algorithm. When $n = 2k$, the problem is more involved: all k edges $(1, k+1), (2, k+2), \ldots, (k, 2k)$ each span $k+1$ nodes on *both* sides of the circle, and thus it is not possible to have a Hamiltonian path with each edge spanning at least $k+1$ nodes (as in the case of points on a line) — a tighter upper bound is needed. We do not explore the problem any further; however, if the points of S are *equally spaced*, then it is easy to see that the above direct algorithm gives an optimal Hamiltonian path.

COROLLARY B.5. *Let S be a set of n points lying on a circle and the distances are Euclidean. If n is an odd integer or the points are equally spaced, then there exists an algorithm that solves the path version of the maximum scatter TSP on S exactly in optimal $O(n)$ time.*