
Close Enough Traveling Salesman Problem: A Discussion of Several Heuristics

Damon J. Gulczynski, Jeffrey W. Heath, Carter C. Price

Applied Mathematics Program
Department of Mathematics
University of Maryland
College Park, MD 20742
damon@math.umd.edu, jheath@math.umd.edu, price@math.umd.edu

Summary. The use of radio frequency identification (RFID) allows utility companies to read meters from a distance. Thus a meter reader need not visit every customer on his route, but only get within a certain radius of each customer. In finding an optimal route – one that minimizes the distance the meter reader travels while servicing each customer on his route – this notion of only needing to be close enough changes the meter reading problem from a standard Traveling Salesperson Problem (TSP) into a variant problem: Close Enough TSP (CETSP). As a project for a graduate course in network optimization various heuristics for finding near optimal CETSP solutions were developed by six groups of students. In this paper we survey the heuristics and provide results for a diverse set of sample cases.

Key words: Traveling Salesman Problem; Radio Frequency Identification; Electronic Meter reading.

1 Introduction

Historically when a utility company measures the monthly usage of a customer, a meter reader visits each customer and physically reads the usage value at each site. Radio frequency identification (RFID) tags at customer locations can remotely provide data if the tag reader is within a certain radius of the tag. This changes the routing problem from one of a standard traveling salesman problem (TSP) to what we call a “Close Enough” TSP (CETSP). Thus the route lengths of the meter readers can be drastically reduced by developing heuristics that exploit this “close enough” feature.

We consider such a meter reading routing problem where each customer is modeled as a point in the plane. Additionally there is a point that represents the depot for the meter reader. A CETSP tour must begin and end at the depot and travel within the required radius, r , of each customer. For simplicity in the cases tested here the meter

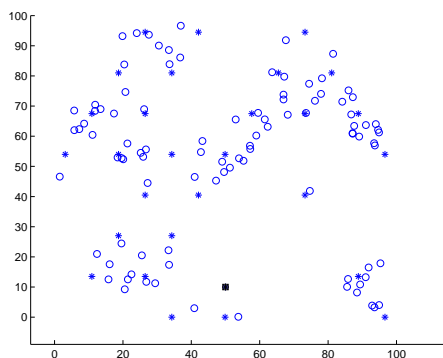


Fig. 1. An example of a supernode set on 100 nodes, with radius 9, and the depot located at (50,10). The circles represent the customer nodes, and the asterisks are the supernodes.

reader was not restricted to a road network. All distances are Euclidean and the objective is to minimize the total distance traveled. The solution to a standard TSP on the customer nodes obviously provides an upper bound for the CETSP.

Essentially the CETSP is a TSP with a spatial window. Thus it is conceptually similar to the TSP with a time window. Several heuristics are discussed in the work of Solomon [9]. In spite of the similarities the heuristics for the TSP problem with a time window do not directly apply to the CETSP. This is because we do not simply change the order of the points visited, we actually change the location of the points themselves. So the CETSP has an uncountable solution space.

As a class project six teams developed heuristics to produce solutions to this problem. This paper highlights the innovative developments from these projects. In Section two we discuss the proposed heuristics. In the third Section we present the numerical results across a test bed of cases. We conclude with some suggestions for further work in Section four. Pseudo code for some of the heuristics is provided in the appendix.

2 Heuristics

All of the heuristics developed have three distinct steps. Given an initial set C of customer nodes (c-nodes), the first step is to produce a *feasible supernode* set, S . A feasible supernode set is a set of points (containing the depot node) with the property that each customer node is within r units of at least one point in the set. In figure 1 the set of asterisks (*) represents a feasible supernode set since each customer (circle) is within r units ($r = 9$) of at least one asterisk. After producing S the second step is to find a near optimal TSP tour, T , on the points in S , as seen in figure 2. Since each

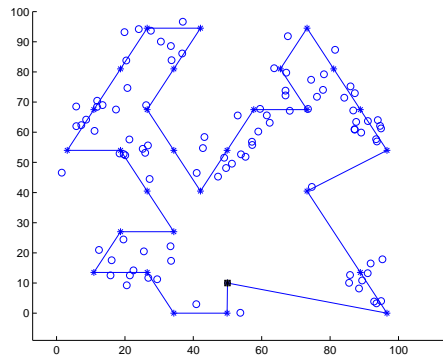


Fig. 2. An example CETSP tour on 100 nodes, with radius 9, and the depot located at (50,10). The circles represent the customer nodes, and the asterisks are the supernodes.

