ICS 2011
12th INFORMS Computing Society Conference

© 2011 INFORMS | ISBN 978-0-9843378-1-1
DOI 10.1287/ics.2011.0004

A Steiner-Zone Heuristic for Solving the Close-Enough Traveling Salesman Problem

William K. Mennell
BAE Systems, AIT, Burlington, Massachusetts 01803, wmennell@gmail.com

Bruce L. Golden
R. H. Smith School of Business, University of Maryland, College Park, Maryland 20742, bgolden@rhsmith.umd.edu

Edward Wasil
Kogod School of Business, American University, Washington, DC 20016, ewasil@american.edu

Abstract In the Close-Enough Traveling Salesman Problem (CETSP), if a salesman is within a specified distance of a node, then the node has been visited. This paper presents a method for solving the CETSP that is based on Steiner zones. We generate test problems and conduct extensive computational experiments comparing our method to other heuristics. Overall, our method is very fast and improves upon heuristics from the literature.

Keywords traveling salesman problem; unmanned aerial vehicle; routing; Steiner zone; CPLEX; Concorde; Lin-Kernighan; generalized traveling salesman problem; genetic algorithm; second-order cone program

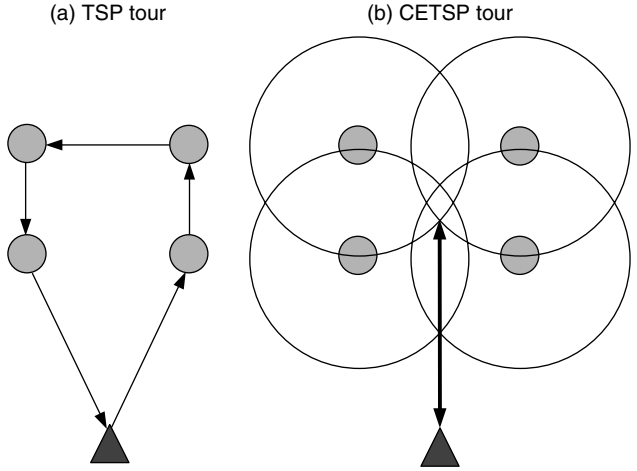
1. Introduction

The Traveling Salesman Problem (TSP) is one of the most studied problems in operations research. A salesman starts and ends at the same city and must visit n cities exactly once traveling the least distance possible. Given a complete, undirected graph $G = (V, E)$, where V is the set of n nodes (cities), E is the set of edges connecting the nodes, and each edge has a length (distance), we must find a tour (Hamiltonian cycle) that minimizes the sum of the lengths of the edges in the symmetric TSP. The edge connecting nodes i and j is denoted e_{ij}. The book by Applegate et al. [1] provides a detailed history of the TSP including applications and computational aspects.

This paper examines the following variant of the TSP. If a salesman is within a specified distance of a node (in other words, a salesman is "close enough"), then the node is considered to have been visited. We illustrate the CETSP in Figure 1. In the standard TSP shown in Figure 2(a), the salesman leaves the depot (depicted by a triangle), visits four nodes, and returns. In the CETSP shown in Figure 2(b), the salesman needs to get within distance r of each node i (this is depicted by the four disks), where r is the same for each node. If the salesman visits a location in the intersection of the four disks, then each node has been visited. Clearly, the out-and-back solution to the CETSP has a shorter total length than the solution with five edges required by the TSP. In the CETSP, the salesman moves freely in 2D Euclidean space, whereas, in the TSP, the salesman travels from node to node.

The CETSP has a number of practical applications. Consider the deployment of a drone aircraft (unmanned aerial vehicle) to monitor several geographic regions. To collect its reconnaissance information, the aircraft needs to fly within a certain distance of each region's

FIGURE 1. A comparison of an optimal TSP tour on five cities to an optimal CETSP tour on the same cities.



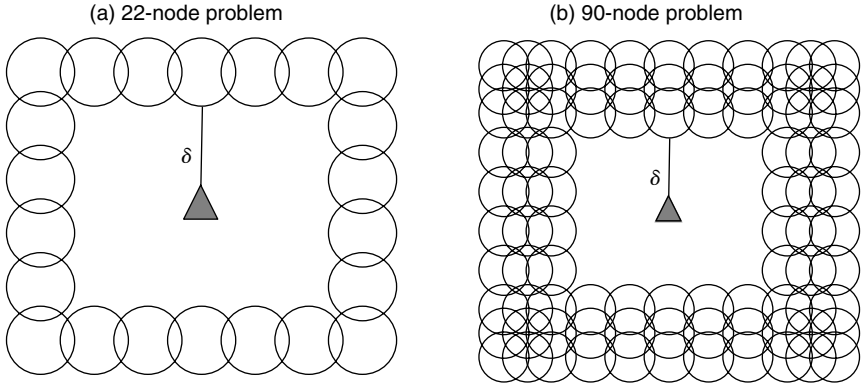
Notes. An optimal CETSP tour is never longer than the optimal TSP tour since any TSP tour is feasible to CETSP. In this example, a simple out-and-back trip from the origin to the intersection of the four disks defines the optimal CETSP tour since visiting the intersection gets close enough to the four cities simultaneously and the straight line minimizes the distance from the depot to the intersection.

center. Resource constraints require that a flight path (tour) must be followed that gets close enough to the center of each region while simultaneously minimizing flying time (distance traveled).

Utility companies use automated meter-reading with radio frequency identification (RFID) to read utility meters from a distance. A meter reader can record the use of water, gas, or electricity within 500 to 1,000 feet of a customer and does not need to traverse every street in a neighborhood. The radius of RFID coverage transforms the traditional meter-reading problem from a TSP to a CETSP.

This paper develops a solution method based on Steiner zones, generate a set of test problems, and conduct extensive computational experiments in which we compare the results of our method to the results of other methods, including a genetic algorithm, a tour generation and modification heuristic, and a greedy heuristic.

FIGURE 2. Two CETSP instances for which the lower bounds on the optimal objective function values are equal, but the optimal objective function values are much different.



Note. Subfigures are not drawn to scale; δ is equal in both.

2. Literature Review

The CETSP is a specific instance of three more general problems that appear in the literature, the Generalized Traveling Salesman Problem (GTSP) (Fischetti et al. [9], Silberholz and Golden [19]), the Geometric Covering Salesman Problem (GCSP) (Arkin and Hassin [2], Gendreau et al. [10]), and the Traveling Salesman Problem with Neighborhoods (TSPN) (Dumitrescu and Mitchell [6], Elbassioni et al. [8], Mitchell [15]). For a thorough discussion of each problem and its relationship to the CETSP see Mennell [14]. Our work focuses solely on the CETSP and should produce better results than any algorithm designed for more general problems such as the GCSP.

The first description of the CETSP is given by Gulczynski et al. [11]. They consider utility companies that use RFID to read meters from a distance. The authors survey six heuristics (three variants of tiling, radial adjacency, sweeping circle, Steiner zone) for solving problems with Euclidean distances and equal r for all disks. The six heuristics have three steps in common.

- *Step* 1. Find a set of points S such that each customer i is within r units of at least one point in S. Each point in S is called a supernode since visiting it allows the salesman to simultaneously visit multiple nodes.
- *Step* 2. Solve a TSP on the points in S to produce a near-optimal solution T that is a feasible solution to the CETSP.
- *Step* 3. Reduce the distance traveled in T while maintaining feasibility.

Dong et al. [4] formulate the CETSP as a mixed integer, nonlinear program (MINLP), but they do not attempt to solve it. Instead, they develop two heuristics that use clustering (merged tiling) or convex hulls to generate the supernodes. The TSP tour over the supernodes is constructed with a convex hull insertion algorithm. The tour is improved with a simulated annealing algorithm. The authors test their two heuristics on 190 random Euclidean problems with 100 to 1,000 customers and 10 different radii. Both heuristics produced good solutions and were very fast with running times under one second.

Yuan et al. [20] consider the problem of routing a mobile robot to visit sensors that are distributed in the Euclidean plane. In order to communicate with a sensor and download data, a robot must be within a certain distance of it (the effective range is specified by a disk and the disks do not overlap). The authors develop an evolutionary approach for solving the robot routing problem and compare its performance to an approximation algorithm on five problems with 19 to 300 sensors. The evolutionary approach generated the shortest tour for each problem although the convergence speed was slow.

Shuttleworth et al. [18] extend the CETSP to street networks. They consider a real-world meter-reading problem with actual data. The authors' heuristic applies a two-stage process. In the first stage, a heuristic selects a subset of street segments to be traversed based on the radius r. In the second stage, street segments are added to the subset of street segments from the first stage to form a cycle. Shuttleworth et al. [18] consider four simple heuristics and two integer programs for selecting the street segments. For example, the weighted bang-for-buck heuristic first selects the street segment that covers the largest number of customers, then selects the street segment that covers the largest number of additional customers, and so on until each customer is covered by at least one street segment. The authors use a commercial vehicle routing system to produce a complete travel path that starts and ends at the depot. Computational results showed that the weighted bang-for-buck heuristic was simple and effective, and generated solutions that saved time and distance when compared to the existing route used by the utility company.

2.1. Formulation and Lower Bounds

Mennell [14] describes a MINLP formulation of the CETSP. Since we do not use that formulation directly in this paper, we do not include it here. The CETSP is a generalization

of the TSP (when $r = 0$ for all nodes, the CETSP reduces to the TSP) and is, therefore, NP-hard. The standard 1-tree (Held and Karp [12]) and 2-matching (Edmonds [7]) lower bounds techniques for TSPs cannot be applied to the CETSP without significant modifications, so we do not use them. Instead, we construct a lower bound in the following way. Let $\bar{d}_{ij} = \min(0, d_{ij} - r)$ when either i or j is the depot, where d_{ij} is the distance between nodes i and j. Otherwise, $\bar{d}_{ij} = \min(0, d_{ij} - 2r)$. If disks i and j overlap, then $\bar{d}_{ij} = 0$. We fix each edge in G to its minimum possible distance and solve a TSP to optimality using the set of all $\bar{d}_{ij}$. In Figure 2, we show two problems that differ in size and optimal CETSP solution but have the same lower bound using our method. Each layer of squares overlaps the next one so that $\bar{d}_{ij} = 0$ for all overlapping disks. Given distances $\bar{d}_{ij}$, the optimal tour lengths are the same for both problems.

