A 2-opt-based Heuristic for the Hierarchical Traveling Salesman Problem

Eric Kuang

May 2012

1 Introduction

The traveling salesman problem (TSP) is a well-known routing problem that, when given a set of locations, involves finding the least-cost route that visits each location exactly once. However, the TSP views each location only in terms of its coordinates; it does not take into account the relative importance of visiting certain locations early due to their priorities. The incorporation of location priorities may better model the realistic constraints of some routing problems.

For example, the salesman may wish to visit the most populous locations first, which would help him generate sales more quickly. Panchamgam et al. called this type of prioritized TSP the hierarchical traveling salesman problem (HTSP), and applied it to modeling humanitarian relief routing. In the event of a natural disaster or epidemic, emergency supplies such as food and medicine should be delivered earlier to locations that are more populous or seriously affected.

The HTSP begins with a set of locations, each with its own coordinates on a bounded plane and a priority. As in the TSP, the objective of the HTSP is to find the least-cost route that visits each of the locations exactly once. We often refer to a location as a "node" and a set of locations as "nodes." However, the priorities introduce constraints regarding the order in which the locations may be visited, which can be strengthened or relaxed depending on the situation. In an HTSP, a lower priority number corresponds to a higher priority, with priority 1 being the highest priority. Panchamgam et al. introduced a constraint in terms of an integer d, which ranges from 0 to $h-1$, where h is the number of priority classes. The value of this "HTSP constraint" d stipulates the permitted priorities that can be serviced by the delivery vehicle at each point in its route. At any point in the route, the delivery vehicle may only visit the priorities from p to $p+d$, where p is the highest priority that contains at least one unvisited location. Hence, when $d = h-1$, the HTSP becomes a classical TSP from the outset; the vehicle can visit any node since $[p, p+d]$ will contain all priorities.

Take the example of an HTSP with a constraint of $d = 2$ and a set of locations that consists of priorities 1 through 6 with at least one node of each priority. At the beginning of the route, $p = 1$ because there is at least one unvisited node of priority 1, which is the highest priority in our problem. As a result, we can visit any of the priority 1, 2, and 3 nodes at the outset because they are contained in the interval $[p, p+d]$. After all the priority 1 nodes have been visited, the delivery vehicle can begin to service priority 4 nodes. That is, we next visit nodes of priority 2, 3, or 4.

Like the TSP, the HTSP has no known polynomial time solution. As a result, we want to find a reasonable heuristic that runs in a low-order polynomial time and can find good solutions to large HTSP problem instances in a reasonable amount of time. The HTSP heuristic should generate a tour that starts at a depot and visits all nodes exactly once before returning to the depot, while minimizing the distance traveled subject to the priority constraint. The goal is to create a heuristic that creates a near-optimal solution, while minimizing the time taken to produce the tour.

2 Existing TSP Heuristics

Although the optimal routes for a TSP and an HTSP with $d = 0$ over the same set of locations usually differs dramatically, the TSP and HTSP are related problems in terms of their objectives. As a result, TSP heuristics could be useful as a basis for an HTSP heuristic.

In particular, we look at the 2-opt procedure, which is an $O(n^3)$ procedure based on the exchange or swap of a pair of edges, where n is the number of locations. Figure 1 shows an example of a potential tour that visits all nodes exactly once and forms a cycle.

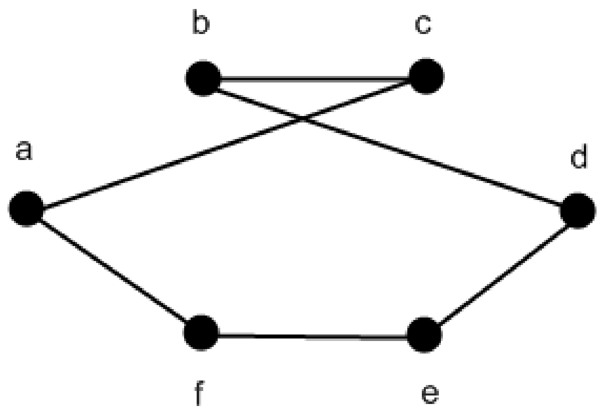

Figure 1: Sample tour with six locations. (a,c) and (b,d) will be swapped

To swap two edges (a,c) and (b,d), the nodes of the pair of edges are rearranged either as (a,b) and (c,d), or (a,d) and (b,c). Only one of these possible rearrangements will maintain a single route as seen in Figure 2(a), while the other will split the route into two disconnected sub-tours, as seen in Figure 2(b). Hence, only the swap that maintains a single route is considered a legal 2-opt swap.

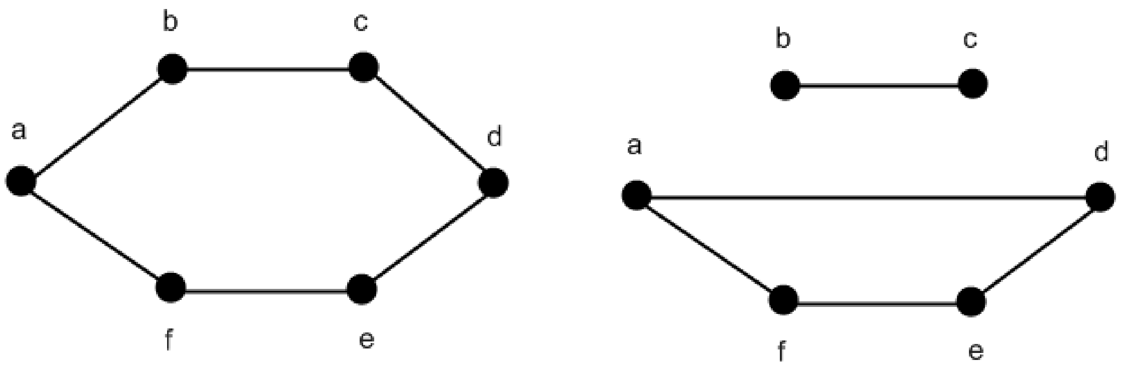

Figure 2: (a) Swap to (a,d) and (b,c) creates a legal tour (b) Swap to (a,b) and (c,d) creates two disconnected tours

As shown in Figure 2(a), 2-opt swaps can modify the tour in a way that shortens the total length of the tour. The 2-opt heuristic repeatedly applies 2-opt swaps by systematically testing all possible pairs of edges in the tour. For each swap, if the resultant tour has a length that is longer than or equal to the length of the tour before the swap, then we undo the swap. Otherwise, the swap must have resulted in a shorter tour. In this case, the swap is preserved. The heuristic then proceeds to swap the next pair of edges. For example, if a tour begins as Figure 2, then a 2-opt swap of (a,b) and (c,d) to (a,c) and (b,d) that results in the tour in Figure 1 would be not be preserved by the 2-opt procedure because it creates a longer tour. However, if the tour begins as Figure 1, then a 2-opt swap of (a,c) and (b,d) to (a,b) and (c,d) that brings the tour to Figure 2(a) would be preserved because the swap results in a shorter tour.

The procedure stops when it reaches a tour that cannot be improved with 2-opt swaps. If we assume Euclidean distances, Figure 2(a) is an example of a tour that cannot be improved with 2-opt swaps, because none of the 15 possible 2-opt swaps (since there are $\binom{6}{2} = 15$ possible pairs of edges) will improve the tour. Hence, once the 2-opt heuristic has tested all 15 possible 2-opt swaps on the tour, the procedure will stop and return the tour.

3 Approach

Our heuristic for the HTSP is based on the 2-opt heuristic for solving a classic TSP. However, we must enforce the priority constraints included in the HTSP. We do so by first finding a feasible solution under this constraint and then refining it. Since finding a feasible solution as close to the optimal as possible in length will likely reduce the number of exchanges required later, we start by creating a feasible initial tour that is, hopefully, of reasonable length.

We create the initial tour using two different methods called the “Sequential TSP” method and the “Giant TSP” method, which will both be used to create candidate solutions. These candidate solutions will then be compared and the one with the least total distance is chosen as the heuristic solution. Using the Sequential TSP Method, the initial tour is created by linking together individual TSP heuristic solutions to subsets of the overall set of nodes. These TSP heuristic solutions are obtained by running the 2-opt heuristic and the subsets are created based on node priority. For an HTSP with constraint d, we group nodes of every $d + 1$ priorities together. That is, priorities 1 to $d + 1$ are one priority group, priorities $d + 2$ to $2d + 2$ are a second priority group, etc., which totals

$\lceil \frac{p}{d+1} \rceil$ groups. We then add the depot to each of the groups and run the 2-opt heuristic to solve each as a classical TSP problem. The TSPs are then linked together to create a feasible initial tour.

