Mathematical and Computational Applications, Vol. 18, No. 3, pp. 459-466, 2013

A PARAMETRIC HYBRID METHOD FOR THE TRAVELING SALESMAN PROBLEM

Gözde Kızılates and Fidan Nuriyeva

Department of Mathematics, Ege University, 35100, Bornova, Izmir, Turkey.
gozde.kizilates@gmail.com, nuriyevafidan@gmail.com

Abstract- The traveling salesman problem (TSP) is one of the typical NP–Hard problems of combinatorial optimization area. This paper proposes a new hyper heuristic algorithm named Parametric Hybrid Method (PHM) based on The Farthest Vertex (FV) and Greedy heuristics for solving the traveling salesman problem. In addition, many problem instances from TSPLIB (traveling salesman problem library) were solved with NN, Greedy and PHM algorithms. The experimental results show that the new hybrid algorithm is more effective and efficient than both Greedy and Nearest Neighbor algorithms.

Key Words- traveling salesman problem, nearest neighbor algorithm, greedy algorithm, parametric hybrid algorithm, hyper heuristic algorithm

1. INTRODUCTION

TSP is a well-known and important combinatorial optimization problem which is studied in operations research and computer science [2, 3]. TSP can be described briefly as follows: a salesman wants to sell his goods in *n* city. Salesman visits each city only once and has to go back into initial city. The problem is investigating the route which has minimum cost to do this job. In this paper, we will consider the symmetric TSP [7].

Formally, the TSP can be stated as follows [14]. The distances between n cities are stored in a distance matrix D with elements where and the diagonal elements are zero. A tour can be represented by a cyclic permutation of where represents the city that follows city i on the tour. The traveling salesman problem is then the optimization problem to find a permutation that minimizes the length of the tour denoted by

$$\sum_{i=1}^{n} d_{i\pi(i)} \ .$$

TSP has important part of applications in many areas including vehicle routing, network design, computer wiring, machine sequencing and scheduling, frequency assignment in communication networks, electronic circuit design, transportation and logic applications [4, 8, 9].

Since the TSP belongs to NP-Hard, it is very hard to develop an efficient algorithm for the problem. It is very important to find quality results in reasonable times for the problem, due to its large application area [6].

The rest of this paper is organized as follows. Section 2 describes some approaches for solving the traveling salesman problem. Section 3 presents our proposed parametric hybrid algorithm. Section 4 illustrates our experimental results. Finally, section 5 concludes the paper.

2. APPROACHES FOR SOLVING TSP

TSP is a simple problem, yet it is dramatically hard to solve [1]. Various methods are used to generate solutions for the TSP [15], however, there is no known algorithm that finds an exact solution to the problem in polynomial time. The problem gets extremely difficult after a certain number of cities. Thus, heuristic and hyper heuristic methods are often preferred.

Heuristic algorithms can produce good solutions, but they do not guarantee that the solution they find is optimal. In general, the heuristic algorithms for TSP are subdivided into the following three classes: tour construction algorithms, tour improvement algorithms and hybrid algorithms [16]. The tour construction algorithms gradually build a tour by adding a new city at each step, the tour improvement algorithms improve upon a tour by performing various exchanges, and finally hybrid algorithms use both composing and improving heuristics at the same time [5, 10-13]. The best results are obviously obtained by using hybrid approaches [1, 15].

Hyper heuristics are algorithms searching the "heuristic space" for solving the hard optimization problems. In this sense, a hyper heuristic decides which heuristic is more efficient to solve the problem instead of using a fixed method. In other words, if there are more than one successful heuristics for a problem, deciding which one of these will be even more successful is called as hyper heuristic. A deciding algorithm in situations where there is more than one heuristic applied to the problem is also called a hyper heuristic.

New hybrid heuristic algorithm we propose is based on our previous algorithms NND and The Farthest Vertex (FV), as well as the well-known Greedy and Nearest Neighbor algorithms. Next, we briefly review those.

2.1. The nearest neighbor algorithm (NN)

Among the tour construction heuristics, the nearest neighbor heuristic is the most simple one. The nearest neighbor (NN) algorithm for determining a traveling salesman tour is as follows. The salesman starts at a city then visits the city nearest to the starting city. Afterwards, he visits the nearest unvisited city, and repeats this process until he has visited all the cities, in the end, he returns to the starting city.