Clearly, this lower bound is weak. Unless there is no disk overlap and r is small, the lower bound is often 0 and provides little information about the optimality gap.

Another lower bound technique is the longest out-and-back tour from the depot to any node. A CETSP cannot have an optimal solution shorter than twice the length of the shortest straight line from the depot out to the boundary of the disk furthest from the depot. This bound is also very weak.

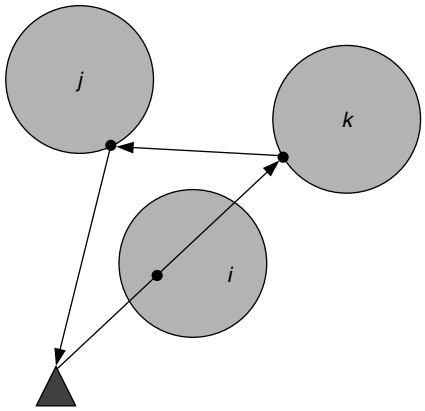
Developing tight lower bounds for the CETSP is a nontrivial task that is beyond the scope of this paper.

3. Steiner-Zone Heuristic

In the CETSP, a tour must cross a disk of radius r surrounding every node and visit the *representative point* selected for each disk as shown in Figure 3. Note that for any tour, a representative point in a disk's interior can always be replaced by a point on the disk's boundary without changing the tour length. In Figure 4, we show how different choices of representative points affect the edge distance between nodes i and j. Node i is a fixed point in the plane at coordinate (a_i, b_i). We refer to the disk surrounding node i as disk(i). Any tour that touches or passes through disk(i) successfully visits node i (with the exception of the depot (node 0) which has no disk). We denote the center of disk(i) as orig(i), the origin of disk(i), to emphasize that each node is the center of a disk. We assume that orig(i) $\neq$ orig(j) for all i and j.

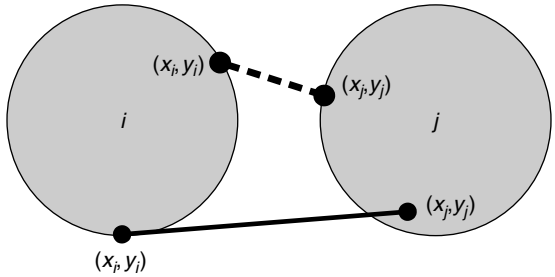
A Steiner zone of degree k, denoted SZ(k), is the nonempty intersection of k convex regions. In this case, the regions are disks. A supernode is any point that simultaneously

FIGURE 3. A four-node CETSP instance and solution depicting the representative point chosen for each disk.



Note. For this set of tour edges, nodes j and k have only one feasible representative point while node i can be represented by any point in the edge crossing i's disk.

FIGURE 4. Edges connecting two possible sets of representative points for nodes i and j.

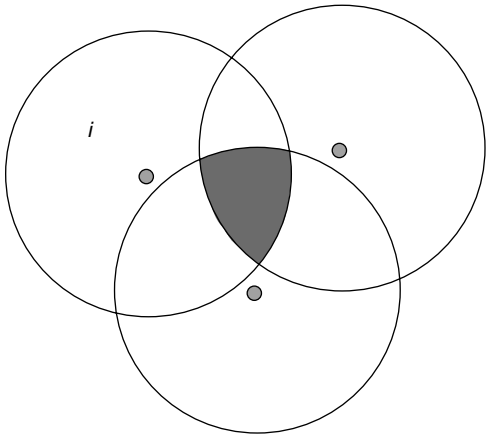


Note. To minimize tour length, representative points must be chosen carefully.

visits more than one node. SZ(k) contains all possible supernodes for its k member nodes, as shown in Figure 5. A Steiner zone of degree 1, SZ(1), is called a *singleton*. For each SZ(k), we designate the node upon which it is built as that Steiner zone's *base node*; the need for a base node will become clear in the next paragraph. For example, node i is the base node for the SZ(k) in Figure 5. Since a point in a Steiner zone is not in general one of the original nodes to be visited and yet a visit to this point may reduce the total length, we envision these zones affecting the TSP as Steiner points affect the minimal spanning tree problem. The name Steiner zone comes from this observation.

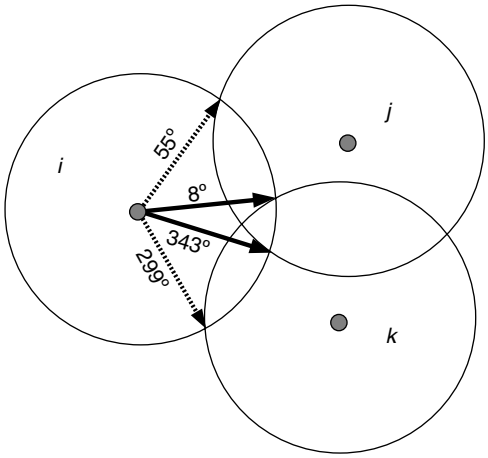
An SZ(k) with base node i is characterized by two polar angles, relative to $orig(i)$, with which the other $k-1$ disks intersect $disk(i)$, and the origins of the $k-1$ disks. The two angles are called upper and lower angles of intersection and are easily generated. For $k \geq 3$, the angles of intersection are obtained by comparing the angles of intersection for two Steiner zones with base node i, an SZ($k-1$) and an SZ(2). If the angles overlap, then the upper and lower angles of that overlap become the upper and lower angles of intersection for the SZ(k). For example, in Figure 6, the SZ(2) formed by $disk(i)$ and $disk(j)$, with base node i, has lower and upper angles of intersection of 343° and 55°, respectively. The SZ(2) formed by $disk(i)$ and $disk(k)$, with base node i, has lower and upper angles of intersection of 299° and 8°, respectively. The two sets of angles overlap such that the SZ(3) formed by $disk(i)$, $disk(j)$, and $disk(k)$, with base node i, has a lower angle of intersection of 343° and an upper angle of intersection of 8°. When two or more disks intersect at exactly one point, the upper and lower angles of intersection are equal. As long as we know the base node and the upper and lower angles of intersection, we can easily generate a point within r of

FIGURE 5. The shaded region defines a Steiner Zone of degree 3 with base node i and is denoted SZ(3).



Note. SZ(3) is the intersection of the three disks.

FIGURE 6. The base node is i.



Note. Only the lower (343°) and upper (8°) angles of intersection are required to specify the SZ(3) formed by $disk(i)$, $disk(j)$, and $disk(k)$.

each $disk(i)$ within an SZ(k). In Figure 7, we show two Steiner zones of equal degree with different size overlaps.

In Figure 8, we see that the larger the radius, the larger the potential savings in tour length. For example, if r is very large, we can expect a much shorter optimal CETSP tour than the optimal TSP tour. If r is near zero, an optimal CETSP result will be only slightly

FIGURE 7. The amount of disk overlap does not matter: The Steiner zone has degree 2 in both cases.

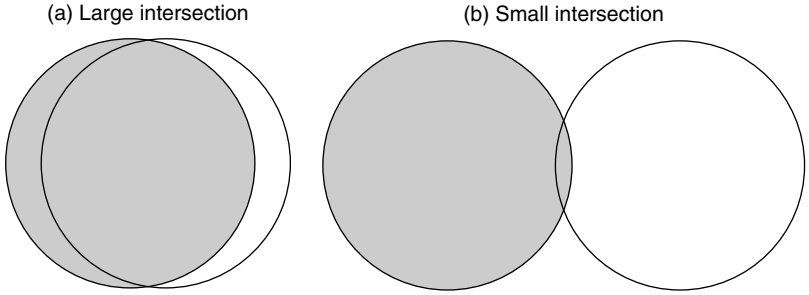
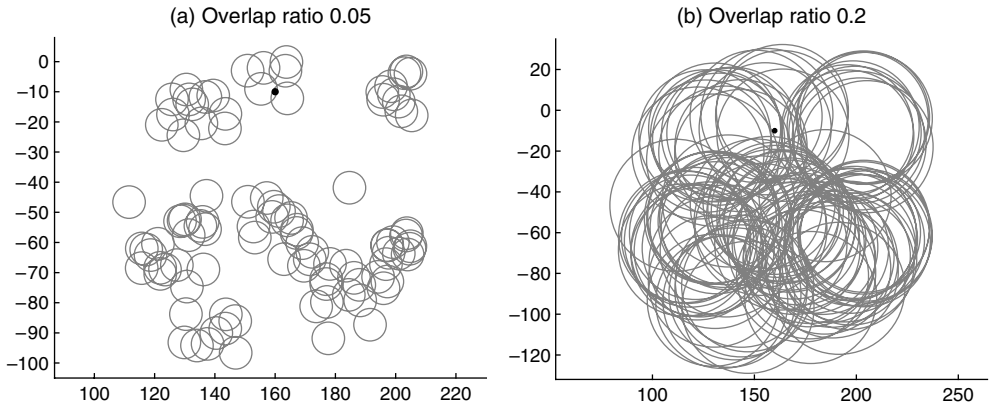


FIGURE 8. The nodes are the same in (a) and (b), but the radius is larger in (b).



Note. The larger the overlap ratio, the greater the potential savings.

shorter than the optimal TSP solution. We define a metric, *overlap ratio*, to indicate how much potential there is for improvement over the TSP solution. For a given CETSP, let l_{contain} be the length of a side of the smallest square containing all n disks. The overlap ratio is the ratio of r to l_{contain}. In Figure 8, we show two different overlap ratios for the same set of nodes. We consider instances with an overlap ratio much larger than 0.3 uninteresting because almost any set of Steiner zones can lead to a near-optimal solution of the CETSP. Overlap ratio is not appropriate for problems which lie in an elongated rectangle rather than a square, but it provides an indication of how much improvement is possible.

