An Approximation Algorithm for the Asymmetric Traveling Salesman Problem

Janez Brest*
Janez Žerovnik **

Abstract: *An approximation algorithm for solving the asymmetric traveling salesman problem based on Arbitrary Insertion Algorithm is tested on all asymmetric instances from the TSPLIB. Surprisingly, the algorithm performs remarkably well both in terms of solution quality and computation time.*

Keywords *asymmetric traveling salesman problem, heuristic, randomize.*

Sommario: *Si effettuano test su un algoritmo approssimato per il TSP asimmetrico, usando tutti gli esempi di TSP asimmetrico presenti nella TSPLIB. Sorprendentemente, l'algoritmo ha prestazioni rimarchevoli sia per la qualità della soluzione che per il tempo di calcolo.*

♮ Revised: June 1, 1999.

* Faculty of Electrical Engineering and Computer Science, University of Maribor, Slovenia.

** Faculty of Mechanical Engineering, University of Maribor, Slovenia.
Also at Institute of Mathematics, Physics and Mechanics, University of Ljubljana, Slovenia

1. Introduction

The traveling–salesman problem (TSP) is one of the most studied problems in combinatorial optimization [8], [6]. The TSP is simply stated, has practical applications, and is a representative of a large class of important scientific and engineering problems. There is given a list of n cities and distances between them; in which order should the cities be visited so that the tour is as short as possible? Each city should be visited once before returning to the starting point.

The TSP can be viewed as a graph–theory problem if the cities are identified with the vertices of a graph, and the links between the cities are associated with arcs. A weight corresponding to the inter–city distance is assigned to each arc. The TSP is equivalent to finding a minimal weighted Hamiltonian circuit in the complete graph K_n. However, in its usual physical interpretation, where the vertices of a graph are cities and edges represent roads interconnecting them, the graph is most likely not complete. To remedy this situation, graph is usually completed by adding arcs with the cost of the shortest path in the original graph.

An instance of the TSP is given by distance matrix $D = (d_{ij})$ of dimension $n \times n$; where d_{ij} represents the weight of the arc from city i to city j in $N = \{1, ..., n\}$. If $d_{ij} = d_{ji}$ for every pair i and j in N then the TSP is *symmetric*, otherwise it is *asymmetric* (ATSP).

TSP is an example of a NP-hard problem. It is therefore reasonable to design approximate algorithms which find near–optimal solutions. According to [1], several hundreds of papers were published on TSP and probably every approach for attacking NP-hard optimization problems has also been tested or has even been formulated for TSP.

Recently, very good results were obtained with an exact algorithm for ATSP on a class of random instances [7]. Studies of the asymmetric traveling salesman polytope give hope to solve large instances of ATSP in general (see, for example [2] or [3]. The exact algorithms tend to be very time consuming, because their time complexity is superpolynomial. An alternative, perhaps more practical approach, is to design approximate algorithms which give solutions of reasonable quality in a short time.

In this paper we study a randomized approximate algorithm for ATSP. Our heuristics is based on the well-known arbitrary insertion procedure [10]. This algorithm was not payed too much attention, maybe because of the known worst case performance [10], [4].* However, because of our relatively good experience with insertion–and–optimization approach on PTSP [9], a probabilistic generalization of TSP [5], we started our experiment, in which we have repeatedly run the arbitrary insertion based procedure followed by a local optimization phase. Interestingly, we got surprisingly good results within short computation times. In this note we give the results of tests of our algorithm on all ATSP instances of the TSPLIB library [8], which were available at the time of the experiment.

The rest of the paper is organized as follows. In the next section our heuristic is described. Computation results and conclusions are given in the last section.

2. The Heuristic

The main idea of our heuristic is based on the *arbitrary insertion algorithm* [10], a relaxation of the cheapest insertion algorithm. The solutions are further improved by a local optimization phase.

Algorithm RAI (Randomized Arbitrary Insertion):

1 Start with a tour consisting of a given vertex and self–loop.
2 Randomly choose a vertex not on the tour.
3 Insert this vertex between neighboring vertex on the tour in the cheapest possible way. If the tour is still incomplete, go to step 2.
4 Keep this tour solution, say S.
5 Repeat n^2-times steps 6 through 10.