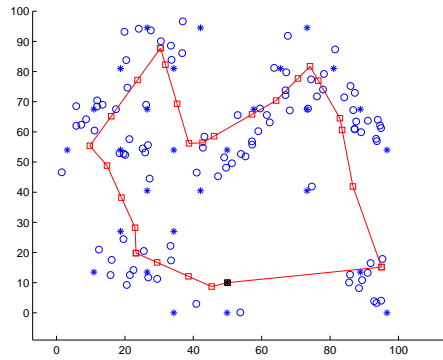


Fig. 3. The results of an economization of the tour in figure 2.

customer node is within r units of a supernode, we are guaranteed that in traversing T we will pass within the required radius of each customer. We call T a feasible CETSP tour. Since the cardinality of the feasible supernode set is smaller than the number of customers, sometimes significantly, it is more efficient to generate the tour on S . Thus performing step 1 prior to step 2 requires much less computational time than starting by generating the TSP on C . The final step is an economization routine that reduces the distance traveled in T while maintaining feasibility, thus generating a shorter CETSP tour T' . The results of this economization can be seen in figure 3.

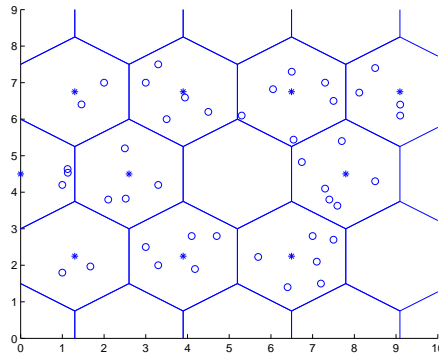


Fig. 4. An example of tiling the plane with regular hexagons of side length $r = 1.5$ units. We ensure that all customers (represented as small circles) in a given tile are within r units of the center of that tile (*).

2.1 Producing a Feasible Supernode Set

Four heuristics with variations were developed for producing a feasible supernode set: tiling with three variations, Steiner zone, sweeping circle, and radial adjacency. Each of these techniques is based on the assumption that it is desirable to have as few supernodes as possible. Examples can be constructed in which the fewest number of supernodes does not result in the shortest CETSP tour; however, empirical tests show that in general reducing the number supernodes is a good strategy.

Tiling methods

For the tiling methods of producing a feasible supernode set, the plane is tiled with polygons that can be inscribed in a circle of radius r . This ensures that each c -node in the polygonal tile is within a distance r of its center. Thus by letting the supernodes be the centers of the tiles which contain at least one c -node we create a feasible supernode set. In implementation regular hexagons with side length r were chosen because over all regular polygons because they minimize the area of overlap between adjacent supernodes (See figure 4). Once a feasible supernode set S has been constructed there are several techniques to reduce the cardinality of S , including shifting, merging, and circular extension.

Shifting

One can translate or shift the tile centers in an attempt to reduce the total number of supernodes. The translation procedure works by performing a series of small shifts (vertically and/or horizontally) on the centers of the tiles. This process creates a collection of feasible supernode sets and from this collection we choose the set with the

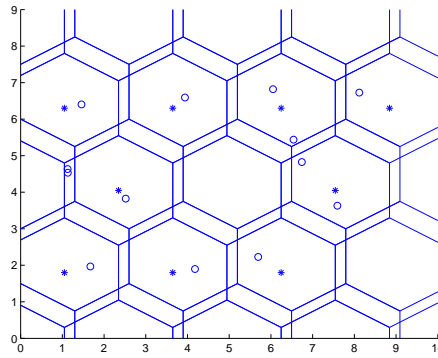


Fig. 5. The tiles from figure 4 are translated as shown to reduce the total number of supernodes from ten to nine.

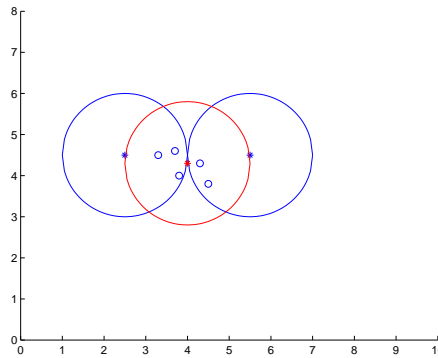


Fig. 6. Two supernodes are merged into one.

smallest cardinality (see figure 5) [5].

