The Traveling Salesman Problem Using the Hungarian Method

Cameron Sanchez[1], Cody Kesler[1], William Pinkston[1], and Lucas Pinto[1]

***Abstract*— The Traveling Salesman is a well studied and dissected problem in computer science. Many algorithms have been proposed for this problem with varying benefits in time and space complexity and optimal paths. We show the benefits of the Hungarian Method–commonly used in assignment problems such as finding the maximum weight in a bipartite weighted graph–to the Traveling Salesman Problem. It has a time complexity of O(n(n-1)!) and a space complexity of O(n^2). Although slower than a greedy algorithm, the solution is more optimal in finding the final path than the greedy implementation of the Traveling Salesman Problem. We provide justification for this method and its trade-offs with the greedy and branch and bound methods.**

I. INTRODUCTION

The Traveling Salesmen Problem (TSP) is a well studied problem in computer science [1]. The problem is: given N number of cities, with corresponding distances between the cities, what is the shortest path that will visit each N city once and return to the originating city. In combinatorial optimization it is considered an NP-hard problem because the algorithm itself cannot be solved in polynomial time. It also falls under the category of NP-complete because it maps to every other NP-complete problem, and it can be verified in polynomial time [2].

The greedy algorithm is a proposed method for solving the TSP, which solves on average of O(n^2). The method is similar to Kruskal's algorithm in that it selects the edges with the least weights first. It will do this iteratively for each subsequent connection unless that node has been visited, until the final path is reached.

Branch and bound is another method for solving the TSP which works by constructing reduced cost matrices of each state in a path, and following that path only if it could potentially reduce the best solution so far (BSSF). When a certain path is known to not improve the BSSF then that entire branch will be pruned (removed from consideration entirely). It runs in worst case time O(n(n-1)!) and space complexity of O(n^3).

II. ALGORITHM DESCRIPTION

The Hungarian Method is an optimization algorithm that has been primarily used for solving the assignment problem in polynomial time [3]. The assignment problem is a core combinatorial optimization problem which consists of finding a minimum or maximum weight match in a weighted bipartite graph. A common example is in residency matching program for physicians. Where all pre-residential physician will rank all the residency programs at which he interviewed in order of most to least preferential. Each residency program will then rank each pre-residential physician who interviewed there in terms of most to least preferential. The max weight is computed so both residency program and pre-residential physician match with each maximum weight.

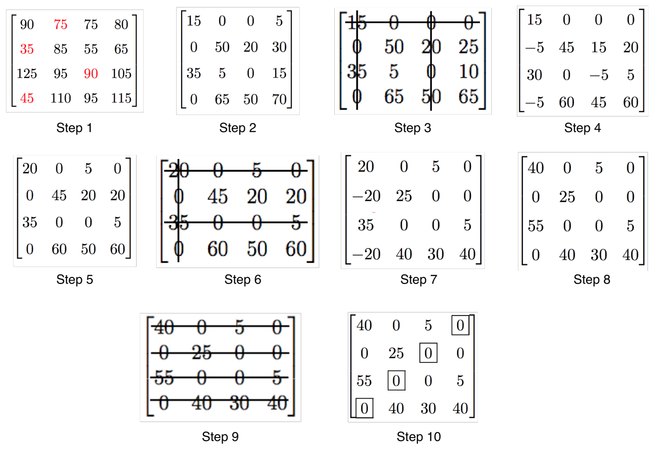

Fig. 1. Step by step process of Hungarian Method on a TSP example. Step 1: Initial cost matrix. Step 2: Reduced cost matrix. Step 3: Crossing out minimum number of columns and rows to cover all zeros. Step 4: Subtract the minimum uncovered value from each uncovered row. Step 5: Add the minimum uncovered value to each covered column. Step 6-8: Repeat steps 3-5. Step 9: Repeat steps 3. Step 10: Final solution matrix. Since previous cross-out step results in N (number of rows/columns) cross-outs the solution has been found.

The Hungarian method has been shown to be an alternate method to solve the TSP [4]. We implement this method here and compare its performance to the two other algorithms discussed. The method works by taking the initial cost matrix of all cities and reducing it. Once it has been reduced, the matrix rows and columns are "crossed out" until you have reached a matrix solution in which the minimum number of "cross-outs" to cover up all the zeros in the matrix is equal to the number of rows/columns (see Figure 1).

After obtaining the initial cost matrix, the goal is to cross out all zeros in the matrix by using the minimum number of cross-outs. This is done by repeating two steps until all zeros are crossed out. Step 1: Scan all the rows for zeros. If exactly one zero is found in a row, the column of the its location is crossed out. Step 2: Scan the columns for zeros with the same process: if exactly one zero is found then the row of its location is crossed out [5].

If the minimal solution cross-out number is not equal to the number of rows or columns, the minimum uncovered value of the matrix at that point (as in step 3) will be

*This work was not supported by any organization

[1]Department of Computer Science, Brigham Young University, Provo, UT, USA.