The steps of the algorithm are as following:

NN Algorithm

Step 1. Select a random city.
Step 2. Find the nearest unvisited city and go there.
Step 3. Are there any unvisited cities left? If yes, go to Step 2.
Step 4. Return to the first city.

A better result can be obtained by running the algorithm over again for each vertex and repeat it for n times.

2.2. The nearest neighbor algorithm from both end points (NND)

The algorithm starts with a vertex chosen randomly in the graph. Then, the algorithm continues with the nearest unvisited vertex to the chosen vertex. We will have two end vertices. We add a vertex to the tour such that this vertex has not visited before

and it is the nearest vertex to these two end vertices. We update the end vertices. The algorithm ends after visiting all vertices.

The steps of the algorithm are as following:

NND Algorithm

Step 1. Choose an arbitrary vertex in the graph.
Step 2. Visit the nearest unvisited vertex to this vertex.
Step 3. Visit the nearest unvisited vertex to these two vertices and update the end vertices.
Step 4. Is there any unvisited vertex left? If yes, then go to Step 3.
Step 5. Go to the end vertex from the other end vertex.

2.3. The greedy algorithm

The Greedy heuristic gradually constructs a tour by repeatedly selecting the shortest edge and adding it to the tour as long as it does not build a cycle with less than N edges, or increase the degree of any node to more than 2.

The steps of the algorithm are as following:

Greedy Algorithm

Step 1. Sort edges by increasing lengths.
Step 2. Select the shortest edge and add it to our tour if it doesn't violate any of the above constraints.
Step 3. Do we have *n* edges in our tour? If no, go to Step 2.

2.4. The farthest vertex algorithm (FV)

This algorithm is about finding the sums of each row in the adjacent matrix. The algorithm continues to add the minimum two distances of each row which includes the maximum distance to the tour. This process is applied to each row.

The steps of the algorithm are as following:

FV Algorithm

Step 1. Find the sums for each row in the adjacency matrix and add them to SUM column.
Step 2. Find the maximum sum of SUM column.
Step 3. From the same row in which this maximum sum exists, select the two minimum distances which do not contain a sub tour and add them to the tour.
Step 4. Delete the row and the column which correspond to the maximum sum.
Step 5. Increase the number of selected vertices by 1.
Step 6. If the number of selected vertices is less than *n* then go to step 2.

3. A NEW PARAMETRIC HYBRID ALGORITHM

A new Hyper Heuristic algorithm is proposed below:

Initially, NN algorithm is to determine a path whose end vertices are used in NND algorithm. Then NND algorithm continues with an implementation of FV algorithm where the farthest vertices are being considered. Finally, k parameter is included to the method so that FV and the Greedy algorithm work together. The parameter k determines the contribution ratio of the algorithms. For k = 0, the hybrid

algorithm will only use the Greedy algorithm; likewise, for k = n, the hybrid algorithm will only use FV algorithm. When k is in interval (0, n), for the first k farthest vertices FV (or the NN algorithm for these vertices) is performed and the Greedy algorithm runs on the rest of the vertices. We call this hybrid algorithm PFVGH.

The steps of the algorithm are as following:

PFVGH Algorithm

Step 1. Find the sums for each row in the adjacency matrix.
Step 2. Sort the sums in descending order.
Step 3. Identify the k parameter.
Step 4. For the first k vertices of the sorted sums, select the two edges with minimum distances that don't form a sub tour, and add them to the tour.
Step 5. Perform the Greedy algorithm for choosing rest of the edges to complete the tour.

3.1. Computational experiments with PFVGH

The results of the computational experiments conducted on the problems ulysses16 and ulysses22 of TSPLIB [18] are represented below.