Our Steiner-zone heuristic, denoted by SZH, reduces a graph of n nodes to a graph of m Steiner zones, sequences the m zones, and locates the best representative point for each zone given the tour sequence. Node 0 is the depot and cannot be part of any Steiner zone. We now describe the three phases of SZH in detail.

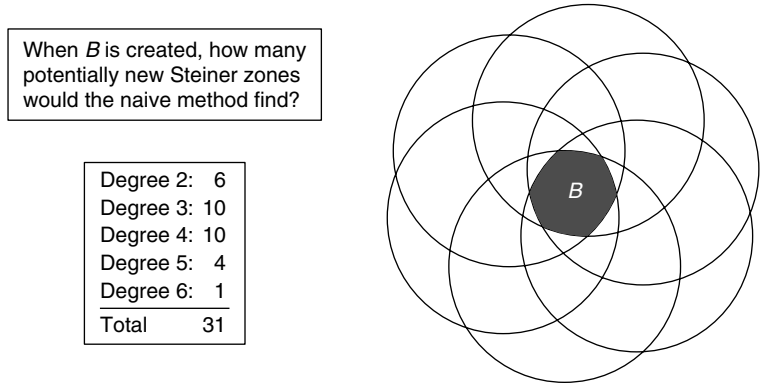
3.1. Phase I—Graph Reduction

We describe two approaches to graph reduction. The first is a slow, naive heuristic and the second an improved, faster version of the first. We begin by sorting the nodes and storing them in a list, denoted list. Because the second version is so fast, we run the entire SZH repeatedly, once for each of four different sorts: closest to and furthest from the depot, and closest to and furthest from the centroid of the graph. We select the first node in list, say i, and identify all Steiner zones that node i can form, storing them in a list called SZ_list_i. To do this, we consider all other nodes and calculate their angles of intersection with i, if possible. Let SZ_{ij} denote the SZ(2) formed between nodes i and j. We compare SZ_{ij} with all previously identified Steiner zones already in SZ_list_i. Whenever SZ_{ij}'s angles of intersection overlap those of a Steiner zone in SZ_list_i, we form a new Steiner zone from their intersection. $\text{SZ}Y\text{_list}_i$ is ordered from highest degree to lowest degree. That is, the Steiner zones capturing the intersection of a large number of disks are stored before those describing the intersections of fewer disks. Steiner zones of equal degree are ordered lexicographically. For example, an SZ(3) consisting of nodes 1, 5, and 6 would be ordered prior to an SZ(3) consisting of nodes 1, 6, and 9. With this ordering, we can easily locate a particular Steiner zone in SZ_list_i. To avoid a long search through SZ_list_i, we use bookmarks that store the location of the first Steiner zone in the list of each degree, k, for small values of k.

Consider the following example. We have selected node 1 from list and have compared it with nodes 2 through 5, such that SZ_list_1 is: $\{1,2,3\}$, $\{1,2\}$, $\{1,3\}$. Considering node 6 next, we find that nodes 1 and 6 form an SZ(2). The lower and upper angles of intersection are 40° and 130°. We compare these angles against the angles of intersection for the three Steiner zones already in SZ_list_1 and find that $\{1,6\}$ overlaps $\{1,2,3\}$. We have found a new Steiner zone, $\{1,2,3,6\}$ and must add it to the list along with all sub-Steiner zones associated with it. SZ_list_1 now looks like: $\{1,2,3,6\}$, $\{1,2,3\}$, $\{1,2,6\}$, $\{1,3,6\}$, $\{1,2\}$, $\{1,3\}$, $\{1,6\}$. After constructing all Steiner zones for node i, we store the highest degree Steiner zone in set SZ_set for later use, and remove its nodes from list. This is repeated until list $= \varnothing$. When there are no nodes to consider, a set of Steiner zones remains, SZ_set, such that any tour visiting each of these Steiner zones is a feasible solution to the CETSP.

In Figure 9, we illustrate the shortcoming of this naive approach to graph reduction. Assume $B = \text{SZ}(k)$ is found, where k is an integer larger than three. There is a set of unique sub-Steiner zones of degrees $k-1, k-2, \ldots, 2$, such that all members of the set contain B and may or may not have been identified yet. Therefore, we must identify all of them individually. As k grows, the number of possible sub-Steiner zones to identify grows exponentially. The vast majority of them, however, are contained in other larger Steiner zones and are thus redundant: this creates giant lists of Steiner zones to store and search. Early experiments confirmed that this approach to graph reduction is computationally expensive. Because of the large number of sub-Steiner zones that must be identified, a large amount of computation time (on the order of days) is needed to find Steiner zones of degree as small as 20.

FIGURE 9. B is an SZ(6).



Note. In the naive Phase I heuristic, all of B's sub-Steiner zones would be examined; in general, the number of sub-Steiner zones explored to find each new SZ(k) with this heuristic grows exponentially with k.

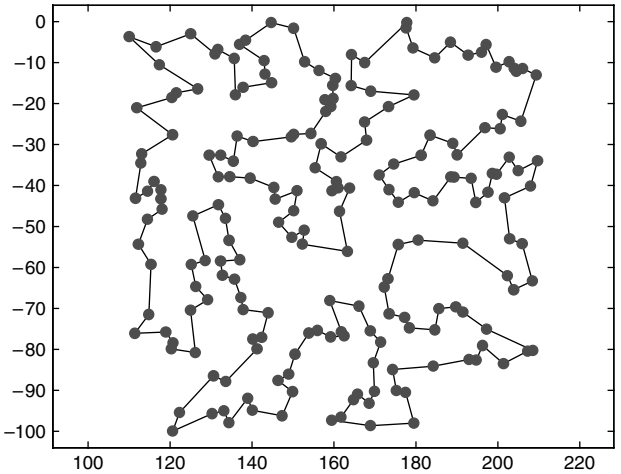
Our second approach to Phase I differs from the naive heuristic only in its treatment of sub-Steiner zones. In addition to adding B to SZ_list$_i$, we add only those sub-Steiner zones of B that have degree no greater than three and have not previously been added. In computational experiments, our second approach identifies Steiner zones with degrees in the hundreds in less than a second. The sub-Steiner zones of degree two and three seem to act as a suitable seedbed upon which to construct the Steiner zones of large degree. The choice of three as a cut-off value is arbitrary. In our experience, using a larger value slows the graph reduction without finding Steiner zones of much higher degree.

In Figure 10, we show the optimal TSP tour for a 200-node problem described in Mennell [14] as team2_200. In Figure 11, we show the Steiner zones that result from applying Phase I to team2_200. The Steiner zones happen to be disjoint. However, the Phase I algorithm does not explicitly require or encourage disjoint Steiner zones, though they are typically so.

3.2. Phase II—Tour Finding

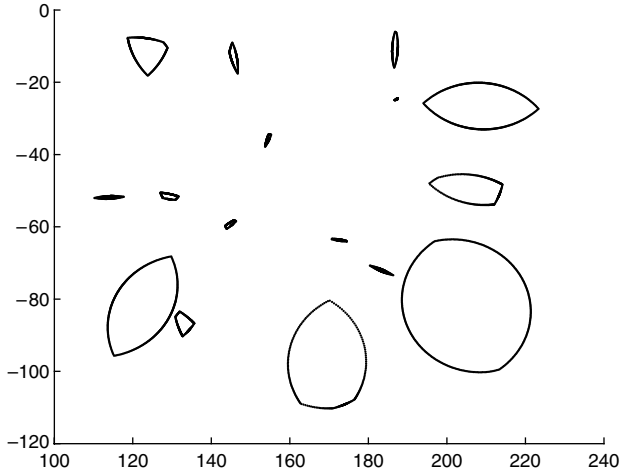
The result of applying Phase I to graph G is graph G'. G' is a graph of Steiner zones rather than nodes. We designate a point in each Steiner zone as its representative point and solve a TSP on these points.

FIGURE 10. A 200-node CETSP instance (team2_200) with the optimal TSP solution shown.



Note. Tour length is about 1,074.

FIGURE 11. The solution to Phase I for the team2_200 instance.



Note. The 200 nodes have been reduced to 16 Steiner zones.

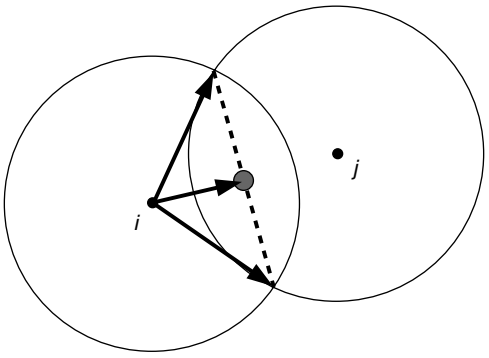
Given an SZ(k) with base node i and $k \geq 2$, a chord may be drawn connecting the points on $disk(i)$ at the upper and lower angles of intersection. Because the intersection of convex sets is convex and an SZ(k) is the intersection of k convex sets (disks), any point along this chord (we select the midpoint of the chord) must lie within the SZ(k). We show this in Figure 12.

For singletons, we choose the point on the disk's boundary closest to the depot. A well-known method, such as the Concorde TSP Solver [3], Applegate et al. [1], can be used to find the TSP tour on this set of points. The result is a feasible solution for the CETSP. Note that many TSP solvers require integer node coordinates. Our test problems do not have integer coordinates. This obstacle is easy to overcome (see Mennell [14] for details).