Merging

Merging works by considering the c -nodes in two adjacent tiles and determining if they can be covered with one circle. In this way it might be possible to merge two adjacent supernodes into one (see figure 6). Specifically we take the maximal and minimal x and y values of the c -nodes in the two adjacent tiles: x_{min} , x_{max} , y_{min} and y_{max} . The midpoint of the extreme x 's and y 's, $(\frac{x_{min}+x_{max}}{2}, \frac{y_{min}+y_{max}}{2})$, is then considered as a potential supernode. If all the constituent c -nodes of the two tiles are within r units of this midpoint the merger is accepted, and the number of supernodes is reduced by one. This process continues until no mergers are possible [2].

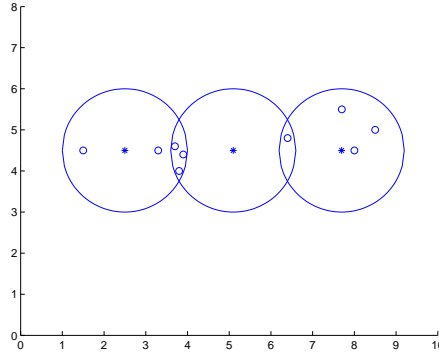


Fig. 7. By applying circular extension the middle supernode can be omitted.

Circular Extension

Given the supernode set, S , obtained from the centers of hexagonal tiles it might be possible to eliminate some elements of S by considering the intersection of their corresponding circumscribed circles. Let us define the *degree* of a supernode as the number of c -nodes that lie within r units of this supernode, and do not lie within r units of any other center. By associating c -nodes that lie in an overlap (i.e. an intersection of two circles) with the center of the largest degree it might be possible to eliminate superfluous centers reducing the size of the feasible supernode set (see figure 7) [3].

Steiner Zone

From the nodes' perspective, the meter reader can visit any point within r units of the node. Thus there is a circle of radius r around the node, representing the node's service region, through which the meter reader must pass. If two c -nodes are less than $2r$ units apart, then their corresponding service region circles will overlap. Thus any node in this overlap region represents a candidate supernode covering these two nodes. We can minimize the cardinality of a supernode set S by choosing points that lie within the intersection of multiple circles. The Steiner zone method consists of finding these intersections first, and then choosing the supernode set in this manner.

Let $D(c_1, r), D(c_2, r), \dots, D(c_k, r)$ be discs of radius r centered at c -nodes c_1, c_2, \dots, c_k respectively, then $D(c_1, r) \cap D(c_2, r) \cap \dots \cap D(c_k, r)$ is the set of points within r units of each of c_1, c_2, \dots, c_k . If this intersection is not empty we call it the *Steiner zone* of c_1, c_2, \dots, c_k and denote it by $Z(c_1, c_2, \dots, c_k)$. Furthermore we say that $Z(c_1, c_2, \dots, c_k)$ covers c_1, c_2, \dots, c_k , and k is called the *degree* of the Steiner zone (see figure 8). Any point in the Steiner zone $Z(c_1, c_2, \dots, c_k)$ can be a supernode covering the c -nodes

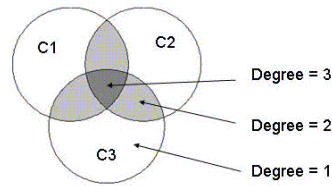


Fig. 8. The steiner zones of a three node group are displayed here.

c_1, c_2, \dots, c_k .

Since our goal is to minimize the number of supernodes, it is advantageous to find Steiner zones of large degree. Ideally we would like to enumerate every Steiner zone and greedily choose those of largest degree. In practice this is unreasonable as the number of distinct Steiner zones on n c -nodes could be as large as 2^n . However, in order to quickly obtain a feasible supernode set yielding a good CETSP tour we need not consider all Steiner zones. Empirical tests have shown that it is sufficient to consider Steiner zones less than some fixed degree, and from those Steiner zones build zones of higher order. For our example results in Section 3 the fixed degree was four. Once all c -nodes are covered by at least one Steiner zone a feasible supernode set is obtained by choosing an arbitrary point in each respective zone. Pseudocode can be found in the Appendix [8].