* In the worst case, the arbitrary insertion on ATSP can give solutions with costs as much as n times the optimal [4], while for the symmetric case the solution is always better than two times the optimum [10].

6 Randomly choose i and j ($i, j \in N = \{1, \ldots, n\}$, $1 \leq i \leq j \leq n$).
7 From the circuit with all vertices remove a path beginning with vertex i through vertex j, and connect vertex $i-1$ with vertex $j+1$.
8 Randomly choose a vertex from the removed path.
9 Insert this vertex between two neighboring vertices on the tour in the cheapest possible way. If the tour is still incomplete go to step 8.
10 Compare current solution with the solution S. Keep the better one.

First four steps generate an initial circuit. In the main loop – steps 6 through 10 an optimization is performed. Some vertices are removed from the circuit and later they are randomly reinserted into the circuit once more in the cheapest possible way.

There is an interesting question on how many times the optimization should be performed (step 5). We repeated it n^2-times. We have no theoretical arguments for this choice; it turned out to give good results in reasonably short time. In each iteration, the number of deleted and reinserted vertices is at most n and for each insertion of a vertex at most n different insertion positions are compared. The worst case time complexity of our algorithm is thus $O(n^4)$.

3. Computational Results

Since we needed optimal solution value for evaluation of the results, we tested the algorithm on all ATSP instances from TSPLIB library [8]. The results obtained in our experiments are shown in Table 1. The second and the third column indicate the name of ATSP instance and the number of cities, respectively. The optimal tours are known for all of problem instances and their costs are shown in the fourth column. For each of the problem instances, 50 independent runs were performed.

In Table [1] there is another measure of complexity, the number of insertions. Recall that in each run n^2 tours are generated. In the next column the average time (in seconds) of 50 independent runs is shown (PC-i586, 166MHz, Linux). In the last two columns the shortest and

	Problem	n	Optimum	#Inserts	$t[s]$	min	%	avg	%
1	br17	17	39	400	0.007	39	100.00	39.00	100.00
2	ftv33	34	1286	2271	0.098	1286	100.00	1291.38	100.42
3	ftv35	36	1473	2553	0.508	1473	100.00	1482.12	100.62
4	ftv38	39	1530	3142	0.674	1530	100.00	1539.94	100.65
5	p43	43	5620	4046	0.997	5620	100.00	5620.76	100.01
6	ftv44	45	1613	4550	1.198	1613	100.00	1638.16	101.56
7	ftv47	48	1776	5400	1.536	1776	100.00	1780.00	100.23
8	ry48p	48	14422	5520	1.598	14422	100.00	14512.50	100.63
9	ft53	53	6905	6724	2.398	6905	100.00	6941.12	100.52
10	ftv55	56	1608	7845	2.878	1608	100.00	1621.16	100.82
11	ftv64	65	1839	11348	5.241	1839	100.00	1858.12	101.04
12	ft70	70	38673	13568	7.068	38806	100.34	39147.60	100.88
13	ftv70	71	1950	14242	7.376	1950	100.00	1965.84	101.23
14	ftv90	91	1579	27106	20.59	1579	100.00	1583.30	100.27
15	kro124p	100	36230	33431	30.34	36241	100.03	37128.80	102.45
16	ftv100	101	1788	33327	31.28	1788	100.00	1791.68	100.21
17	ftv110	111	1958	43004	46.09	1958	100.00	1963.68	100.29
18	ftv120	121	2166	53138	65.34	2166	100.00	2175.58	100.44
19	ftv130	131	2307	42493	90.53	2307	100.00	2325.38	100.80
20	ftv140	141	2420	67112	124.9	2420	100.00	2432.60	100.52
21	ftv150	151	2611	93075	164.4	2611	100.00	2656.80	101.75
22	ftv160	161	2683	110556	216.2	2683	100.00	2725.42	101.58
23	ftv170	171	2755	123089	276.1	2755	100.00	2802.88	101.74
24	rbg323	323	1326	623454	3874	1342	101.21	1354.54	102.15
25	rbg358	358	1163	810019	6825	1167	100.34	1174.25	100.97
26	rbg403	403	2465	1094209	11137	2465	100.00	2465.95	100.04
27	rbg443	443	2720	1383453	17126	2720	100.00	2720.37	100.01

Table 1 - The results for the asymmetric TSP instances.

the average solution length for the 50 runs are shown, respectively. For all but four instances (problems: ft70, kroa124p, rbg323, rbg358) the optimal solutions have been found. If we look at the averages of the 50 independent runs we can see that our heuristic has solved the problems within 3% from optimum on average. If we look at the best solutions, we can see that our heuristic has solved majority of problems within 0.5% from optimum. there was only one exception: the best solution obtained for the problem instance rbg323 was 101.21% from optimum.