3.3. Phase III—Tour Improvement

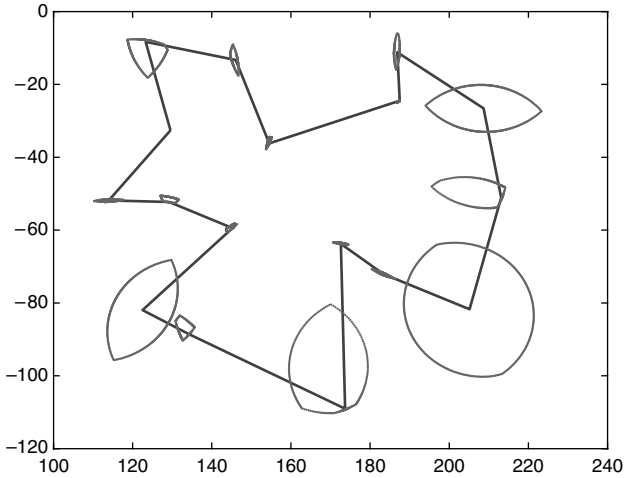
The result of applying Phase II to G' is a feasible solution to the CETSP. The sequence in which the tour visits the Steiner zones is fixed, but the location of each Steiner zone's representative point can still be modified. In Figure 13, we show the results after applying Phase II to team2_200. We notice that: (1) the tour length is much less than the tour length of the optimal TSP solution, and (2) the tour could be shortened by changing the placement

FIGURE 12. An example calculation of a Steiner zone representative point.



Notes. The base node for this Steiner zone is node i. The boundary of $disk(j)$ has at most two points of intersection with the boundary of $disk(i)$. The representative point is the midpoint of the chord that connects the two points of intersection.

FIGURE 13. The solution to Phase II for the team2_200 instance.



Notes. After selecting a single point to represent each Steiner zone found in Phase I, a TSP tour is found. The tour length is about 435.

of representative points for each Steiner zone. Given the sequence of Steiner zones produced by Phase II, we solve the Touring Polygons Problem (TPP) described in Dror et al. [5] to determine the optimal placement of the representative points. We call this the Touring Steiner Zones Problem (TSZP) since we consider convex regions (Steiner zones) instead of polygons. Given a tour on a set of disks, the TSZP minimizes the distance such that each Steiner zone (and thus each disk) remains feasibly covered and the sequence of Steiner zones visited remains unchanged.

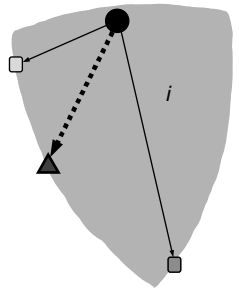
We can formulate the TSZP as a second-order cone program (SOCP) and solve it to optimality using CPLEX. Our heuristic approach, denoted IP_{PhIII}, is briefly described below and illustrated by means of an example in A. Both methods are examined in detail in Mennell [14].

IP_{PhIII} utilizes two initialization iterations, 1 and 2, to find a 60° bounded region on each Steiner zone's boundary where the optimal representative point is likely to be. In iteration 1, we replace each Steiner zone with the three points on its boundary with polar angles 0°, 120°, and 240° relative to its representative point. In iteration 2, we choose the points at 60°, 180°, and 300°. This ensures that the bounded region produced for each Steiner zone spans 60° of its boundary. Letting the depot represent both the departure and destination point, we find the shortest path for the network in iteration 1 and the shortest path for the network in iteration 2.

For iterations $k \geq 3$, we use the solution points from the previous two iterations to converge towards a near-optimal solution where the representative points are typically, but not restricted to be, in the 60° bounded regions produced by iterations 1 and 2. We replace each Steiner zone with three points and solve the shortest path problem on the resulting network. For each Steiner zone, we let the first point be the solution point chosen in iteration $k-1$, the second point be the solution point chosen in iteration $k-2$, and the third point be the point on the Steiner zone's boundary at the polar angle bisecting the other two points. We illustrate this in Figure 14. If the same points were chosen in iterations $k-1$ and $k-2$, then points two and three are chosen from the Steiner zone's boundary on either side of the point chosen in iteration $k-1$ (except in iteration 5—see Mennell [14]). IP_{PhIII} typically converges to a satisfactory level by iteration 10, but more or fewer iterations may be used.

In Figure 15, we show the solution to team2_200 after solving Phase III with IP_{PhIII}. We point out that a single run of SZH involves a complete iteration of the Phase I–Phase II–Phase III cycle for each of the four sorting criteria mentioned in §3.1 (we found that

FIGURE 14. Choosing the third point to approximate SZ_i in IP_{PhIII}.



Notes. The circle is the representative point for SZ_i. The squares pointed at by solid lines illustrate the points selected by the previous two iterations of IP_{PhIII}. The third point used to approximate SZ_i in the next iteration is the triangle pointed at by the dashed arrow. This point lies on the boundary of SZ_i at the angle bisecting the other two points.

at least one of the four always produced a shorter tour than a single run using a random ordering).

4. Heuristics

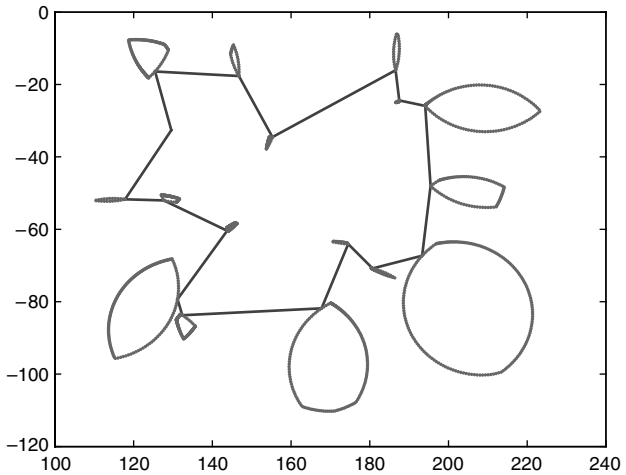
We coded 15 CETSP solution methods for comparison purposes.

SZ1 and SZ2 are variants of SZH. SZ1 applies the IP_{PhIII} heuristic to solve Phase III. After Phase III, we re-apply Phase II and stop. SZ1 emphasizes computational speed over tour length. SZ2 solves Phase III to optimality using the SOCP formulation. SZ2 repeats the Phase II–Phase III cycle until no improvement is found. SZ2 emphasizes tour length over computational speed.

LK-SOCP finds a near-optimal TSP tour on the original n nodes using the Concorde (Concorde TSP Solver [3], Applegate et al. [1]) implementation of the Lin-Kernighan heuristic (Lin and Kernighan [13]) and then minimizes the distance of the tour by solving the SOCP formulation to optimality. That is, LK-SOCP solves Phases II and III of SZH. It does not use Steiner zones.

YUAN+ finds the optimal TSP tour on the original n nodes using Concorde (Concorde TSP Solver [3]) and then minimizes the distance of the tour by solving the SOCP formulation to optimality. YUAN+ differs from LK-SOCP only in that Phase II is solved to optimality

FIGURE 15. The solution to Phase III for the team2_200 instance.



Note. The final tour length is about 304.

whereas LK-SOCP finds a heuristic solution. We think of YUAN+ as an improved version of the algorithm given in Yuan et al. [20]. Their algorithm finds the optimal TSP tour but solves Phase III with an evolutionary algorithm. The evolutionary algorithm does not guarantee the optimality of the Phase III solution whereas the solution to the SOCP solved in YUAN+ is the optimal Phase III solution.

GTSP1 finds near-optimal solutions to a generalized TSP (GTSP) approximation of the CETSP using a genetic algorithm developed by Silberholz and Golden [19]. On average, their algorithm was only 0.03% from optimality on the standard GTSP test problems (up to 442 total nodes) and found new best solutions for the larger problems with unknown optimal solutions. Our GTSP problem sizes range from 2,400 nodes to 24,000 nodes plus the depot.

By replacing each disk with k points, we create a GTSP where there are n clusters to be visited, each with k possible points to choose from. We used 3-, 6-, 12-, and 24-point approximations of each disk. Only the 24-point approximations were competitive, however, so we do not report on the 3-, 6-, and 12-point approximations.

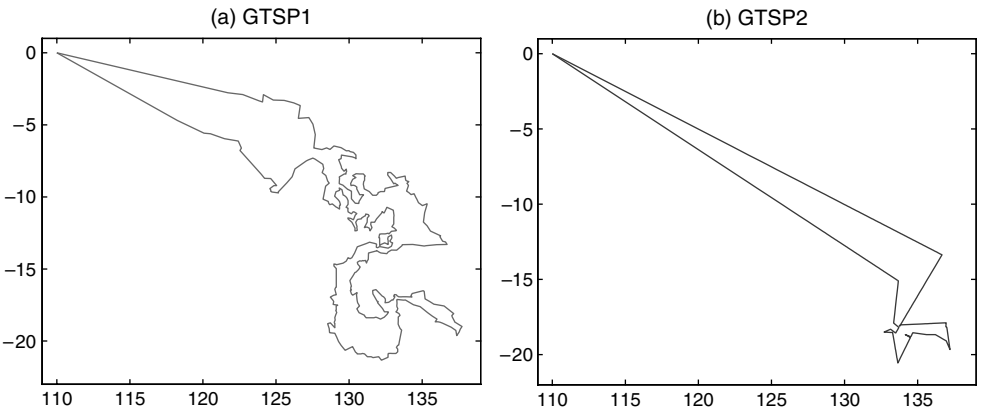
Because GTSP1 solves a discrete approximation of the CETSP, we can improve upon it by fixing the sequence of disks in the GTSP1 tour and solving Phase III to optimality with the SOCP. This is what GTSP2 does. In Figure 16, we show the difference between a GTSP1 solution and a GTSP2 solution for a problem with high overlap ratio.

HYBRID1 and HYBRID2 are hybrid algorithms that combine SZH with the GTSP approximation of the CETSP. HYBRID1 and HYBRID2 apply Phase I of SZH to reduce the graph of n nodes to m Steiner zones. The boundary of each Steiner zone is discretely approximated by 24 points and a GTSP is solved on the set of $m+1$ clusters and $24m+1$ points (the depot is a cluster containing exactly one point). Phase III of SZH is applied to the GTSP solution, with HYBRID1 using the $\mathrm{IP}_{\mathrm{PhIII}}$ heuristic and HYBRID2 solving to optimality with the SOCP. The GTSP and Phase III are solved only once.