The Giant TSP Method starts by using the 2-opt heuristic to solve a classical TSP over the set of all nodes, including the depot. After the 2-opt solution has been found, we once again group nodes of every $d+1$ priorities together into a priority group, where d is the HTSP constraint. Starting at the depot, we visit each node in the first priority group in the order in which it appears in the 2-opt solution, before visiting each node in the second priority group also in the order it appears in the 2-opt solution. We then follow a similar process for the remaining priority groups, until all nodes have been included to into the tour. Finally, the last node to be included in the tour is connected to the depot to create a feasible initial tour

Next, we use a modified 2-opt procedure to improve this initial tour. Like the 2-opt heuristic, this modified 2-opt procedure relies on systematically exchanging two edges until no exchange can be found that will improve the tour. However, 2-opt exchanges can create an invalid tour under our HTSP constraint. We must, therefore, restore the tour to satisfy the HTSP constraint after each 2-opt exchange, before determining whether the 2-opt exchange has improved our tour. Like the original 2-opt heuristic, this modified 2-opt heuristic for the HTSP will stop when we reach a tour where there does not exist a 2-opt exchange followed by restoration that will improve the tour distance.

Finally, we repeat the above procedure five times and return the best tour of the five. This step aims to increase the quality of the results while increasing the run time by a constant factor.

4 Procedure

4.1 Generating the Initial Tour

The first step for our HTSP heuristic involves creating an initial tour that satisfies our HTSP constraint. This tour should be of reasonably high quality in order to reduce the number of exchanges performed in the later steps of the heuristic.

As before, the HTSP constraint is represented by the integer d. We group each $d+1$ priorities together, starting from highest to lowest priority. For a problem with p priorities where the highest priority is 1, there are $\lceil \frac{p}{d+1} \rceil$ "priority groups" and the ith group includes all nodes of priority $1+(i-1)(d+1)$ to $i(d+1)$. For example, an HTSP with 8 priorities and constraint d=2 will be divided up into three groups. The first priority group has priorities {1,2,3}, the second has priorities {4,5,6}, and the third has the remaining priorities {7,8}. These groupings are chosen because the HTSP constraint allows any node of priority 1 to d+1 to be visited first, which includes exactly the nodes in the first priority group. Therefore, when all of these nodes are visited, only nodes with priorities $d+2$ or greater remain. Any nodes of priority $d+2$ to $2d+2$, which are the nodes in the second priority group, can now be visited under the HTSP constraint. This pattern continues where once we visit all nodes in the ith group, we can then visit any node in the $(i+1)$th group under the HTSP constraint d. Therefore, we can create a valid initial tour in this manner under the HTSP constraint by starting at the depot, visiting all nodes of each of the i priority groups in order, and then returning to the depot.

4.1.1 Generating an Initial Tour Using the Sequential TSP Method

To reduce the total distance traveled, we visit the nodes in each group by finding a solution to a classical TSP over each group. The depot location is also added to each priority group to help connect the various TSP solutions together. Each TSP can be solved using the 2-opt heuristic as described earlier.

At this point, we have a TSP solution for each priority group. The next step is to link these TSP solutions together in priority group order. The TSP solutions for the first and last priority groups will also be connected to the depot, thus creating a complete tour.

To link the individual TSP solutions of consecutive priority groups, we look only at the two nodes called "edge nodes" in each TSP that are connected to the depot. For each priority group except the final ($\lceil \frac{p}{d+1} \rceil$) group, we randomly select one of the two edge nodes for the ith group and connect it to a randomly selected edge nodes of the $(i+1)$th group. Following this procedure will leave exactly one node from the first priority group and one node from the final priority group connected to the depot. Figure 3 shows an example of these steps with two priority groups. The figure starts with the TSP solutions of the two priority groups and shows how these TSP solutions are then linked together to create a valid HTSP tour. In each step, the edge nodes of each priority group are circled. All of the TSP solutions will also be connected in order of priority group, thus producing a valid initial tour for the HTSP.

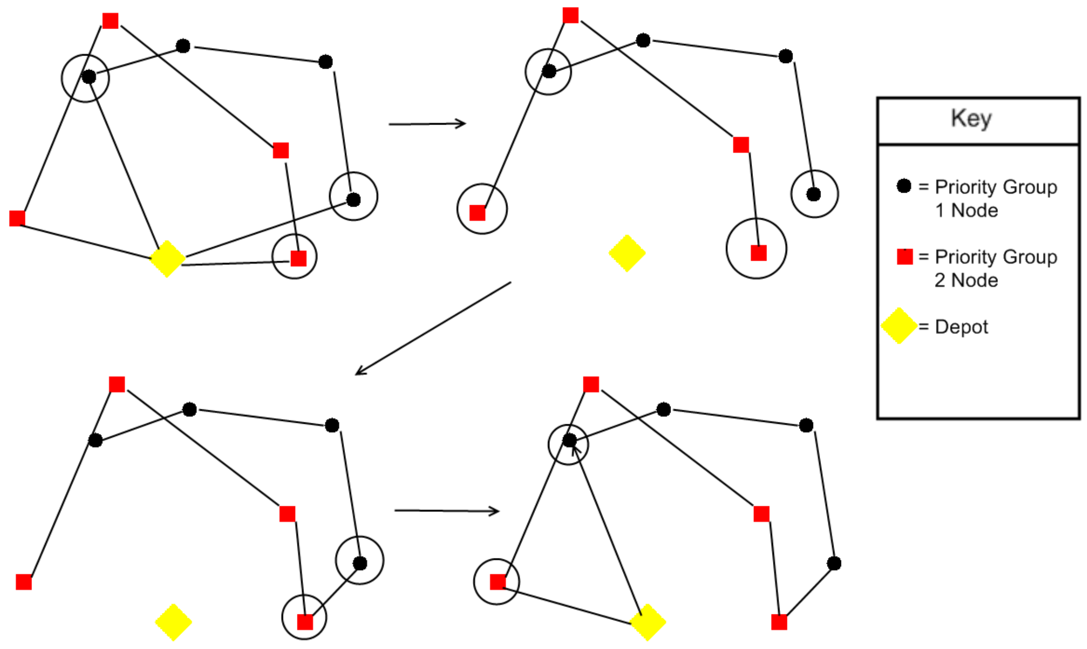

Figure 3: Example of Creating an Initial HTSP Tour

4.1.2 Generating an Initial Tour Using the Giant TSP Method

Another method for creating a feasible solution is based on solving a classical TSP over all nodes, including the depot. By noting the order that the nodes appear in the solution of a classical TSP, we can improve the quality of our initial tour. We first solve a classical TSP over all nodes using the 2-opt heuristic. We then create a TSP subtour for each priority group by connecting the depot and all nodes in the priority group together in the order they appear in the 2-opt heuristic solution.

For example, the subtour created for the first priority group would start at the depot and then trace through the 2-opt heuristic solution in a clockwise direction. We remove from the subtour each node that is not the depot and does not belong to the first priority group (i.e. does not have priority 1 to d+1). At the end, the subtour will include only the nodes in the first priority group and the depot, while also visiting them in the order they appeared in the 2-opt solution for a classical TSP over all nodes.

After repeating this process for creating subtours of the remaining priority groups, we then link together all the subtours using the same method as we did used in the Sequential TSP Method to connect the individual TSP solutions. This method can be found in section 4.1.1.

4.2 Refining the Initial Tour using Modified 2-opt

With our initial tour completed, we begin to improve the tour using a modified 2-opt heuristic that takes into account the HTSP constraint. The approach for the modified 2-opt heuristic is almost identical to the classic 2-opt. We systematically make 2-opt exchanges for each pair of edges in a tour, and preserve the 2-opt exchange only if it improves the tour by lowering the total distance. Otherwise we undo, the 2-opt exchange.