Table 1. The results of the computational experiments on the problem ulysses16

k	0	1	2	3	4	5	6	7
Result	88.923	79.133	79.133	78.716	78.147	78.147	76.509	77.144

k	8	9	10	11	12	13	14	15	16
Result	75.813	75.041	75.138	75.138	75.138	75.138	75.138	75.138	75.138

Table 2. The results of the computational experiments on the problem ulysses22

k	0	1	2	3	4	5	6	7	8	9	10
Result	89.436	81.783	81.783	82.159	82.159	79.491	78.193	78.193	78.193	77.868	78.503

k	11	12	13	14	15	16	17	18	19	20	21	22
Result	78.503	77.494	77.494	78.675	78.675	78.675	78.675	78.675	78.675	78.675	78.675	78.675

As shown in Table 1, on the problem ulysses16 the best result was achieved when k was 9. As shown in Table 2, on the problem ulysses22 the best result was achieved when k was 12 and 13.

Similarly, Figure 1 shows the results PFVGH produces for the problem Eil76, depending on the parameter k.

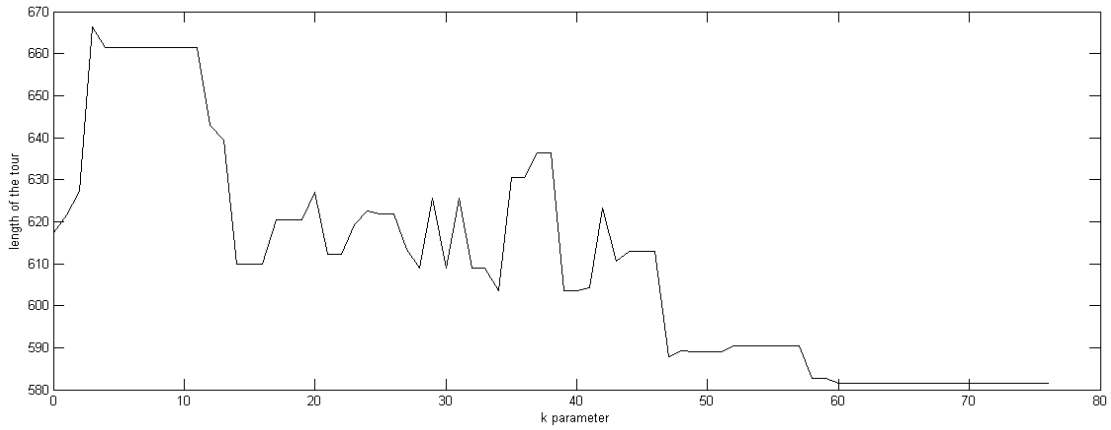

Figure 1. Correlation between the parameter k and the results PFVGH produces for the problem Eil76.

As it is seen in Figure 1, PFVGH produces better results with larger k values, until a particular value.

The experimental results are shown in Table 3, on 10 different problems with a variety of dimensions from TSPLIB. The first column contains names of the problems and their dimensions. Second column contains the optimum tour length and third column contains the best result achieved with PFVGH and associated k values. Likewise, fourth column contains the worst results and associated k values. Last 2 columns contain the results achieved for k = 0 and k = n.

Table 3. The results PFVGH produces on 10 different problems

Problem	**Optimum Result**	**Best Result**	**Worst Result**	**Result for k = 0**	**Result for k = n**
ulysses16	74.108	75.041	88.923	88.923	75.138
		k=9	k=0		
ulysses22	75.665	77.494	89.436	89.436	78.675
		k=12,13	k=0		
bayg29	9074.148	9186.771	10372.301	9886.208	9196.496
		k=14-22	k=13		
dantzig42	699	746.860	843.737	843.737	784.908
		k=22	k=0		
att48	33523.708	36325.328	40434.100	38849.621	36325.328
		k=41-48	k=11		
eil51	429.983	440.746	519.591	481.518	440.746
		k=46-51	k=5,6,8		
berlin52	7544.365	8201.316	9954.062	9954.062	8618.198
		k=19	k=0		
st70	678.597	727.778	779.968	746.044	727.778
		k=64-70	k=12		
eil76	545.387	581.407	666.441	617.131	581.407
		k=60-76	k=3		
pr76	108159.438	116288.923	147438.691	127897.984	117569.531
		k=14	k=1		

As seen in Table 3, the better results are always achieved for ($0 < k < n$).

Similarly, Figure 2 shows the results PFVGH produces for the problems ulysses 16, ulysses 22, bayg29, att48, pr76, depending on the parameter k.