CTD, circular trapezoidal decomposition, applies a modified trapezoidal decomposition algorithm of Mulmuley [16] to subdivide a graph of disks into nonoverlapping regions such that each region is bordered on the left and right by vertical lines and on the top and bottom by arcs of a disk. We show such a subdivision in Figure 17. We compute the degree of each distinct subdivision in the graph and greedily select the regions with the highest degree until the n original nodes are covered. We apply Phase II to the selected subdivisions and solve Phase III to optimality using the SOCP formulation. See Mennell [14] for more details.

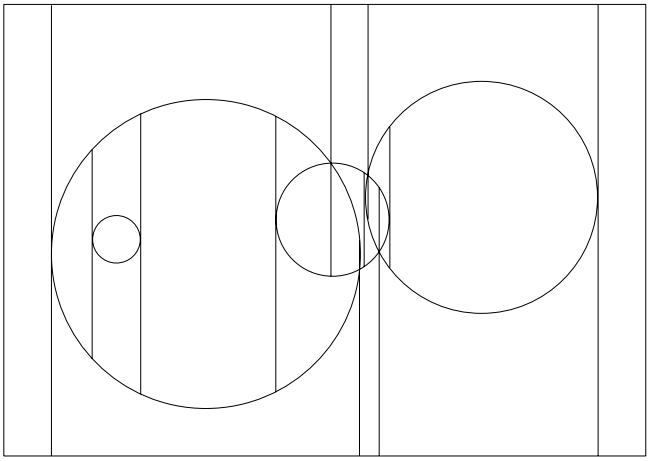
TILE1, TILE2, TILE3, and TILE4 use the shifted tiling method from Gulczynski et al. [11]. The plane is tiled with regular hexagons. Every node is no more than r units from

FIGURE 16. Solutions produced by the two GTSP-based heuristics for a 500-node CETSP.



Notes. GTSP2 solves Phase III to optimality on the sequence of nodes in the GTSP1 tour. In this example, the improvement is substantial because the overlap ratio is large (0.3).

FIGURE 17. An example of the circular trapezoidal decomposition of a graph of disks.



Note. Every point inside bounded region i of the decomposition has the same degree of intersection.

the center of at least one hexagon. Tile centers are shifted a small step-size in an attempt to reduce the number of tiles needed to cover all n nodes. A TSP is solved on centers of tiles that contain nodes and the tour distance is then further minimized using a heuristic.

The authors of Gulczynski et al. [11] provided their code for use in our experiments. Their code uses a genetic algorithm to solve Phase II. To ensure a fair comparison with our heuristics, TILE1 and TILE3 solve Phase II to optimality using the Concorde TSP Solver [3]. TILE2 and TILE4 find TSP tours using the chained Lin-Kernighan code from Concorde (Concorde TSP Solver [3]). TILE1 and TILE2 set the step-size equal to the disk radius. TILE3 and TILE4 set the step-size to 50 for all problems. This ensures that problems with a small radius spend as much effort finding supernodes as problems with a large radius.

MERGE and HULL are the merged tiling heuristic and the convex-hull heuristic, respectively, given in Dong et al. [4]. MERGE tiles the plane with regular hexagons so that each node is no more than r units from the center of at least one hexagon. If all nodes in a pair of adjacent hexagons can be covered by a single one, the two hexagons are merged. HULL creates a set of supernodes that cover the n original nodes using a convex-hull-based heuristic. MERGE and HULL solve a TSP on the supernodes using simulating annealing and further minimize the distance of the tour by perturbing the location of each supernode and resolving the TSP. The authors provided versions of their code optimized for computational speed to use in our experiments.

5. Computational Results

We selected seven TSPs from TSPLIB [17] that ranged in size from 100 to 1,000 nodes. Two problems have clustered nodes, one has nodes arranged in rough lines, two have uniformly distributed nodes, and two are drill-press problems. In order to create CETSPs from the TSPs, we assigned three different radius patterns: (1) the overlap ratio is 0.02 (very low), (2) the overlap ratio is 0.1 (moderate), and (3) the overlap ratio is 0.3 (very high). In a problem, the radius is the same for all disks. We refer to these as *TSPLIB* problems. We used the implementation of the Lin-Kernighan TSP solver provided in the Concorde software package. All problem data and the complete set of results, including the best-known solution for each problem, are given in Mennell [14].

We did not test our heuristic against the heuristic described in Shuttleworth et al. [18]. They solve the CETSP over a street network (the triangle inequality does not hold). None

of the algorithms in our experiments were designed to work for the CETSP over a street network.

All experiments were performed on a 2.4 GHz Pentium machine with 3 GB RAM running Windows XP Professional. We note that different computers have different floating-point architectures. Very small numerical differences can add up over a computation and affect the outcome of SZH. Thus, for a small number of problems, we obtained a different solution for the same problem on different computers. All solutions are checked for feasibility. All methods use the same code to solve Phase III. Except for the genetic algorithm and the hybrid heuristics, all solution methods use the same code to solve Phase II. No special attention was given to the Steiner-zone methods to improve performance. In the first table, bold font indicates the best value. Computation time is given in CPU seconds.

The tour-length results for the 15 heuristics on the seven *TSPLIB* problems using three overlap ratios and the Euclidean norm are given in Table 1. The average deviations from the best solutions in Table 1 are given in Table 2. Columns in both tables are ordered by the overall average deviation from best (left) to worst (right).

In Table 1, for the seven low overlap problems, GTSP2 produces all the shortest tours. For the seven medium overlap problems, GTSP2 produces the shortest tour five times. For the seven high overlap problems, LK-SOCP and YUAN+ each produce the shortest tours four times.

In Table 2, we see that, on average, GTSP2 and GTSP1 produce the smallest deviations (0.00% and 0.55%) for problems with a low overlap ratio. For problems with a medium overlap ratio, GTSP2 produces the smallest average (3.58%). For problems with a high overlap ratio, HYBRID2 and SZ2 produce the smallest average deviations (0.77% and 0.95%). GTSP2 and GTSP1 produce the largest deviations (40.62% and 103.50%). $rd400$ and $pcb442$ are challenging problems for all heuristics except for GTSP1 and GTSP2.

Over all 21 problems, the four methods that find Steiner zones using Phase I of SZH (HYBRID2, HYBRID1, SZ2, SZ1) have the lowest overall average deviations, ranging from 3.22% to 4.24%.

In Table 3, we give the computation times for the 15 heuristics on the *TSPLIB* problems. The columns are ordered from fastest (left) to slowest (right). All heuristics except for CTD, TILE1, TILE2, YUAN+, and LK-SOCP achieve faster times as the overlap ratio increases. CTD slows down because the trapezoidal decomposition takes much more time to compute as the overlap ratio increases. The running times for TILE1 and TILE2 are linked to the overlap ratio by the stepsize parameter (the radius, r, is the stepsize) so that higher overlap ratios lead to longer computation times. The running times for YUAN+ and LK-SOCP remain relatively constant over all overlap ratios. For methods that apply Phase I, the size of the TSP solved in Phase II decreases as the overlap ratio increases. YUAN+ and LK-SOCP do not apply Phase I, so their running times do not decrease as the overlap ratio increases.

HULL, MERGE, LK-SOCP, SZ1, and SZ2 are very fast, taking one-third to one and one-half seconds on average to solve a problem. The two methods based on the generalized TSP (GTSP1, GTSP2) are computationally expensive. They take an exorbitant amount of time to solve problems (on the order of several months for the largest problems!). The hybrid methods (HYBRID1, HYBRID2), though not as slow as the GTSP methods, are much slower than the other 11 methods.

We also used the Manhattan norm on the 21 *TSPLIB* test problems and found that HYBRID2, HYBRID1, SZ2, and SZ1 were the best methods based on overall accuracy. Computation times were comparable to the Euclidean results. The complete details are given in Mennell [14].

Based on the computational experiments with the *TSPLIB* problems using the Euclidean norm, we observe the following. (1) As the overlap ratio increases from low to high, tour length decreases in almost all cases (except for GTSP1 on problems $rat195$, $d493$, and $dsj1000$), and computation times decrease for many heuristics, including the Steiner-zone

TABLE 1. Tour lengths for 15 heuristics on *TSPLIB* problems with equal radius and 2D Euclidean norm.