The key difference between the modified 2-opt heuristic and the classical 2-opt heuristic is the need to satisfy the HTSP constraint. Since 2-opt exchanges can be performed on any pair of edges in the tour, they can modify an initially valid tour in a way that violates the HTSP constraint. Therefore, we present a"repair" method that modifies the tour so it again satisfies the HTSP constraint.

The repair method begins with the removal of all "out-of-order" nodes, which are nodes that violate the HTSP constraint. A node with priority n is considered to violate the HTSP constraint if its position in the tour occurs before all nodes of the first $n-(d+1)$ priorities have been visited. To find the out-of-order nodes, we examine each node in both a counter-clockwise and clockwise direction from the depot. We add a node to the set of out-of-order nodes if we reach the node, which has priority n, while there is at least one unvisited node of with priority 1 to $n-(d+1)$. After going through the tour in both a clockwise and counter-clockwise direction, we record the direction that has the smaller number of out-of-order nodes as the "direction of choice."

At this point, we are now interested in re-inserting the out-of-order nodes back into the tour. To do so, we select from the out-of-order nodes at random and use a trial-and-error method to insert that node into every possible position in the tour. We start by inserting to the location next to the depot in the direction of choice. After each insertion, we record whether the tour is valid under the HTSP constraint and if so, we also record the distance of the current tour. Afterwards, we undo the insertion and repeat the insertion procedure by inserting the node into the next position in the direction of choice. After all possible locations for insertion have been tested and resulting tour distances recorded, we finally insert the node into the location that produces a valid HTSP tour under the HTSP constraint with the lowest distance.

The rest of the out-of-order nodes are inserted in the same way, where a random node is selected and inserted into the least-cost location that maintains a valid tour under the HTSP constraint. The repair method ends when all out-of-order nodes have been placed into the tour, resulting in a tour that again includes all nodes and also satisfies the HTSP constraint.

The remainder of the modified 2-opt proceeds identically to the classic 2-opt heuristic. The heuristic continues to systematically make 2-opt exchanges followed by the repair method, preserving the exchange and subsequent repair only if they improve the tour by lowering the total distance. The procedure concludes when we find a tour where no 2-opt exchange followed by the repair method will improve the tour.

This modified 2-opt procedure works by iteratively improving the existing tour and allowing the heuristic to compare numerous valid tours under the HTSP constraint. The heuristic eventually returns the tour with the lowest cost of those that are compared.

4.3 Selecting from the Best from Multiple Runs

To improve our path results, we run the previous steps a total of 5 times to generate 5 candidate tours based on an initial tour generated using the Sequential TSP Method and 5 candidate tours based on an initial tour generated using the Giant TSP Method, thus totaling 10 candidate tours. These tours are likely to have difference because of the random elements of the heuristic. The total distances of these tours will be compared and the one with the minimum total length will be selected as the heuristic-generated tour. Although this step will increase the run-time by a factor of 10, the final heuristic solution will also be improved.

5 Example

5.1 Generating an Initial Tour using the Sequential TSP Method

The following example consists of nodes with priorities 1 through 4 with 4 nodes of each priority, resulting in 16 total nodes in addition to a depot. The nodes are placed in a 1 by 1 square plane. The example will be solved for an HTSP constraint d of 1. For notation, we refer to nodes with priority n as a pn nodes.

We begin by selecting our priority groups, which gives us will give us $\lceil \frac{4}{1+1} \rceil = 2$ priority groups. Since the ith priority group consists of nodes with priority $1 + (i-1)(d+1)$ to $i(d+1)$, p1 and p2 nodes are in the first group and p3 and p4 nodes are in the second group. In Figure 4, nodes in the first priority group are circled and nodes in the second priority group are boxed. The depot is identified by the solid circle.

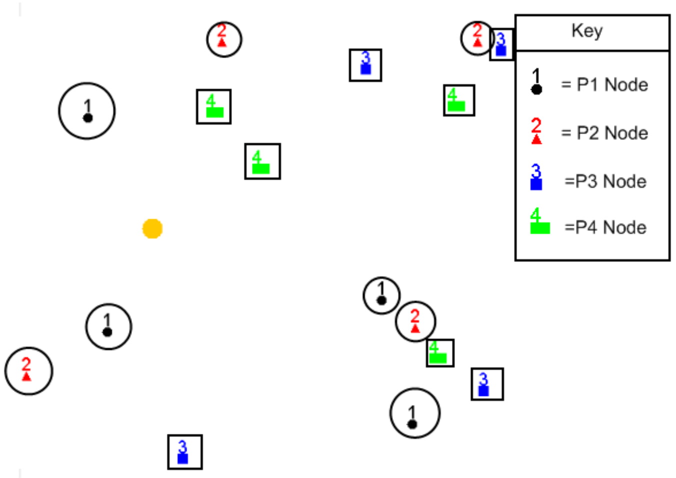

Figure 4: Divisions into groups for our given p, d value combination

We next add the depot to each priority group and solve a classical TSP over each group using 2-opt. Figure 5 shows the approximated TSP solutions as solved by 2-opt for both priority groups.

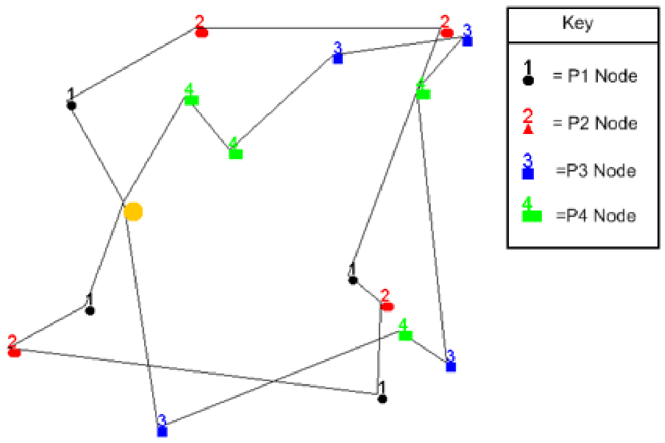

Figure 5: The 2-opt TSP solutions for the two priority groups under d=1.

Next, we link the TSPs together to create a valid initial tour. In our case, the edge nodes that must be rearranged are circled in Figure 6. Since our procedure calls for an edge node of the ith priority group and $(i+1)$th priority group be connected, we need only connect an edge node from the first priority group to the second priority group because there are only two groups in this example.

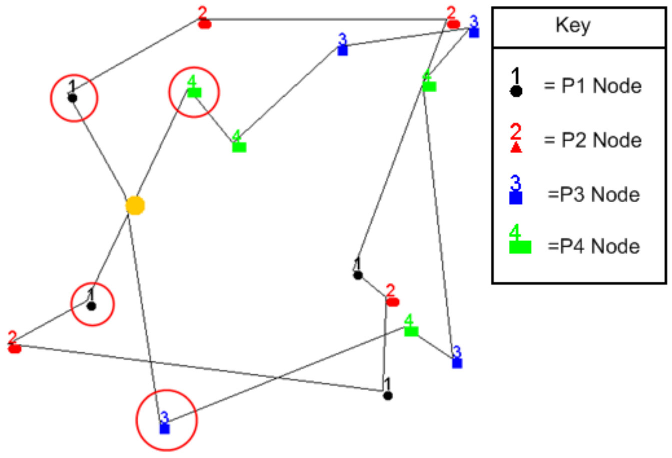

Figure 6: The Edge Nodes in our example are connected to the depot

We randomly select an edge node from the first priority group and connect it to the randomly selected edge node from the second priority group. The two randomly selected edge nodes for this example are circled in Figure 7. The result of connecting these edge nodes is shown in Figure 8. At this point, we have created our initial tour that satisfies our HTSP constraint. By starting from the depot and going in the direction of the adjacent p1 node, the delivery vehicle visits all p1 nodes before visiting a p3 or p4 node, and all p2 nodes before visiting a p4 node. Therefore, the tour is valid under the TSP constraint of $d = 1$.