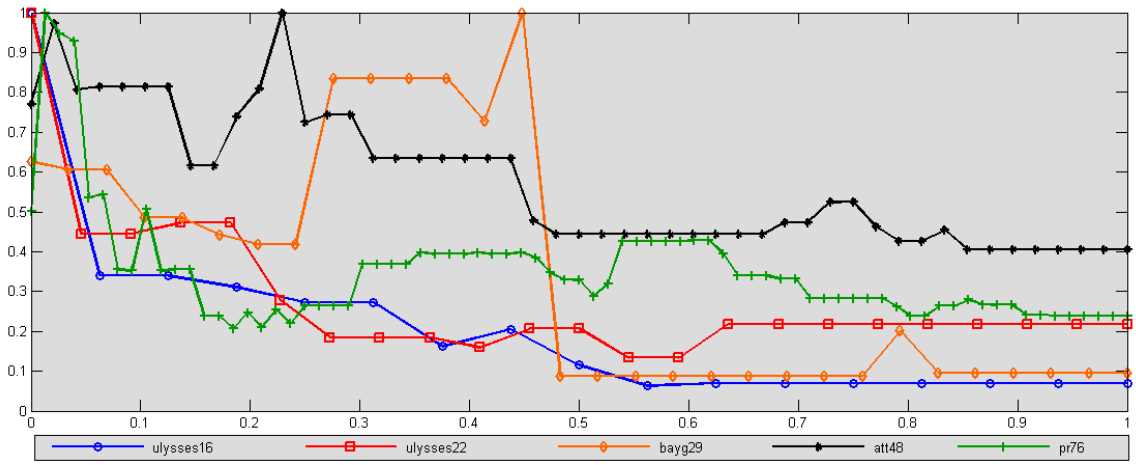

Figure 2. Correlation between the parameter k and the results PFVGH produces for the problems ulysses 16, ulysses 22, bayg29, att48, pr76.

As it is seen in Figure 2, PFVGH produces better results with larger k values, until a particular value for different problems with a variety of dimensions.

For validating the efficiency of the PFVGH algorithm, experimental results conducted on 22 different problems with a variety of dimensions are given in the next section. Different values of the parameter k, like [0.3*n], [0.5*n], [0.8*n] and [0.9*n] were chosen to observe the behavior of the algorithm and the result are presented along with the results of the NN and the Greedy algorithms.

4. EXPERIMENTAL RESULTS

This section presents the results of the computational experiments for the proposed hyper heuristic algorithm. The sample problems used in these experiments are taken from the [17]. And the optimum solutions for each of these problems are taken from the [18].

Table 4 shows the length of the optimal tour and the tours produced by different heuristic algorithms and the new hybrid algorithm. In Table 4, filled cells show the best results achieved by heuristics.

Table 4. The experimental results

Problems	Optimal	NN	Greedy	k=[0.3*n]	k=[0.5*n]	k=[0.8*n]	k=[0.9*n]
ulysses16	74.108	78.127	88.923	78.147	75.041	75.138	75.138
ulysses22	75.665	86.906	89.436	78.193	77.494	78.823	78.823
bayg29	9074.148	9964.781	9886.208	10159.256	9186.771	9196.495	9196.495
dantzig42	699	822.095	843.737	839.244	746.860	771.017	775.506
att48	33523.708	39236.885	38849.621	37908.551	36586.350	36474.922	36325.329
eil51	429.983	505.774	481.518	488.118	449.856	466.334	458.462
st70	678.597	761.689	746.044	742.822	753.461	732.507	727.778
eil76	545.387	612.656	617.131	619.172	603.460	581.839	581.839
pr76	108159.438	130921.005	127897.984	122592.863	119428.967	117493.719	117639.821
gr96	512.309	603.302	580.101	589.769	567.027	548.430	549.956
kroA100	21236.951	24698.497	24197.285	23735.891	23748.697	24799.237	24129.596
kroB100	22141	25882.973	25815.214	24383.732	23880.966	23459.891	23506.210
kroC100	20750.762	23566.403	25313.671	25572.433	23259.258	23384.664	23384.664
kroD100	21294.290	24855.799	24631.533	27771.079	24221.925	24101.979	24613.012
eil101	642.309	736.368	789.112	767.548	742.973	712.200	707.324
gr120	1666.508	1850.263	1915.918	1765.153	1718.043	1768.585	1708.057
ch130	6110.860	7198.741	7142.045	6776.120	6757.945	6705.272	6788.495
pr136	96772	114560.902	119553.703	105947.888	104309.555	107841.922	105814.516
kroA150	26524	31482.020	31442.994	29832.483	30125.133	29940.798	30053.487
kroB150	26130	31320.340	31519.083	30418.430	30725.288	29736.184	29780.122
rat195	2323	2628.561	2957.176	2680.773	2570.351	2578.752	2534.335
kroA200	29368	34547.691	37650.812	35426.901	33938.628	33754.659	33476.067