	HYBRID2	HYBRID1	SZ2	SZ1	CTD	YUAN+	LK-SOCP	GTSP2	TILE2	TILE3	TILE1	TILE4	HULL	MERGE	GTSP1
Overlap ratio: 0.02															
kroD100	160.46	160.88	163.76	163.78	165.30	164.31	164.31	**159.05**	172.84	170.88	173.05	170.32	210.03	188.07	159.28
rat195	164.51	165.37	171.29	171.35	170.33	170.89	172.51	**158.79**	187.37	181.99	190.00	188.76	229.23	189.12	159.08
lin318	2,903.26	2,914.47	2,941.00	2,941.53	2,946.25	2,949.71	3,003.94	**2,867.46**	3,217.70	3,191.60	3,183.20	3,155.60	3,824.50	3,261.49	2,875.34
rd400	1,039.29	1,042.62	1,066.16	1,066.44	1,081.09	1,069.26	1,066.15	**1,033.42**	1,180.50	1,164.00	1,177.90	1,166.60	1,513.97	1,243.61	1,035.95
pcb442	327.26	328.72	333.23	333.27	339.19	340.59	351.61	**323.03**	371.53	359.93	359.47	366.64	450.22	401.09	324.21
d493	209.82	211.05	213.50	213.64	212.73	221.41	217.82	**204.71**	230.83	227.55	231.85	236.76	261.70	249.62	206.77
dsj1000	998.35	1,015.30	1,010.82	1,012.71	996.84	1,005.80	1,010.43	**935.74**	1,156.60	1,156.20	1,141.10	1,140.00	1,456.97	1,266.38	951.09
Overlap ratio: 0.1															
kroD100	96.68	97.58	96.68	97.09	95.21	95.64	95.64	**89.94**	102.49	116.63	115.81	114.81	107.85	121.88	91.53
rat195	73.89	74.41	74.16	74.22	74.67	77.66	88.44	**68.26**	87.74	82.34	84.12	84.45	85.83	90.81	72.25
lin318	1,554.59	1,563.48	1,556.82	1,557.47	1,529.60	1,557.07	1,471.93	**1,467.03**	1,711.80	1,842.00	1,646.80	1,679.50	1,886.08	1,828.95	1,555.56
rd400	521.91	526.96	521.91	526.73	535.04	528.13	542.43	**473.70**	584.86	602.22	623.32	601.34	593.45	608.71	498.51
pcb442	161.77	162.61	162.26	162.73	158.41	214.50	209.68	**147.24**	185.63	179.91	180.30	186.02	177.65	197.41	159.06
d493	110.75	110.98	110.90	111.03	**108.22**	114.16	122.04	112.55	115.05	119.74	121.24	115.65	112.37	124.89	133.81
dsj1000	**404.60**	408.69	**404.60**	406.91	430.94	464.26	398.78	482.85	455.71	429.90	447.28	445.21	462.72	515.66	632.79
Overlap ratio: 0.3															
kroD100	**58.54**	58.69	**58.54**	**58.54**	58.70	**58.54**	**58.54**	62.15	61.45	61.45	61.45	61.45	64.55	78.45	71.32
rat195	45.79	45.91	45.79	45.91	45.82	**45.70**	45.74	48.19	52.11	52.79	51.82	52.97	52.92	58.15	75.04
lin318	772.61	780.56	782.40	782.96	803.37	776.58	**765.96**	946.67	809.06	809.06	809.06	809.06	814.65	1,116.14	1,304.03
rd400	233.29	237.59	233.29	233.75	241.30	**228.95**	229.11	348.99	294.05	300.32	298.04	300.32	239.87	255.27	445.58
pcb442	**85.12**	85.17	**85.12**	85.18	85.41	100.72	106.32	126.21	93.60	93.64	93.60	91.71	88.77	89.82	155.40
d493	70.75	70.84	70.75	70.84	75.38	**69.76**	**69.76**	82.83	80.71	80.09	78.94	82.89	73.19	76.95	148.97
dsj1000	201.92	203.08	201.92	201.95	209.86	204.16	**199.95**	459.22	214.87	214.38	214.87	214.38	219.71	277.26	754.84

TABLE 2. Percentage deviation from the best solution for 15 heuristics on *TSPLIB* problems with equal radius and 2D Euclidean norm.

	HYBRID2	HYBRID1	SZ2	SZ1	CTD	YUAN+	LK-SOCP	GTSP2	TILE2	TILE3	TILE1	TILE4	HULL	MERGE	GTSP1
Overlap ratio: 0.02															
kroD100	0.89	1.15	2.96	2.97	3.93	3.31	3.31	0.00	8.67	7.43	8.80	7.09	32.05	18.24	0.14
rat195	3.60	4.14	7.87	7.91	7.27	7.62	8.64	0.00	18.00	14.61	19.65	18.87	44.36	19.10	0.18
lin318	1.25	1.64	2.56	2.58	2.75	2.87	4.76	0.00	12.21	11.30	11.01	10.05	33.38	13.74	0.27
rd400	0.57	0.89	3.17	3.20	4.61	3.47	3.17	0.00	14.23	12.64	13.98	12.89	46.50	20.34	0.24
pcb442	1.31	1.76	3.16	3.17	5.00	5.44	8.85	0.00	15.01	11.42	11.28	13.50	39.37	24.16	0.37
d493	2.50	3.10	4.29	4.36	3.92	8.16	6.40	0.00	12.76	11.16	13.26	15.66	27.84	21.94	1.01
dsj1000	6.69	8.50	8.02	8.23	6.53	7.49	7.98	0.00	23.60	23.56	21.95	21.83	55.70	35.34	1.64
Average:	2.40	3.03	4.58	4.63	4.86	5.48	6.16	0.00	14.93	13.16	14.28	14.27	39.89	21.84	0.55
Overlap ratio: 0.1															
kroD100	7.49	8.49	7.49	7.95	5.86	6.34	6.34	0.00	13.95	29.68	28.76	27.65	19.91	35.52	1.77
rat195	8.25	9.01	8.64	8.73	9.39	13.77	29.56	0.00	28.54	20.63	23.23	23.72	25.74	33.04	5.85
lin318	5.97	6.57	6.12	6.16	4.27	6.14	0.33	0.00	16.68	25.56	12.25	14.48	28.56	24.67	6.03
rd400	10.18	11.24	10.18	11.19	12.95	11.49	14.51	0.00	23.47	27.13	31.59	26.95	25.28	28.50	5.24
pcb442	9.87	10.44	10.20	10.52	7.59	45.68	42.41	0.00	26.07	22.19	22.45	26.34	20.65	34.07	8.03
d493	2.34	2.55	2.48	2.60	0.00	5.49	12.77	4.00	6.31	10.65	12.03	6.86	3.83	15.41	23.65
dsj1000	1.46	2.49	1.46	2.04	8.06	16.42	0.00	21.08	14.28	7.80	12.16	11.64	16.03	29.31	58.68
Average:	6.51	7.26	6.65	7.03	6.87	15.05	15.13	3.58	18.47	20.52	20.35	19.66	20.00	28.64	15.61
Overlap ratio: 0.3															
kroD100	0.00	0.26	0.00	0.00	0.27	0.00	0.00	6.17	4.97	4.97	4.97	4.97	10.27	34.02	21.83
rat195	0.19	0.46	0.19	0.46	0.26	0.00	0.08	5.44	14.02	15.52	13.39	15.91	15.80	27.24	64.20
lin318	0.87	1.91	2.15	2.22	4.88	1.39	0.00	23.59	5.63	5.63	5.63	5.63	6.36	45.72	70.25
rd400	1.90	3.77	1.90	2.10	5.39	0.00	0.07	52.43	28.43	31.17	30.18	31.17	4.77	11.50	94.62
pcb442	0.00	0.06	0.00	0.07	0.34	18.32	24.91	48.27	9.96	10.01	9.96	7.75	4.28	5.52	82.57
d493	1.42	1.55	1.42	1.55	8.06	0.00	0.00	18.74	15.70	14.81	13.17	18.83	4.91	10.30	113.55
dsj1000	0.99	1.57	0.99	1.00	4.96	2.11	0.00	129.67	7.46	7.22	7.46	7.22	9.88	38.67	277.51
Average:	0.77	1.37	0.95	1.06	3.45	3.12	3.58	40.62	12.31	12.76	12.11	13.07	8.04	24.71	103.50
Overall average:	3.22	3.88	4.06	4.24	5.06	7.88	8.29	14.73	15.24	15.48	15.58	15.67	22.64	25.06	39.89

TABLE 3. Computation time (CPU seconds) for 15 heuristics on *TSPLIB* problems with equal radius and 2D Euclidean norm.

	HYBRID2	HYBRID1	SZ2	SZ1	CTD	YUAN+	LK-SOCP	GTSP2	TILE2	TILE3	TILE1	TILE4	HULL	MERGE	GTSP1
							Overlap ratio: 0.02								
kroD100	0.235	0.570	0.109	0.235	0.328	0.279	0.427	3.158	3.275	0.459	0.938	78.296	78.906	665	666
rat195	0.532	1.453	0.140	0.484	0.594	0.447	1.651	6.645	8.885	0.821	25.265	347.578	344.313	7,780	7,781
lin318	0.508	0.297	0.375	1.547	1.329	0.909	13.923	10.686	26.076	0.991	6.313	357.015	349.063	48,543	48,545
rd400	1.219	0.649	0.453	1.250	1.782	0.659	5.031	15.217	17.232	1.308	61.328	1,254.547	1,225.765	76,007	76,010
pcb442	1.266	1.492	0.657	1.235	1.485	3.223	51.103	20.172	73.710	1.511	52.094	20,289.579	24,625.969	222,043	222,047
d493	0.477	1.211	0.860	1.422	2.688	0.825	1.962	16.344	18.677	1.893	86.796	348.875	322.609	309,067	309,071
dsj1000	1.117	1.040	2.297	3.969	8.718	2.270	10.558	57.731	75.471	2.091	509.515	1,474.328	1,876.625	12,682,539	12,682,545
Average:	0.765	0.959	0.699	1.449	2.418	1.230	12.093	18.564	31.904	1.296	106.036	3,450.031	4,117.607	1,906,663	1,906,666
							Overlap ratio: 0.1								
kroD100	0.133	0.055	0.093	0.109	0.250	0.188	0.204	2.173	2.172	0.640	0.532	6.109	6.438	1,482	1,483
rat195	0.125	0.094	0.109	0.187	0.390	0.204	0.237	4.292	4.247	1.779	25.438	7.812	8.454	5,949	5,951
lin318	0.196	0.063	0.329	0.313	0.797	4.982	5.029	7.242	7.216	2.458	6.578	14.953	15.843	60,249	60,252
rd400	0.289	0.063	0.500	0.407	0.797	0.566	0.830	10.171	10.220	3.021	60.219	26.375	27.531	90,999	91,003
pcb442	0.180	0.063	0.500	0.437	1.203	0.224	0.287	11.664	11.662	3.450	51.891	17.407	20.281	250,482	250,485
d493	0.133	0.110	0.828	0.484	1.844	0.289	0.336	13.581	13.636	21.045	87.406	6.250	8.953	61,711	61,715
dsj1000	0.157	0.070	2.187	0.922	4.640	2.898	2.937	36.875	36.837	35.886	512.094	15.750	31.938	7,807,740	7,807,747
Average:	0.173	0.074	0.649	0.408	1.417	1.336	1.408	12.285	12.284	9.754	106.308	13.522	17.063	1,182,659	1,182,662
							Overlap ratio: 0.3								
kroD100	0.079	0.039	0.078	0.078	0.141	0.184	0.231	1.611	1.693	1.246	0.625	0.391	0.500	865	865
rat195	0.078	0.047	0.125	0.156	0.344	0.234	0.297	3.233	3.358	3.241	25.187	1.016	1.250	6,322	6,323
lin318	0.070	0.039	0.312	0.265	0.625	35.652	35.722	6.255	6.326	9.659	8.109	1.937	3.703	40,164	40,166
rd400	0.078	0.040	0.406	0.375	1.000	2.807	2.877	7.860	7.782	14.015	60.516	4.250	5.547	61,116	61,120
pcb442	0.078	0.055	0.422	0.329	0.750	0.630	0.661	9.650	9.736	24.881	52.093	1.704	3.234	76,133	76,137
d493	0.086	0.047	0.843	0.375	0.687	0.743	0.821	12.086	12.154	43.793	126.485	0.625	1.250	59,105	59,109
dsj1000	0.188	0.032	2.265	1.063	2.766	10.990	11.036	20.177	20.231	313.022	512.032	3.797	12.265	2,623,492	2,623,498
Average:	0.094	0.042	0.636	0.377	0.902	7.320	7.378	8.696	8.754	58.551	112.150	1.960	3.964	409,600	409,603
Overall average:	0.344	0.358	0.661	0.745	1.579	3.295	6.960	13.182	17.647	23.200	108.164	1,155.171	1,379.545	1,166,307	1,166,310