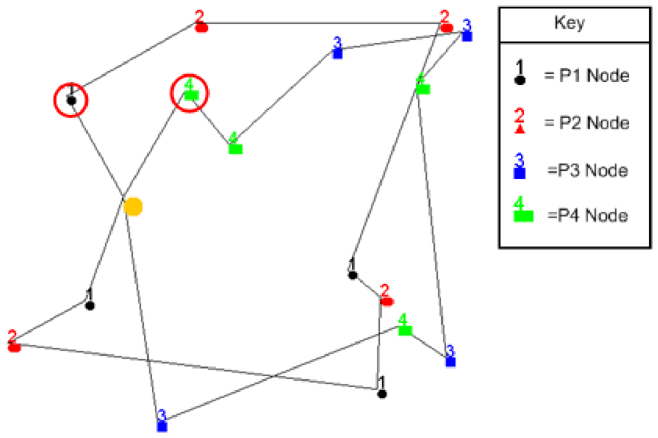

Figure 7: The two randomly selected edge nodes (one from each priority group) that will be connected to each other

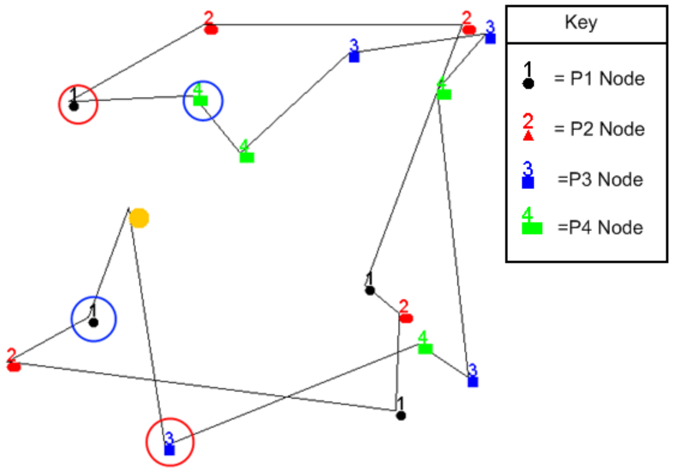

Figure 8: Connecting the edge nodes to create a valid initial tour

5.2 Generating the Initial Tour using the Giant TSP Method

The Giant TSP Method starts with a classical TSP over all nodes. By running the 2-opt heuristic, we see find a heuristic solution to the classical TSP. For our example, this tour is presented in Figure 9.

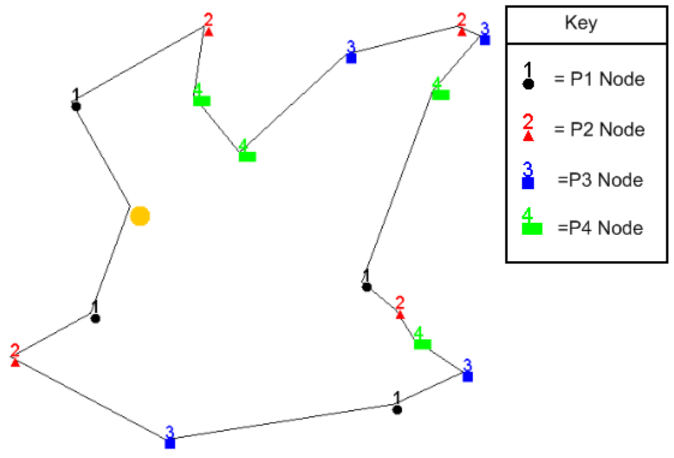

Figure 9: Solution of 2-opt done over TSP that includes all nodes.

Next, we need to form one subtour for each priority group. Priority groups are still divided in the same way as in the Sequential TSP Method, which means we once again have two priority groups. Our first priority group includes p1 and p2 nodes, and the second priority group includes p3 and p4 nodes. For each priority group, we trace through the tour and remove all nodes that do not belong to that priority group. Therefore, the first subtour will include only the depot, p1, and p2 nodes. The p1 and p2 nodes are visited in the same order as they appear in the 2-opt solution to the TSP. The second subtour is created in the same manner for the second priority group, which includes p3 and p4 nodes. The two resulting subtours are illustrated in Figure 10.

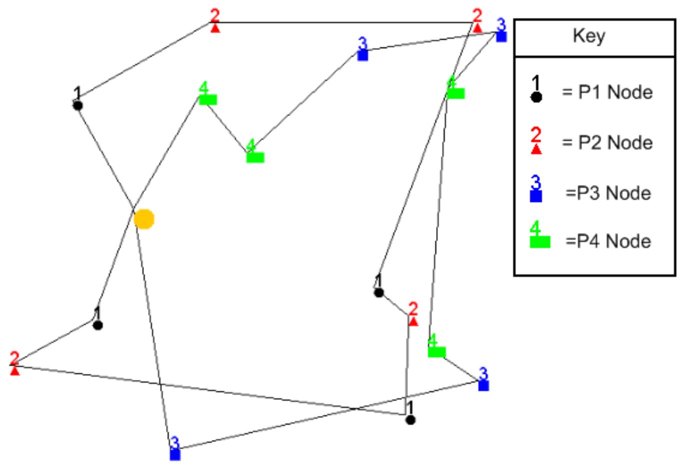

Figure 10: Image of the two subtours for both priority groups

After finding our subtours, we next have to connect them together to form a valid initial tour. As in the Sequential TSP Method, we connect a randomly selected edge node from the first subtour to the second subtour. By connecting an edge node from each subtour of one priority group with the subtour of the next priority group, we create an initial tour that satisfies the HTSP constraint. In this example, we randomly select the edge nodes circled in Figure 11 and connect them to form a valid initial tour as illustrated in Figure 12.

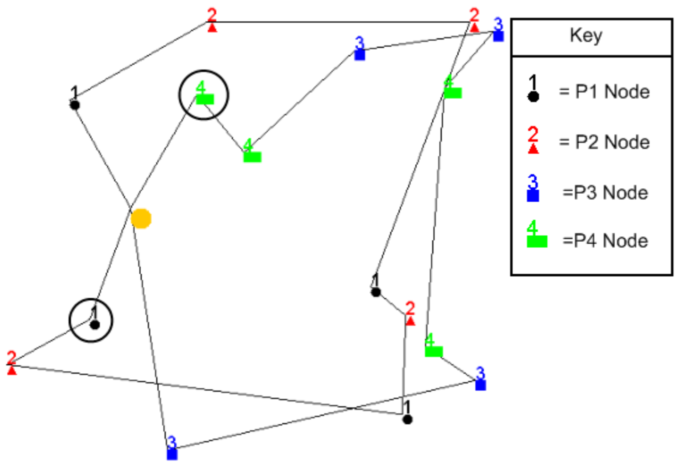

Figure 11: Image of the two subtours and the two randomly selected edge nodes that will be connected

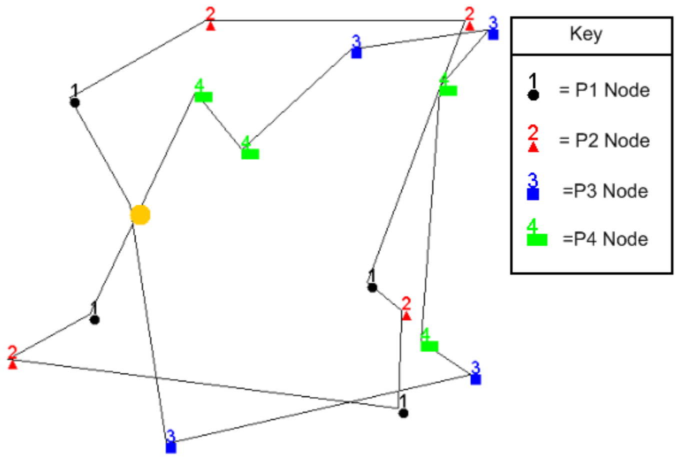

Figure 12: The initial tour created using the Giant TSP Method

5.3 Refining the Initial Tour using Modified 2-opt

With our initial tour completed, we begin to refine the tour. Here, we illustrate two iterations of the 2-opt exchange and repair method. One iteration illustrates the case where the 2-opt exchange does not violate the HTSP constraint, while the other illustrates the case where the HTSP constraint is violated due to the exchange.

Starting off with the path in Figure 8, we perform the 2-opt exchange illustrated in Figure 13, which exchanges the two circled edges. The result creates a new tour that will be tested against the tour before the exchange. However, we must first check whether this tour satisfies the HTSP constraint. By starting at the depot and traveling through the tour in the direction of the p1 node, we see that all of the p1 nodes are visited before the p3 nodes and p4 nodes, and the p2 nodes before the p4 nodes. Under the d=1 HTSP constraint, the exchange produced a valid tour.