TABLE I
EMPIRICAL ANALYSIS RESULTS

Algorithm:	Random	Greedy			Branch and Bound			Hungarian Algorithm		
# Cities	Path Length	Path Length	Time (s)	% Improved	Path Length	Time (s)	% Improved	Path Length	Time (s)	% Improved
15	19297	10734	0.01	0.44	9504	22.64	0.12	10459	1.60	0.03
16	24239	12118	0.01	0.50	9084	150.79	0.25	11821	33.96	0.02
17	24239	12118	0.01	0.50	9084	150.79	0.25	11821	33.96	0.02
18	23689	11613	0.01	0.51	9962	292.42	0.14	11236	18.63	0.03
19	27784	13108	0.01	0.53	10713	302.49	0.18	11272	4.34	0.14
20	26402	14497	0.02	0.45	10669	351.42	0.26	12006	11.06	0.17
21	28788	14252	0.02	0.51	TB	TB	TB	13371	73.30	0.06
22	32099	13680	0.02	0.57	TB	TB	TB	TB	TB	TB
25	38506	16726	0.02	0.57	TB	TB	TB	TB	TB	TB
30	39909	17021	0.04	0.57	TB	TB	TB	TB	TB	TB
60	77875	25310	0.20	0.68	TB	TB	TB	TB	TB	TB
100	132688	35954	0.87	0.73	TB	TB	TB	TB	TB	TB
200	256459	56331	6.75	0.78	TB	TB	TB	TB	TB	TB

subtracted from all uncovered rows (step 4). This same value is added to all covered columns (step 5). These steps are repeated until the final matrix solution is found by following the zero paths.

Once the final matrix is computed, starting from an arbitrary initial city (or row) the matrix is traversed through all zero locations. If a complete tour of the cities if found then it is the closest to optimal. If during traversal the path clearly stops or repeats itself the search will restart from the initial city including the next minimum of the matrix in its search scope allowing other plausible paths. All edges are then culminated to return a valid path.

III. Performance Analysis

This algorithm has a worst-case O(n(n-1)!). Creating a matrix of the cities takes O(n) time. Reducing the matrix takes O(n^2) time. Covering the zeros with the minimal number of lines takes O(n^3) time repeating at most n times. All of these happen in a linear fashion. Once the matrix is completely reduced, finding a path through the matrix takes O(n(n-1)!) time. The algorithm looks through each branch reachable through zero; if a complete route is not found, it then looks for every branch reachable considering both zero and the next minimal number in the matrix. If a complete route is still not found, the step is repeated with the next minimum, and so on. At worst case, this could generate (n-1)! branches if a complete route is not found until every number in the matrix is considered as a possible path. This will repeat n times by recursion, bringing the functionality to find a solution to O(n(n-1)!). This happens in linearity with the other functions bringing the complexity to O(n(n-1)! + n^4) - reducing to O(n(n-1)!).

Although the worst-case complexity is factorial, in practice the algorithm will usually finish much sooner as long as the graph has a solution. Overall, it performs much faster, theoretically and empirically than the typical Branch and Bound algorithm, at the expense of a little bit of optimality in the cost of the overall solution.

A. *Empirical Analysis*

Table 1 shows the results of the empirical analysis, which first compares the results of the greedy implementation to a random solution (to ensure the greedy algorithm is implemented properly as a baseline) and then compares both the Branch and Bound and the Hungarian Method to the results of the greedy algorithm. As shown, the greedy algorithm is incredibly fast compared to both the branch and bound as well as the Hungarian method, but it shown to not produce the optimal solution and in fact will effectively never produce the optimal solution on problems larger than 5 or 6 cities. This is where the advantage of the Hungarian Method shone forth, it is faster than the Branch and Bound and produces a more optimal path than greedy.

1) Considering the optimality of the solution: The Hungarian Method empirically produces the optimal solution approximately 33% of the time, with an average gain of 3-5% on the cost of the optimal solution. This is a major improvement over the greedy implementation, which tends to produce solutions that cost over 10% more than the optimal solution and never the optimal solution.

2) Considering the optimality of the time complexity: While theoretically it only has a n^2 advantage over the Branch and Bound algorithm, in practice it produces a solution in a much more efficient manner averaging a 200-300% increase in time spent running the algorithm. Observed runtime for the Hungarian Method showed wide variance between problems of the same size (n), indicating the likely existence of a unifying characteristic among these pathological cases. Similarly, the algorithm was also observed to produce the solution found by the full execution of Branch and Bound within hundredths of a second from when the greedy algorithm terminated execution. Thus it is strongly probable that the Hungarian Method could prove effective within a specific application or context.

These results are exciting as they shows that time complexity can be greatly improved while not sacrificing an equally large amount of optimality in the found solution, making the trade off for time complexity an overall improvement to finding the solution to the TSP problem.

IV. CONCLUSIONS

As seen from the algorithm and results above, the Hungarian Method takes each requirement of space, time, and optimality into consideration. The theoretical run time of O(n(n-1)!) has an edge over the theoretical complexity of the Branch and bound algorithm which runs at O(n^3(n-1)!), but is also slower than the greedy which runs at O(n^2). The Branch and Bound algorithm is guaranteed to find the optimal solution given enough time while the Hungarian Method, although not always as accurate, empirically produces the optimal solution 33% of the time while running at much faster speeds. The greedy algorithm, on the other hand, is on average 10% worse than the optimal solution. Though slower, the Hungarian Method consistently finds a more optimal solution than greedy and has a good probability of achieving the optimal solution. The Hungarian Method is not the perfect catch-all to solving the TSP problem, but it provides a new look at balancing space, time, and optimality while not compromising.

ACKNOWLEDGMENT

We thank Dr. Farrell, for his support and teaching us throughout the year, and all the TAs for the time they spent in person and on slack answering questions.

V. REFERENCES

REFERENCES

[1] APPLEGATE, D. L. *The traveling salesman problem: a computational study*. Princeton University Press, 2006.

[2] GOLDREICH, O. *P, NP, and NP-Completeness: The Basics of Computational Complexity*. Cambridge University Press, 2010.

[3] KUHN, H. W. The hungarian method for the assignment problem. *Naval Research Logistics Quarterly 2* (1955), 29–47.

[4] UNIVERISTY, H. The assignment problem and the hungarian method.

[5] WISE, K. [#1]assignment problem—hungarian method—operations research[solved problem using algorithm].