Sweeping Circle

The sweeping circle heuristic covers the plane with overlapping circles with centers offset by $d_{min} = \min(\sqrt{2}r, \min(\{d_{ij}\}))$, where d_{ij} is the Euclidean distance between c -nodes i and j (see figure 9). This heuristic is greedy because it chooses the center of the circle containing the most c -nodes and adds it to the set of supernodes S in each iteration (See figure 10). All c -nodes within this circle are now covered. The sweeping and selection steps are repeated until all c -nodes in C are covered, at which point S is a feasible supernode set. A more detailed description of this procedure is provided in the Appendix [7].

There is a wealth of literature concerning covering a set of points on the plane with circles. For an efficient algorithm refer to Gonzalez [4].

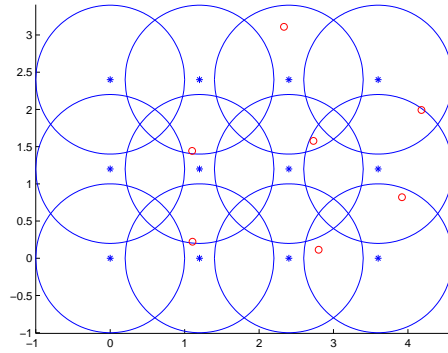


Fig. 9. Graphical depiction of initial sweeping circles.

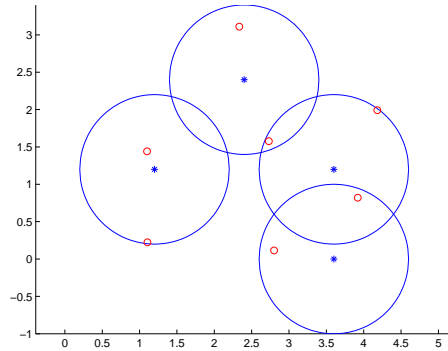


Fig. 10. Those circles containing the most c-nodes are selected until all c-nodes are covered.

Radial Adjacency

First a matrix is created containing the distances between each pair of c -nodes. Two c -nodes are said to be adjacent if the distance between them is at most r . An adjacency matrix A is constructed on the c -nodes, where entry (i, j) in A is 1 if the c -nodes i and j are adjacent, and 0 otherwise (by convention A is 1 along the main diagonal). We define the *degree* of a c -node i as the sum of the entries in row i of A , i.e., it is the number of nodes to which i is adjacent.

The supernode set S is created by an iterative method where at each step we consider the c -node k with the highest degree. The geometric mean of k and all vertices adjacent to k is then computed. If this geometric mean is adjacent to more c -nodes than k , then it is designated as a supernode at this step. Otherwise, k is designated as a supernode (see figure 11). A more detailed description of this procedure is provided

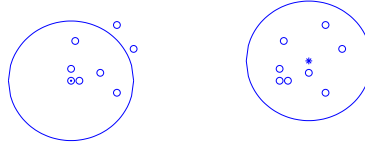


Fig. 11. In this case the geometric mean (* in right figure) of the points adjacent to k has a higher degree than k (the solid circle in left figure).

in the Appendix.

In place of the greedy selection step, the following variation of this iteration was used for sparse data sets in the hope of minimizing the large travel distances needed to reach remote c -nodes. First select the c -node with the smallest degree. From all the c -nodes adjacent to the selected node, choose the one with the highest degree, k . Add k to S and remove k and all c -nodes adjacent to k from consideration. The heuristic ends when all nodes are considered [1].

2.2 TSP solver

Once a feasible supernode set is produced the second step is to find a TSP tour, T , on the supernodes. There is a wealth of literature on this subject and we will not go into the topic here [6]. In practice the groups used a variety of TSP heuristics resulting in near-optimal tours. Thus we expect only minimal variation in the tour lengths due to the software selection.

2.3 Economization

The third and final step of the heuristics is the economization algorithm. This algorithm is based on minimizing the marginal cost of visiting a given node in the tour T . We first enumerate the supernodes on the tour in the order they are visited. If node i were not in T , the shortest path from node $i - 1$ to node $i + 1$ would be a straight line between the two. So we determine the line between $i - 1$ and $i + 1$ and move node i as close to this line as possible while still maintaining feasibility.

Problem	Data Type	c -nodes	TSP length	Radius	Method	$\ell(T)$
1	clustered	100	655.09	9	Shifted Tiling	344.89
2	random	200	1154.06	20	Merging Tiling	288.16
3	clustered	300	1120.49	7	Merging Tiling	537.17
4	random	400	1533.95	5	Merging Tiling	797.04
5	clustered	500	1189.40	2	Merging Tiling	798.60
6	random	500	1627.91	27	Shifted Tiling	246.08
7	clustered	1000	2231.40	12	Steiner Zone	461.82

Table 1. Shortest tour lengths $\ell(T)$ generated by heuristics discussed.