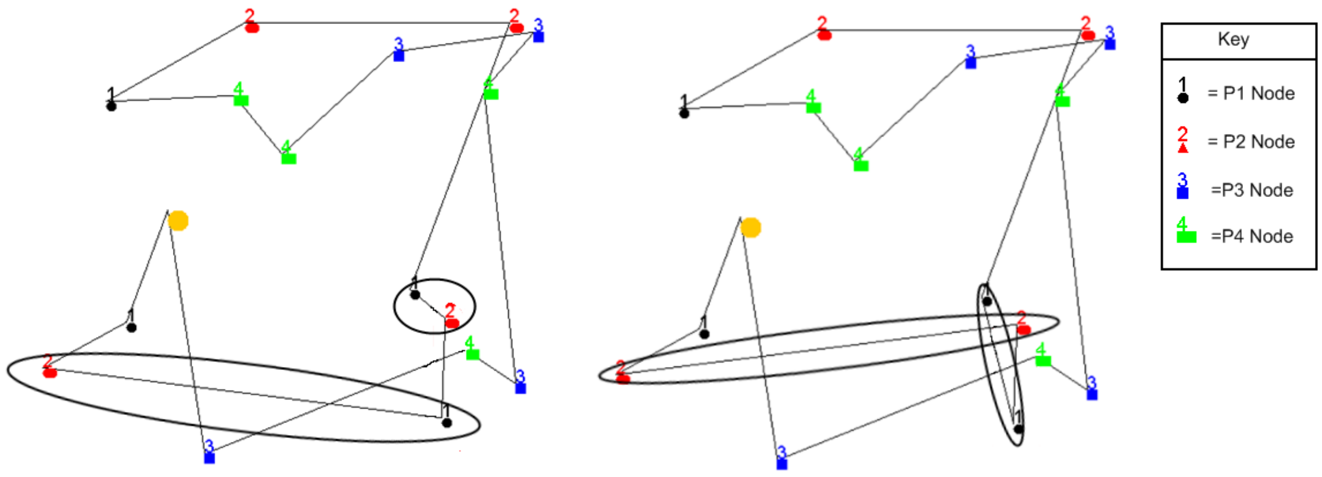

Figure 13: The tour before and after the 2-opt exchange

We next measure the total lengths of the tour before and after the exchange to determine whether this 2-opt exchange should be preserved. We only preserve the exchange if the total length of the tour is improved by the exchange.

In Figure 13, the initial tour before the exchange had a total distance of 5.524, while the new tour after the exchange had a total distance of 5.692. Since the tour before the exchange is shorter than the new one, we do not preserve the exchange.

The next case outlines the procedure when a 2-opt exchange modifies the tour in a way that violates the HTSP constraint. Figure 14 shows our current best tour, where the two circled edges will be exchanged. The result of the 2-opt exchange is illustrated in Figure 15.

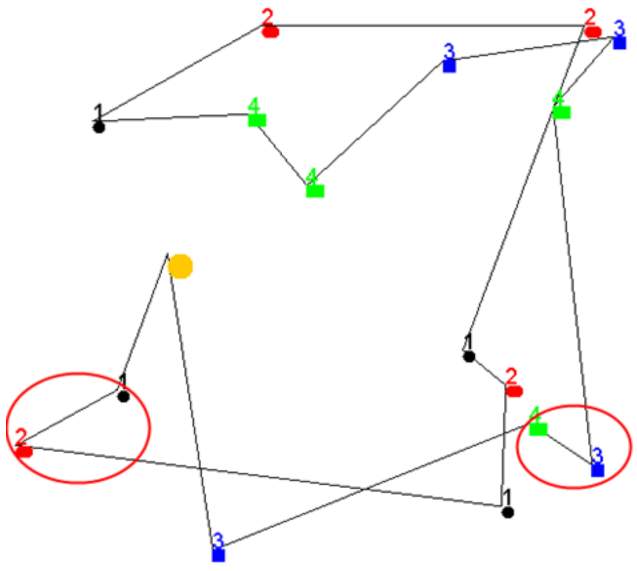

Figure 14: The tour before the 2-opt exchange

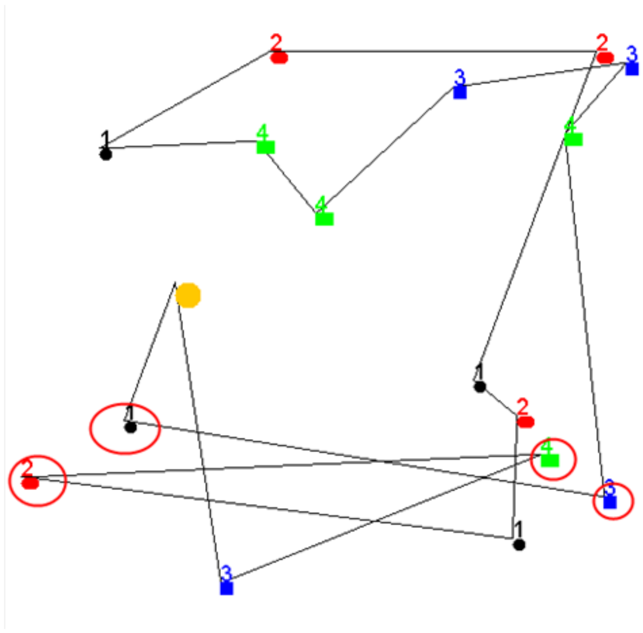

Figure 15: The tour after the 2-opt exchange

Following our exchange, we must next determine whether this new tour satisfies the HTSP constraint. Going right from the depot, we immediately hit a p3 node when not all p1 nodes have been satisfied. Therefore, we must now check the other direction, which turns out to be the direction of choice for this exchange since going left initially from the depot means having fewer out-of-order nodes. Starting from the depot in the direction of choice, we first visit a p1 node, followed by a p3 node. However, visiting the p3 node violates our $d = 1$ constraint because we have yet to visit the three remaining p1 nodes.

Therefore, we must employ the repair method and change the tour to against satisfy the HTSP constraint. Starting at the depot and going in the direction of choice, we remove each node that occurs before it is permitted under the $d = 1$ HTSP constraint. The set of out-of-order nodes includes all nodes circled in Figure 16.

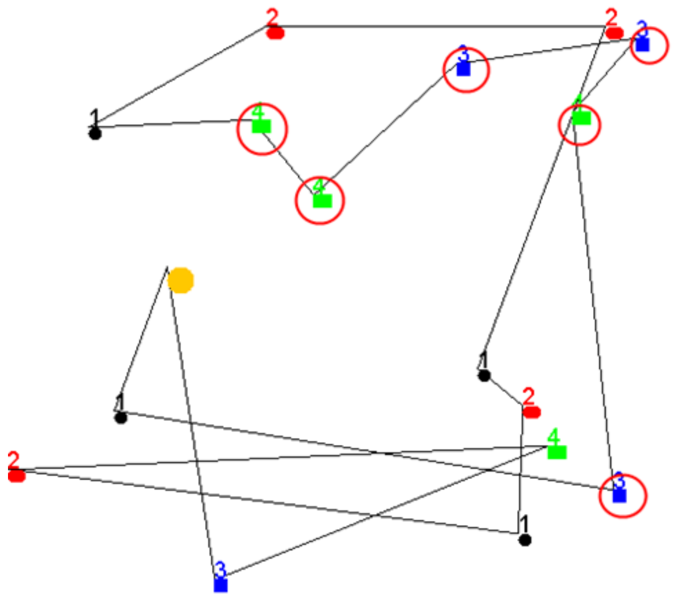

Figure 16: The tour with out-of-order nodes circled in red

Figure 17 shows the tour following the removal of the out-of-order nodes. The surrounding nodes, which are circled, are then reconnected. After reconnecting, the tour satisfies the HTSP constraint.