Figures 1 and 2 show how the quality of the solutions (minimum, maximum and average) is improving with time on two particular in-

stances.

We have also compared our heuristic with Farthest Insertion (far), and Farthest Insertion followed by OR-opt [6] (p. 220). OR-opt local optimization proceeds as follows. For each connected string of s cities (s equals 3 first, then 2, then 1), we test to see if the string can be relocated between two other cities at reduced cost. If it can, we make the appropriate changes. After considering all strings of three cities, all strings of two cities and then all strings of one city are considered. When no further exchanges improve the solution, the algorithm terminates.

The results are given in Table 2. Farthest insertion was repeated n times, each time another vertex was used as initial tour. The running times for farthest insertion are much shorter, therefore we only give the shortest solution length. The tours constructed by farthest insertion were then improved by OR–opt. The solutions after local optimization are given in column far+OR. The execution times were measured and then the algorithm RAI was let running for the same amount of time (column $t[s]$). The solutions obtained by RAI are given in column RAI. Note that the solutions obtained by RAI algorithm were usually better with only three exceptions, the instances ry48p, rbg323, and rbg358.

We conclude that the fast and simple heuristics RAI performs remarkably well and is competitive both in terms of solution quality and execution times with the best heuristics proposed in the literature.

Acknowledgement

The authors wish to thank the referees for constructive criticism. In particular, one of the referees suggested additional comparison with farthest insertion.