As seen in Table 4, PFVGH outperforms the NN algorithm and the Greedy algorithm in any instance.

5. CONCLUSION

In this work, we proposed a hyper heuristic that uses the previous algorithms we developed [11-13] and some well-known heuristics. The implementations of the algorithms were coded in C++ language. Then, we conducted computational experiments with the new algorithm and the other heuristics on a variety of problems. These experiments showed that the new algorithm outperforms the other heuristics.

6. REFERENCES

1. D. L. Appligate, R. E. Bixby, V. Chavatal and W. J. Cook, The Traveling Salesman Problem, *A Computational Study, Princeton Univesity Press*, Princeton and Oxford, 2006.
2. D. Davendra, Traveling Salesman Problem, Theory and Applications, In Tech, 2010.
3. G. Gutin, A. Punnen (eds.), The Traveling Salesman Problem and Its Variations, *volume 12 of Combinatorial Optimization*, Kluwer, Dordrecht, 2002.
4. L. J. Hubert, F. B. Baker, Applications of Combinatorial Programming to Data Analysis: The Traveling Salesman and Related Problems, *Psychometrika* **43(1),** 81-91, 1978.
5. D. Johnson, C. Papadimitriou (1985b), *Performance guarantees for heuristics*, In Lawler et al, chapter 5, 145-180, 1985.
6. D. S. Johnson and L. A. McGeoch, *The Traveling Salesman Problem: A Case Study*, *Local Search in Combinatorial Optimization*, 215-310, John Wiley & Sons, 1997.
7. E. L. Lawler, J. K. Lenstra, A. H. G. Rinnoy Kan, D. B. Shmoys, *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, John Wiley & Sons, 1986.
8. J. Lenstra, A. R. Kan, Some simple applications of the traveling salesman problem, *Operational Research Quarterly* **26(4),** 717-733, 1975.
9. J. K. Lenstra, Clustering a Data Array and the Traveling-Salesman Problem, *Operations Research* **22(2),** 413-414, 1974.
10. S. Lin, B. Kernighan, An effective heuristic algorithm for the traveling-salesman problem, *Operations Research* **21(2),** 498-516, 1973.
11. F. Nuriyeva, New heuristic algorithms for traveling salesman problem, *25th Conference of European Chapter on Combinatorial Optimization*, (ECCO XXV), Antalya, Turkey, April 26 – 28, 2012.
12. F. Nuriyeva, G. Kizilates, M. E. Berberler, Experimental Analysis of New Heuristics for the TSP, *IV International Conference "Problems of Cybernetics and Informatics"* Baku, Azerbaijan, September 12 – 14, 2012.
13. F. Nuriyeva, G. Kızılateş, A New Hyperheuristic Algorithm for Solving Traveling Salesman Problem, *2nd World Conference on Soft Computing*, p. 528-531 Baku, Azerbaijan, December 3-5, 2012.
14. A. Punnen, The *Traveling Salesman Problem: Applications, Formulations and Variations*, In Gutin and Punnen (2002), chapter 1, p. 1-28, 2002.
15. G. Reinelt, *The Traveling Salesman: Computational Solutions for TSP Applications*, Springer-Verlag, Germany, 1994.
16. D. J. Rosenkrantz, R. E. Stearns, P. M. Lewis, An Analysis of Several Heuristics for the Traveling Salesman Problem, *SIAM Journal on Computing* **6**, 563-581 (6.1).
17. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/
18. http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/