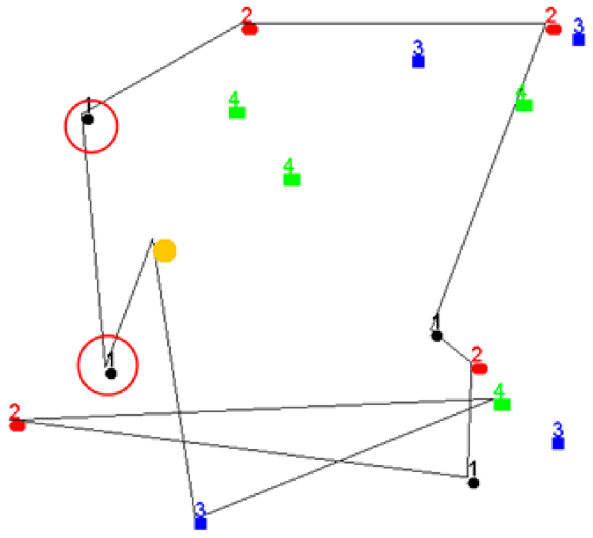

Figure 17: Modified tour with our-of-order nodes removed

Next, we re-insert the out-of-order nodes back into the tour in locations that will not violate the HTSP constraint. We begin by randomly selecting the node indicated in Figure 18. We then try inserting the node into all possible locations along the tour, starting from the position between the depot and the p1 node to the depot's left, and continuing in the direction of choice.

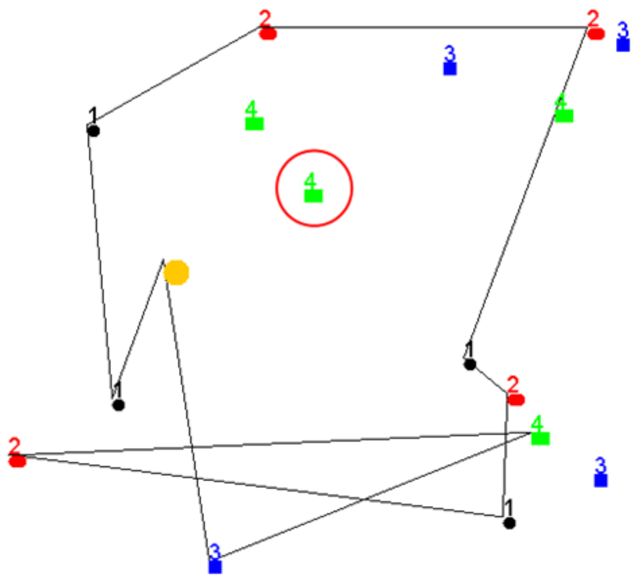

Figure 18: A random out-of-order node is selected for re-insertion to the tour

After inserting the node to the first location, the tour looks like Figure 19. We then test whether the tour is valid under the HTSP constraint. Since this path is invalid under the HTSP constraint, we ignore this tour and move on to the next position. The next few positions are also invalid because the p4 node can only be inserted after all p1 nodes and p2 nodes have been visited. Therefore, the p4 can only be inserted to positions later in the path.

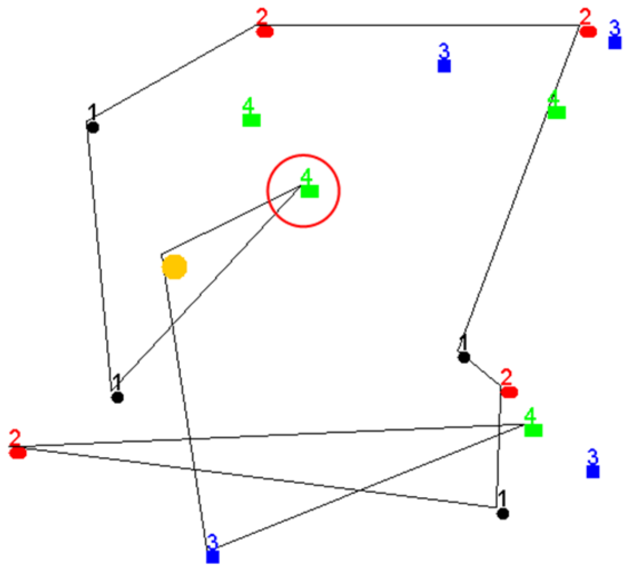

Figure 19: Re-insertion of a randomly selected out-of-order node to one possible position

Figure 20 shows one possible insertion location for the selected node, which creates a tour that satisfies the HTSP constraint. Because the tour is valid, we next calculate the distance of the tour. This particular insertion produces a tour with a total distance of 5.168. However, the node may also be inserted according to Figure 21, which also creates a valid HTSP tour with a lower total distance of 5.122. After trying to insert the selected node at all locations, we would conclude that the optimal place to insert the select node is at the location in Figure 21 because it produces the valid HTSP tour with the lowest distance.

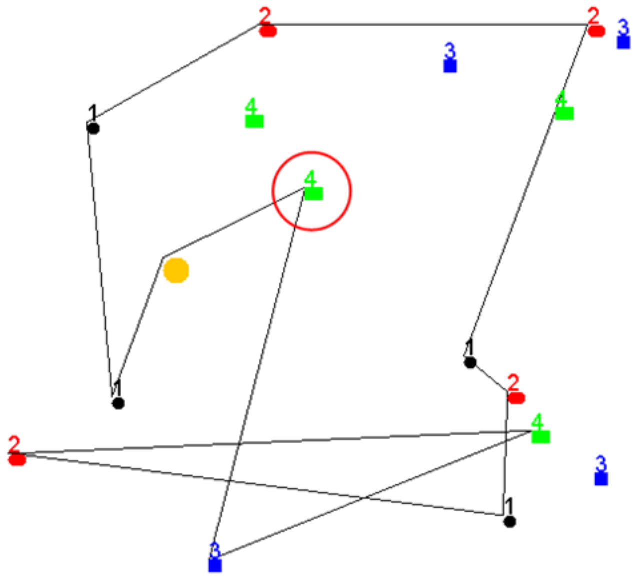

Figure 20: Re-insertion of a randomly selected out-of-order node to another possible position

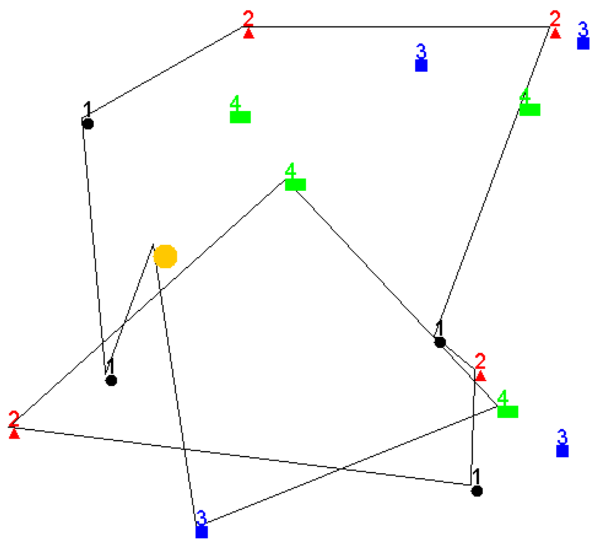

Figure 21: Re-insertion of a randomly selected out-of-order node to another possible position

The other out-of-order nodes are re-inserted in an identical manner to complete the repair method. The new tour created by the repair process is illustrated in Figure 22. The next step is to determine whether the 2-opt exchange and repair resulted in an improvement of the tour's distance. We see that the total distance of 6.498 is worse than the distance of the tour before the exchange of 5.524. Therefore, we undo the exchange and return the tour to the state illustrated in Figure 14. We then continue with the next 2-opt exchange.

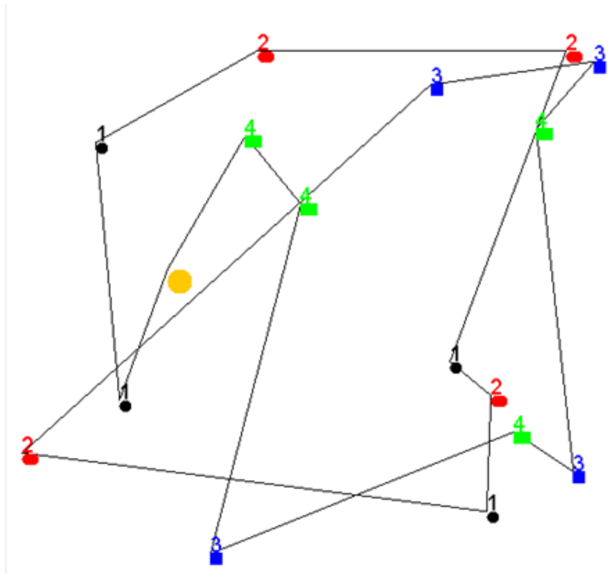

Figure 22: A tour that has been fixed with the repair procedure to satisfy our d constraint after an exchange. This tour will then be compared with our current best tour

Figure 23 shows the final tour returned by the modified 2-opt heuristic. This tour was returned because no possible 2-opt exchange followed by the repair method was found to improve the tour's distance. In comparison to our beginning tour in Figure 14, we see some distinct differences in the placement of the p3 and p4 nodes. The total distance for this path is 5.213. Compared to the initial in Figure 14 of 5.524, this process resulted in a 22.5% decrease in tour length.