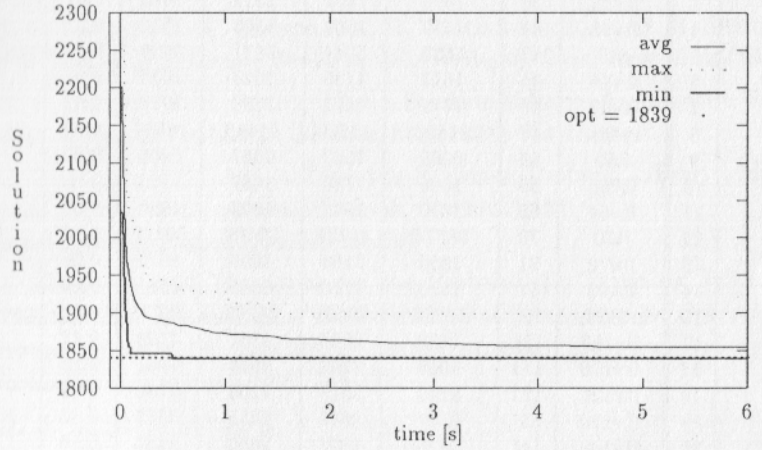

Figure 1 - Solution vs. time of ftv64 instance

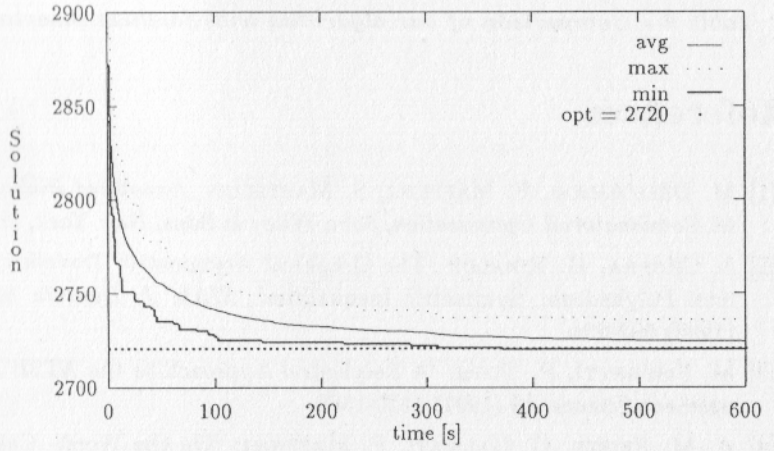

Figure 2 - Solution vs. time of rbg443 instance

	Problem	n	Optimum	far	far+OR	RAI	$t[s]$
1	br17	17	39	39	39	39	0.00
2	ftv33	34	1286	1298	1286	1286	0.01
3	ftv35	36	1473	1512	1512	1473	0.01
4	ftv38	39	1530	1569	1569	1530	0.01
5	p43	43	5620	5644	5631	5620	0.01
6	ftv44	45	1613	1786	1623	1613	0.01
7	ftv47	48	1776	1867	1784	1777	0.02
8	ry48p	48	14422	15239	14446	14507	0.02
9	ft53	53	6905	7567	7087	6905	0.02
10	ftv55	56	1608	1728	1637	1623	0.03
11	ftv64	65	1839	2043	1913	1850	0.05
12	ft70	70	38673	40739	39489	39179	0.06
13	ftv70	71	1950	2184	1986	1973	0.06
14	ftv90	91	1579	1770	1585	1581	0.13
15	kro124p	100	36230	40201	36771	36321	0.19
16	ftv100	101	1788	2064	1847	1796	0.17
17	ftv110	111	1958	2202	2026	1961	0.25
18	ftv120	121	2166	2513	2256	2186	0.32
19	ftv130	131	2307	2615	2455	2335	0.43
20	ftv140	141	2420	2822	2606	2436	0.56
21	ftv150	151	2611	3058	2735	2619	0.73
22	ftv160	161	2683	3130	2879	2711	0.91
23	ftv170	171	2755	3297	2957	2839	1.13
24	rbg323	323	1326	1672	1359	1392	10.75
25	rbg358	358	1163	1563	1219	1233	15.17
26	rbg403	403	2465	2720	2469	2469	22.62
27	rbg443	443	2720	3163	2733	2726	30.84

Table 2 - Comparison of our algorithm with farthest insertion.

References

[1] M. Dell'Amico, F. Maffioli, S. Martello: *Annotated Bibliography in Combinatorial Optimization,* John Wiley & Sons, New York, (1997).

[2] S. Chopra, G. Rinaldi: 'The Graphical Asymmetric Traveling Salesman Polyhedron: Symmetric Inequalities', *SIAM J. Discrete Math*, 9 (1996) 602-624.

[3] M. Fischetti, P. Toth: 'A Polyhedral Approach to the ATSP', *Management Science*, 43 (1997) 1520-1536.

[4] A. M. Frieze, G. Galbiati, F. Maffioli: 'On the Worst- Case Performance of some Algorithms for the Asymmetric Traveling Salesman Problem', *Networks*, 12 (1982) 23-39.

[5] P. Jaillet: 'A Priori Solution of a Traveling Salesman Problem in which a Random Subset of the Customers are Visited', *Operation Research*, 36 (1988) 929-936.

[6] E. L. Lawler, J. K. Lenstra, A. H. G. Rinnooy Kan, D. B. Shmoys: *The Traveling Salesman Problem*, John Wiley & Sons, New York (1985)

[7] D. L. Miller, J. F. Pekny: 'Exact Solution of Large Asymmetric Traveling Salesman Problems', *Science,* 251 (1991) 754-761.

[8] G. Reinelt: 'TSPLIB - a Traveling Salesman Problem Library', *European Journal of Operations Research,* 52 (1991) 125. (http://softlib.rice.edu/softlib/tsplib)

[9] J. Žerovnik: 'A Heuristics for the Probabilistic Traveling Salesman Problem', in: *Proceedings of the International Symposium on Operational research SOR'95* (V. Rupnik, M. Bogataj, eds.), Slovenian Society Informatika, Ljubljana (1995) 165-172.

[10] D. J. Rosenkrantz, R. R. Stearns, P. M. Lewis: 'An Analysis of Several Heuristics for the Traveling Salesman Problem', *SIAM J. Comput.*, 6 (1977) 563-581.