methods (SZ1, SZ2, HYBRID1, HYBRID2), GTSP1, GTSP2, and the two methods from Dong et al. [4] (HULL, MERGE). (2) For low and medium overlap, the methods that use Steiner zones clearly outperform the methods that do not (LK-SOCP and YUAN+). This suggests that for these problems, Steiner zones improve the quality of the solution. (3) The two methods based on the generalized TSP (GTSP1, GTSP2) produce very good, short tour lengths for low and medium overlap problems and poor tour lengths for high overlap problems. The computation times are very long. (4) The hybrid methods (HYBRID1, HYBRID2) produce results that fall between the Steiner zone methods and the GTSP methods. For low overlap problems, the hybrid methods find much better tours than the Steiner zone methods but not for the medium and high overlap problems. They are much slower than the Steiner zone methods and much faster than the GTSP methods. (5) The two methods that do not use Steiner zones (YUAN+ and LK-SOCP) perform better on most high overlap ratio problems than all Steiner-zone methods. Because LK-SOCP is so fast, it may be worthwhile to incorporate LK-SOCP into SZH as a hybrid heuristic. (6) If we consider the order in which each heuristic appears in Tables 2 and 3 to be performance rankings, then SZ1 and SZ2 have the best mean ranking, 4, of all 15 heuristics tested. LK-SOCP is next with an average rank of 5. All other methods, including those from the literature, have average rank greater than 6.

In separate experiments, we have shown that each phase of our Steiner-zone heuristic is vital to the overall accuracy of the heuristic. Due to space limitations, we do not present these results here.

6. Conclusions

This paper has presented a new, straightforward heuristic for the solving the CETSP. The Steiner-zone heuristic reduces a graph of nodes to a set of Steiner zones, assigns a good tour sequence to them, and then minimizes the tour's length with respect to the Steiner zones. We compared 15 different heuristics, including seven from the literature, for solving the CETSP on 21 test problems. We found that the combination of low computation time and high solution quality made the Steiner-zone heuristics very competitive methods.

Our future work will apply our heuristics to problems with an arbitrary radius for each node. We will also solve problems with 3D Euclidean and 3D Manhattan norms.

Acknowledgment

We thank Scott Nestler, Kiran Panchamgam, Matthew Reindorp, Yufeng Tu, Wenli Wang, and the anonymous referees for their comments and suggestions which have greatly improved our paper. We thank Damon Gulzcynski and Jing Dong for providing codes used in our experiments.

Appendix

A. The IP_{PhIII} Heuristic

IP_{PhIII} solves a succession of shortest path problems on 3-point approximations of the Steiner zones produced in Phase I using the fixed sequence of Steiner zones produced in Phase II. As long as the Steiner zones are sufficiently disjoint, the algorithm converges to a near-optimal solution of the Touring Steiner Zones Problem (TSZP). Though Phase I does not guarantee disjoint Steiner zones, we have not observed a single case in which the produced Steiner zones had too much overlap to affect the convergence of IP_{PhIII}. In Figures A.1 through A.9 we illustrate eight iterations of IP_{PhIII} for a 3-node problem. For simplicity of illustration, all nodes are singleton Steiner zones.

FIGURE A.1. Layout and first iteration of IP_{PhIII} (initialization).

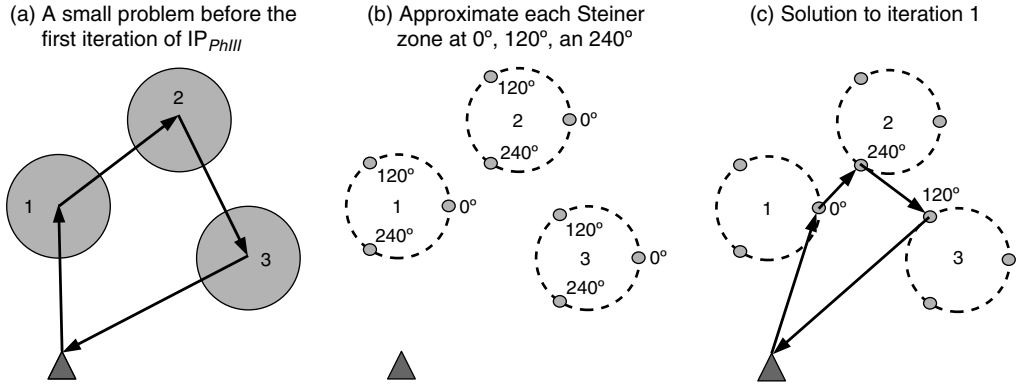
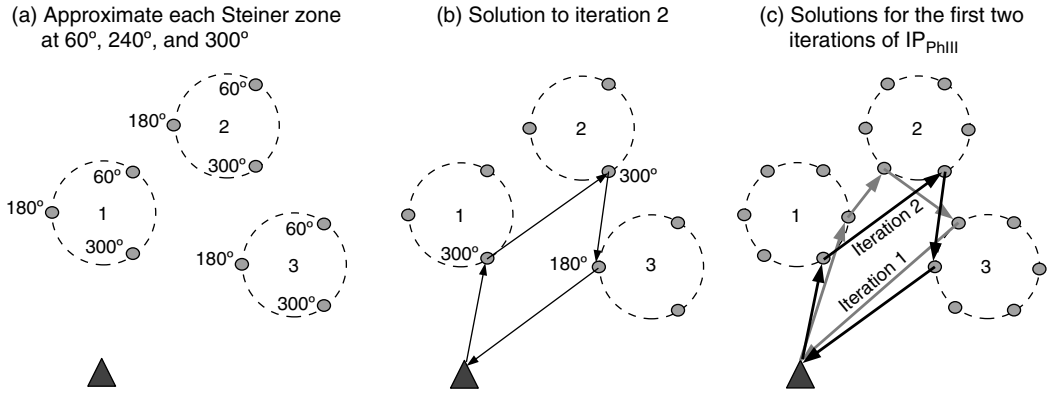


FIGURE A.2. Second iteration of IP_{PhIII} (initialization).



Note. Combining the first two solutions illustrates the bounded regions in which the optimal representative points are likely to lie.

FIGURE A.3. Iteration 3.

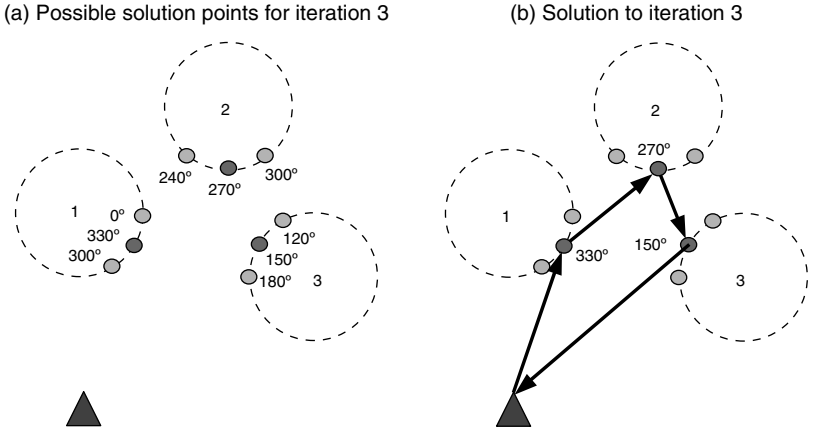


FIGURE A.4. Iteration 4.

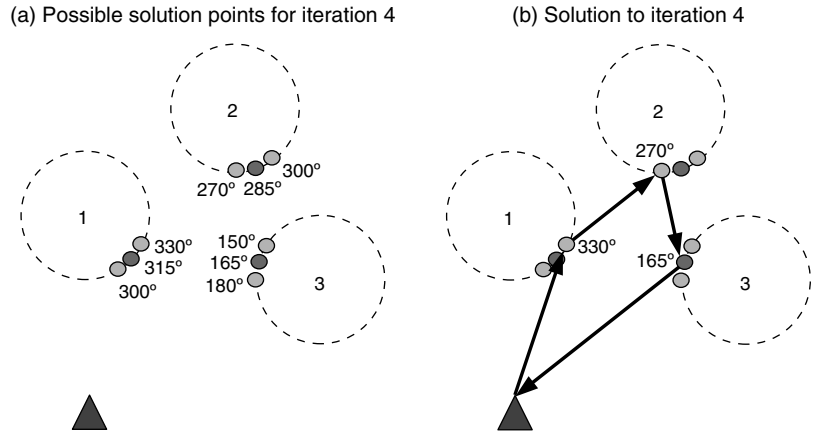


FIGURE A.5. Iteration 5.