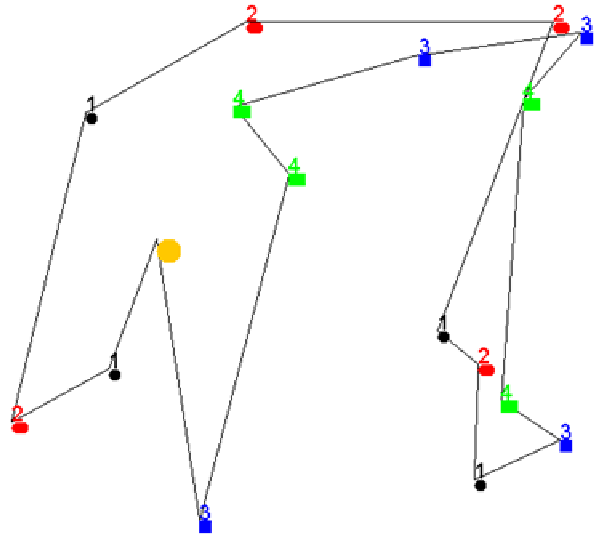

Figure 23: The new tour after the refining process. The total distance is 5.213

5.4 Selecting from the Best from Multiple Runs

Running the previous steps 5 total times and selecting the tour with the minimum total length as the heuristic-generated path still produces the path in Figure 23. In our example, Figure 23 turns out to be the optimal solution.

6 Data Analysis

6.1 Analysis of Performance against Known Solutions

We analyzed three sample data sets consisting of a depot and 16 nodes of 4 distinct priorities, with in 4 nodes of each priority. The values for these data sets are given in Appendix A. The Data Set 1 was arbitrarily selected, while Data Set 2 and Data Set 3 consist of randomly placing nodes across a 1 unit by 1 unit plane. The optimal solutions were calculated for each of these data sets for all possible values of the HTSP constraint d, which for four priorities means d =0, 1, 2, and 3.

6.1.1 Testing Procedure

For each data set, we ran the heuristic 100 times for each d value and calculated summary statistics for the results. We then compared the values and found a percent error by calculating $\frac{LengthofHeuristic-PathLength-LengthofOptimalSolution}{LengthofOptimalSolution} \times 100\%$. Finally, we plot a histogram of the resulting percent errors and calculate the mean for each case. The 5 possible values of d and 4 data sets created 12 different cases for our test.

6.1.2 Results

The results of our tests on Data Sets 1, 2, and 3 are summarized below. Each value represents the average % error over 100 runs of the heuristic. Histogram data for the results are also provided in Appendix A in Figures 24, 25, and 26.

Table 1: Heuristic Solution Compared Against Optimal Solution
% Error from Optimal Solution

	d = 0	d = 1	d = 2	d = 3
Data Set 1	0.00%	0.28%	0.34%	0.00%
Data Set 2	0.00%	1.85%	0.00%	0.00%
Data Set 3	0.00%	0.00%	0.00%	0.00%

6.1.3 Discussion

Overall, the heuristic produced tours that were close to the optimal solutions in most situations. The average percent errors relative to the optimal solutions for the twelve cases ranged from a 0% to 1.85% greater than the length of the optimal tour. However, only one case of when $d = 1$ for Data Set 2 averaged more than 0.34% different from optimal. All other cases generated tours that averaged within 0.34% of the optimal solutions.

6.2 Analysis of Runtime and Empirical Determination of Algorithmic Complexity

The next step required testing the heuristic on larger data sets to determine how the heuristic's performance behaves as the data set size grows. The heuristic was run multiple times on randomly data sets with varying numbers of nodes. For each arbitrarily selected value of # Nodes = X, we generated X nodes with random locations that were evenly split among p1 to p4. We then ran the 2-opt-based heuristic for a given HTSP constraint d, and then repeated this process 30 times to record 30 values for the execution time of the heuristic. We then repeated this process for d =0, 1, 2, and 3 with the number of nodes taking the values 8, 12, 16, 24, 32, 40, 52, and 64. Finally, we averaged the execution times to provide estimates for the execution time in each case. The resulting averages are summarized below.

Table 2: Runtime for Heuristic on Problems of Varying Sizes
Average Time Per Trial in ms

# Nodes	d = 0	d = 1	d = 2	d = 3
8	4.90	7.03	20.10	9.00
12	226.03	153.10	88.70	15.47
16	754.13	447.47	256.53	28.83
24	4934.07	3307.33	1705.67	98.30
32	20600.20	12625.50	7274.80	226.07
40	61954.87	37321.90	19443.33	454.97
52	247432.37	135330.67	67082.57	953.87
64	674019.60	451086.07	188809.27	1972.93

As the table shows, the time for $d = 3$ is consistently less than the time for other d values with the same number of nodes. The case where $d = 0$ is also consistently faster to solve than $d = 1$ and $d = 2$. The longest run time occurred with $d = 1$ and 64 nodes, which took 146 seconds or about 2.5 minutes to solve a single problem. However, larger problems will take significantly longer to solve. Overall, the run time is a function of n, d, and nd. A regression analysis produces a least-squares best fit equation of $O(n^5+n^5d^3+d^3)$, which has an r^2 value of 0.903 against the above data.

7 Appendix A: Data Sets

7.1 Data Set 1

7.1.1 Node Locations

Data Set 1 consists of four priorities, having a depot and four nodes of each priority. The coordinates of the depot and each node is displayed below.

Table 3: Data Set 1 - 16 Nodes and a Depot

Depot	P1	P2	P3	P4
(.2614, .4496)	(.1441, .2434)	(.7854, .6538)	(.3304, .9097)	(.8585, .2218)
	(.7191, .5993)	(.0258, .7468)	(.9195, .7774)	(.8220, .7138)
	(.7790, .8406)	(.4074, .0948)	(.6896, .1498)	(.4774, .3423)
	(.1829, .6605)	(.9064, .0952)	(.9526, .1137)	(.3867, .2338)

7.1.2 Performance of Heuristic

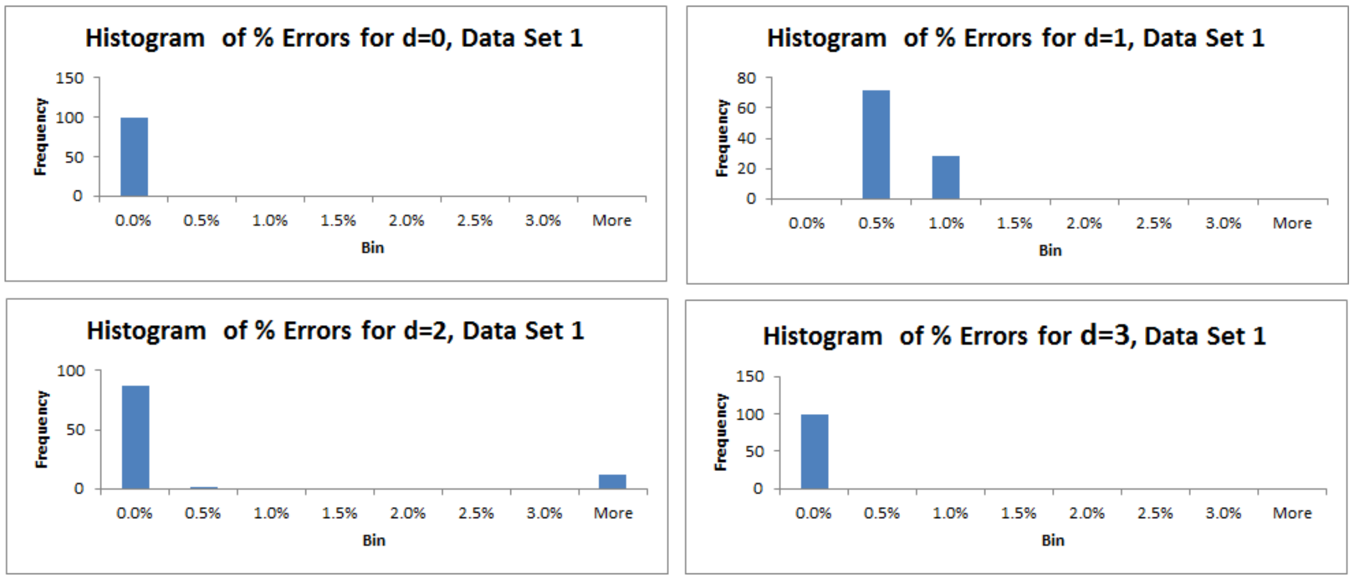

Figure 24: Histogram of % Errors For 100 Runs of Heurstic Solution Against Optimal Solution for d = 0, 1, 2, and 3