Problem	Data Type	c -nodes	Radius	Method	<i>supernodes</i>
1	clustered	100	9	Steiner Zone	18
2	random	200	20	Steiner Zone	11
3	clustered	300	7	Steiner Zone	38
4	random	400	5	Steiner Zone	18
5	clustered	500	2	Steiner Zone	147
6	random	500	27	Steiner Zone	8
7	clustered	1000	12	Merging Tiling	30

Table 2. Fewest supernodes generated by heuristics discussed.

We repeat this routine for each node in T except for the depot node which must remain fixed. The resulting tour will have a total distance traveled no greater than the length of T . By repeating this technique on each new tour, a sequence of tours is created: T_1, T_2, T_3, \dots each one of length no greater than the one before it. The process terminates when the improvement between two tours drops below a specified tolerance. The resulting tour is our CETSP solution T' [5].

3 Numerical Results

These heuristics were tested on a diverse set of test cases. The distances for each method were compared and the best tour lengths are reported below along with the method that produced the best result. Also reported are the methods that resulted in the fewest supernodes for each problem. Some the data sets have randomly generated c -nodes while others have clusters of c -nodes. The TSP length represents the total distance of a near optimal solution of the TSP on the set of c -nodes. The data sets have a variety of values for the radius to provide a representative sample for real world applications. All of this was done in a 100 by 100 grid.

Problem	Steiner Zone	Sweep	Radial Adjacency	Circular extension	Merging	Shifting
1	375.56	378.65	512.19	410.56	377.87	344.89
2	288.39	300.83	448.33	342.34	288.16	327.31
3	560.26	562.64	758.95	651.01	537.17	597.94
4	838.72	849.42	1154.70	1141.80	797.04	870.68
5	819.90	1014.98	1040.50	1662.50	798.60	827.76
6	278.20	246.79	373.18	304.38	279.06	246.08
7	461.82	468.88	628.16	504.44	468.54	484.42

Table 3. Fewest supernodes generated by heuristics discussed.

4 Conclusion

These heuristics produce CETSP tours in the test bed of cases that have significantly shorter length than TSP tours on the c -nodes. The hexagonal tiling heuristics were the most successful, particularly if an extension such as the shifting or merging heuristics is used. The Steiner zone method also proved to be quite effective. While the methods that result in fewer supernodes generally have the shorter final tour lengths, the method with the fewest supernodes does not always produce the shortest tour. The Steiner zone method produced the fewest supernodes in most cases but the tiling heuristics generally produced the shortest tour.

Clearly the use of RFID technology can reduce the travel distance of meter readers substantially, though further work is required to port this method from Euclidean space to more realistic road networks.

Acknowledgments

This work would not have been possible without Dr. Bruce Golden's ability to develop such interesting problems. Furthermore we would like to thank all of the students in Dr. Golden's BMGT831 course for their cooperation and insight. The credit for so aptly naming the Close Enough Traveling Salesman Problem belongs to Damon's father.

References

1. B. Davis, T. Retchless and N. Vakili. Radio-frequency adoption for home meter reading and vehicle routing implementation. Final project for BMGT831 R. H. Smith School of Business. University of Maryland. Spring 2005.
2. J. Dong, N. Yang and M. Chen. A heuristic algorithm for a TSP variant: Meter reading shortest tour problem. Final project for BMGT831 R. H. Smith School of Business. University of Maryland. Spring 2005.

3. T. Gala, B. Subramanian and C. Wu. Meter Reading Using Radio Frequency Identification. Final Project for BMGT831 R. H. Smith School of Business. University of Maryland. Spring 2005.
4. T. Gonzalez. Covering a Set of Points in Multidimensional Space. *Proceedings of the Twenty-eighth Annual Allerton Conference on Communications, Control, and Computing*, Oct 1990.
5. D.J. Gulczynski, J.W. Heath and C.C. Price. Close Enough Traveling Salesman Problem: Close only counts in horseshoes, and hand grenades ... and meter reading? Final Project for BMGT831 R. H. Smith School of Business. University of Maryland. Spring 2005.
6. G. Gutin and A.P. Punnen, *The Traveling Salesman Problem and Its Variations*. Springer, 2002.
7. S. Hamdar, A. Afshar and G. Akar. Sweeping Circle Heuristic. Final Project for BMGT831 R. H. Smith School of Business. University of Maryland. Spring 2005.
8. W. Mennell, S. Nestler, K. Panchangam and M. Reindorp. Radio Frequency Identification Vehicle Routing Problem. Final Project for BMGT831 R. H. Smith School of Business. University of Maryland. Spring 2005.
9. M. Solomon. Algorithms for the vehicle routing and scheduling problem with time window constraints. *Operations Research*, 35:254-265, 1987.