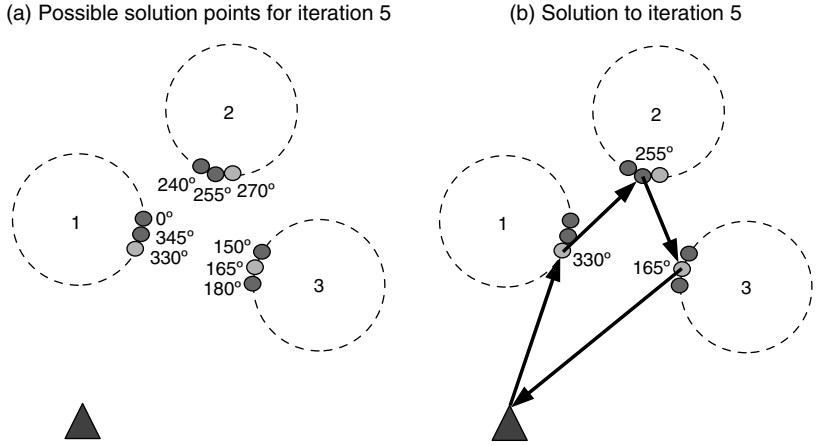


FIGURE A.6. Iteration 6.

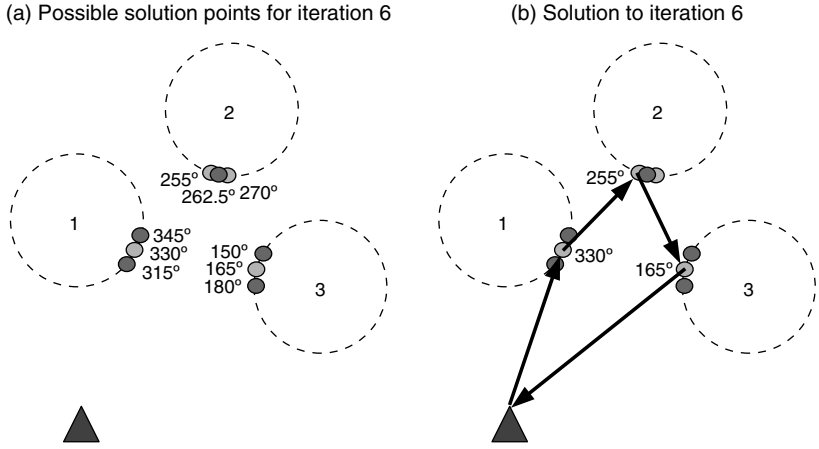


FIGURE A.7. Iteration 7.

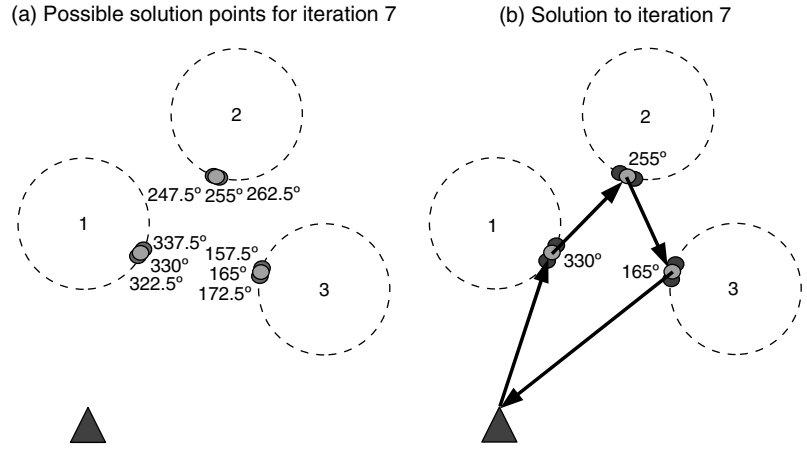


FIGURE A.8. Iteration 8.

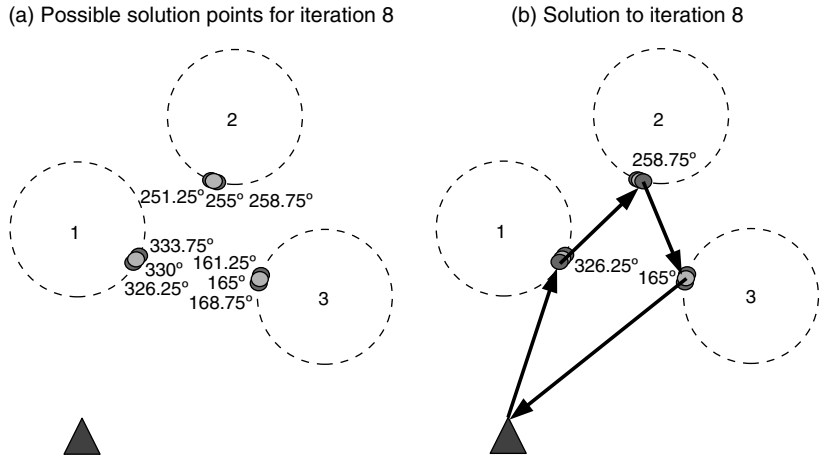
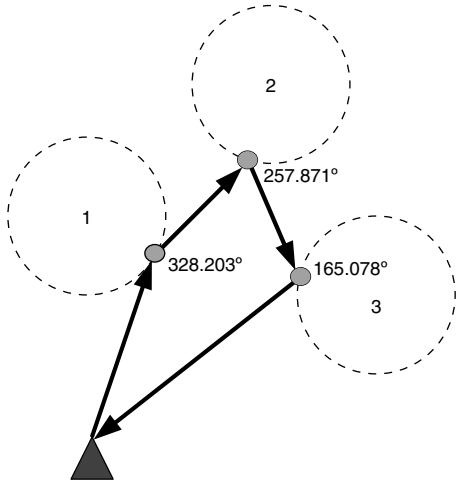


FIGURE A.9. The final result (iteration 30).



References

[1] D. Applegate, R. Bixby, V. Chvátal, and W. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, NJ, 2006.

[2] E. Arkin and R. Hassin. Approximation algorithms for the geometric covering salesman problem. *Discrete Applied Mathematics* 55:197–218, 1994.

[3] Concorde TSP Solver. http://www.tsp.gatech.edu/concorde/index.html.

[4] J. Dong, N. Yang, and M. Chen. Heuristic approaches for a TSP variant: The automatic meter reading shortest tour problem. E. Baker, A. Joseph, A. Mehrotra, and M. Trick, eds. *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*. Springer, New York, 145–163, 2007.

[5] M. Dror, A. Lubiw, A. Efrat, and J. Mitchell. Touring a sequence of polygons. *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, ACM Press, New York, 473–482, 2003.

[6] A. Dumitrescu and J. Mitchell. Approximation algorithms for TSP with neighborhoods in the plane. *Journal of Algorithms* 45:135–159, 2003.

[7] J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics* 17:449–467, 1965.

[8] K. Elbassioni, A. Fishkin, and R. Sitters. On approximating the TSP with intersecting neighborhoods. *Proceedings of the 17th International Symposium on Algorithms and Computation*, New York, 213–222, 2006.

[9] M. Fischetti, J. Gonzalez, and P. Toth. A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research* 45:378–394, 1997.

[10] M. Gendreau, G. Laporte, and F. Semet. The covering tour problem. *Operations Research* 45:568–576, 1997.

[11] D. Gulczynski, J. Heath, and C. Price. Close enough traveling salesman problem: A discussion of several heuristics. F. Alt, M. Fu, and B. Golden, eds. *Perspectives in Operations Research: Papers in Honor of Saul Gass' 80th Birthday (Operations Research/Computer Science Interfaces Series)*. Springer, New York, 271–283, 2006.

[12] M. Held and R. Karp. The traveling salesman problem and minimum spanning trees. *Operations Research* 18:1138–1162, 1970.

[13] S. Lin and B. Kernighan. An effective heuristic algorithm for the travelling salesman problem. *Operations Research* 21:2245–2269, 1973.

[14] W. Mennell. Heuristics for solving three routing problems: Close-enough traveling salesman problem, close-enough vehicle routing problem, sequence-dependent team orienteering problem. Ph.D. thesis, University of Maryland, College Park, 2009.

[15] J. Mitchell. A PTAS for TSP with neighborhoods among fat regions in the plane. *Proceedings of the 18th Annual ACM–SIAM Symposium on Discrete Algorithms*, SIAM, Philadelphia, PA, 11–18, 2007.

[16] K. Mulmuley. A fast planar partition algorithm, part I. *Journal of Symbolic Computation* 10:253–280, 1990.

[17] Reinelt, G. TSPLIB. Ruprecht Karls Universität Heidelberg, http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95, 2006.

[18] R. Shuttleworth, B. Golden, S. Smith, and E. Wasil. Advances in meter reading: Heuristic solution of the close enough traveling salesman problem over a street network. B. Golden, S. Raghavan, and E. Wasil, eds. *The Vehicle Routing Problem: Latest Advances and New Challenges*, Vol. 43. Springer, New York, 487–501, 2008.

[19] J. Silberholz and B. Golden. The generalized traveling salesman problem: A new genetic algorithm approach. E. Baker, A. Joseph, A. Mehrotra, and M. Trick, eds. *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*. Springer, New York, 165–181, 2007.

[20] B. Yuan, M. Orlowska, and S. Sadiz. On the optimal robot routing problem in wireless sensor networks. *IEEE Transactions on Knowledge and Data Engineering* 19:1251–1261, 2007.