7.2 Data Set 2

7.2.1 Node Locations

Similarly, Data Set 2 consists of four priorities, having a depot and four nodes of each priority. However, the locations of Data Set 2's nodes were selected using a pseudo-random number generator. The coordinates of the depot and each node is displayed below.

Table 4: Data Set 2 - 16 Nodes and a Depot

Depot	P1	P2	P3	P4
(0.604,0.601)	(0.835,0.178)	(0.057,0.009)	(0.842,0.388)	(0.431,0.680)
	(0.643,0.106)	(0.849,0.253)	(0.521,0.157)	(0.646,0.775)
	(0.393,0.279)	(0.217,0.189)	(0.631,0.903)	(0.629,0.443)
	(0.447,0.417)	(0.286,0.617)	(0.323,0.112)	(0.152,0.630)

7.2.2 Performance of Heuristic

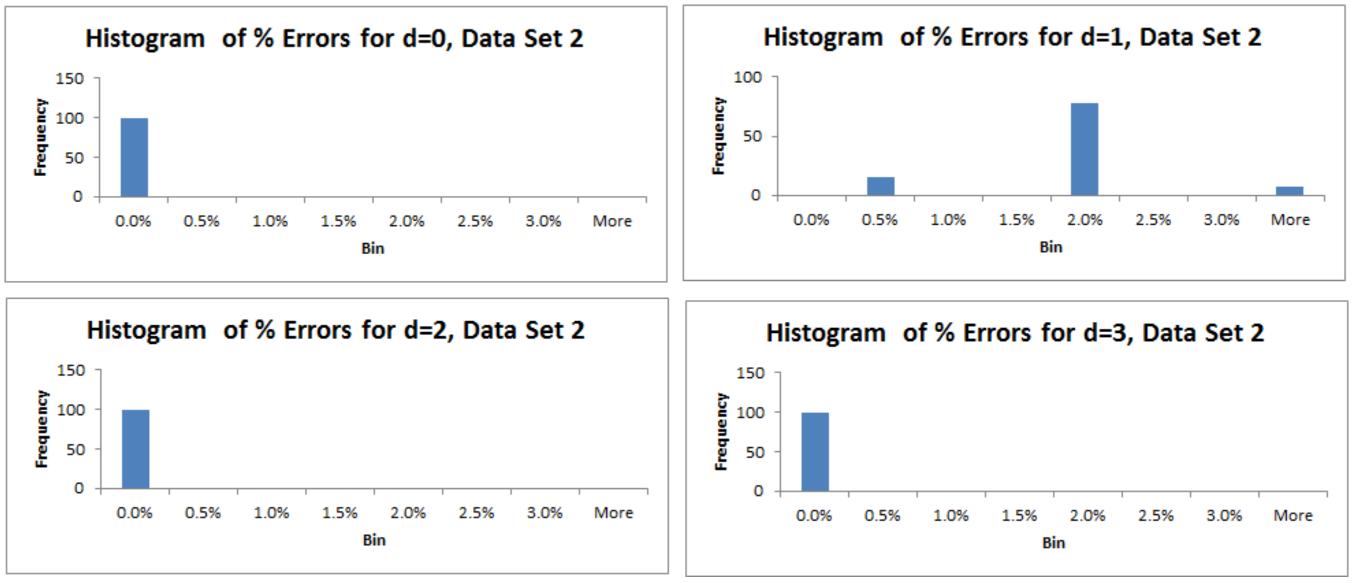

Figure 25: Histogram of % Errors For 100 Runs of Heurstic Solution Against Optimal Solution for d = 0, 1, 2, and 3

7.3 Data Set 3

7.3.1 Node Locations

Similarly, Data Set 3 consists of four priorities, having a depot and four nodes of each priority. Like Data Set 2, the locations of Data Set 3's nodes were selected using a pseudo-random number generator. The coordinates of the depot and each node is displayed below.

Table 5: Data Set 3 - 16 Nodes and a Depot

Depot	P1	P2	P3	P4
(0.729,0.825)	(0.114,0.325)	(0.672,0.499)	(0.139,0.811)	(0.859,0.820)
	(0.329,0.942)	(0.672,0.584)	(0.144,0.507)	(0.677,0.150)
	(0.199,0.670)	(0.254,0.625)	(0.544,0.212)	(0.847,0.561)
	(0.226,0.728)	(0.118,0.157)	(0.904,0.139)	(0.947,0.973)

7.3.2 Performance of Heuristic

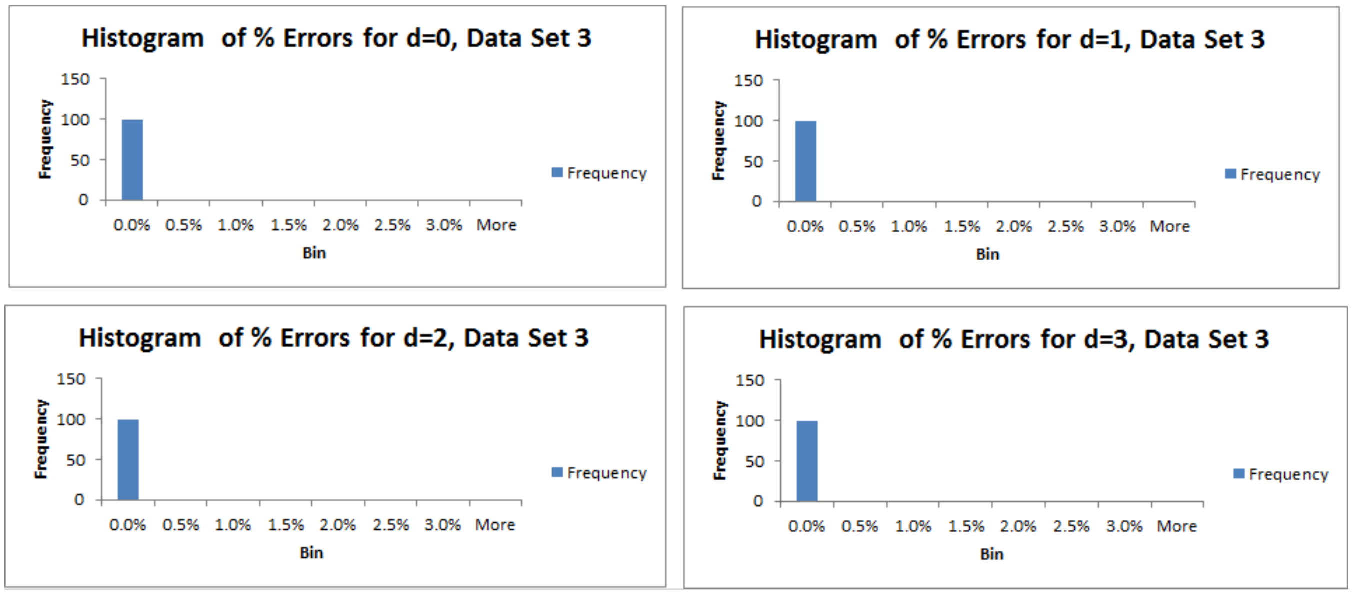

Figure 26: Histogram of % Errors For 100 Runs of Heurstic Solution Against Optimal Solution for d = 0, 1, 2, and 3