A Appendix: Psuedocode for Iterative Methods

Heuristic: Steiner Zone

1. Enumerate the c -nodes c_1, c_2, \dots, c_n .
2. $C' := \{c_1, \dots, c_n\}$ (c -nodes)
3. $I := \{1, \dots, n\}$ (index set)
4. $S := \{depot\}$ (supernode set)
5. **repeat**
6. $k := \min\{i \in I\}$
7. $ZLIST := \emptyset$ (list of Steiner zones covering c_k)
8. **for** all $j \in I$ with $j > k$
9. **if** $D(c_k, r) \cap D(c_j, r) = \emptyset$ (only non-empty intersections are considered)
10. **if** $D(c_j, r) \cap Z = \emptyset$ for all $Z \in ZLIST$ (no bigger Steiner zones covering c_k and c_j can be built)
11. Add $Z(c_k, c_j)$ to $ZLIST$
12. **else** (bigger zones can be built)
13. Let $Z := Z(c_k, c_{i(1)}, \dots, c_{i(r-1)})$ be the Steiner zone of largest degree in $ZLIST$ such that $Z(c_k, c_j) \cap Z \neq \emptyset$
14. Add $Z(c_k, c_{i(1)}, \dots, c_{i(r-1)}, c_j)$ to $ZLIST$ (a zone of degree $r + 1$ is built)
15. **if** $r \leq 4$ (for reasonable runtime we only store and subsequently build off zones of at most degree four)
16. Add all sub-Steiner zones of $Z(c_k, c_{i(1)}, \dots, c_{i(r-1)}, c_j)$ containing c_k and c_j to $ZLIST$ (i.e. if $r = 3$, $Z(c_k, c_{i(1)}, c_{i(2)}, c_j)$ then $Z(c_k, c_j), Z(c_k, c_{i(1)}, c_j)$ and $Z(c_k, c_{i(2)}, c_j)$ are all added; this is the seedbed of small degree zones from which zones of larger degree are built)
17. **end-if**

18. **end-if-else**
19. **end-if**
20. **end-for**
21. Let $Z := Z(c_k, c_{i(1)}, \dots, c_{i(m-1)})$ be the zone of largest degree in $ZLIST$
22. Add a point $z \in Z$ to S
23. Remove $c_k, c_{i(1)}, \dots, c_{i(m-1)}$ from C'
24. Remove $k, i(1), \dots, i(m-1)$ from I
25. **until** $C' = \emptyset$.

Heuristic: Sweeping Circle

1. Generate distance matrix, $D = [d_{ij}]$, on c -nodes
2. Initialize $C' := C, S := \{depot\}$.
3. **repeat**
4. Calculate $d_{min} = \min(\sqrt{2}r, \min(\{d_{ij}\}))$.
5. Cover the plane with circles of radius r translated vertically and horizontally by integer multiples of d_{min} .
6. Set $P :=$ centers of the circles.
7. Assign each node in P a *degree* equal to the number of c -nodes within radius r of that node.
8. Add p to S where p is the node in P of highest degree.
9. Remove from C' the c -nodes within r of p .
10. Update the degrees of P by considering only those c -nodes in C' .
11. **until** $C' = \emptyset$.

Heuristic: Radial Adjacency

1. Construct the adjacency matrix on the c -nodes.
2. Calculate the degree of each c -node.
3. Set $C' := C, S := \{depot\}$.
4. **repeat**
5. Select $k \in C'$ with the largest degree.
6. Calculate k^* , the geometric mean of k and all c -nodes adjacent to k .
7. **If** $\text{degree}(k^*) \geq \text{degree}(k)$
8. Add k^* to S and remove all vertices adjacent to k^* from C' .
9. **Else**
10. Add k to S and remove k and all vertices adjacent to k from C' .
11. **End If-Else**
12. **until** $C' = \emptyset$.