INDIAN INSTITUTE OF MANAGEMENT CALCUTTA

WORKING PAPER SERIES

WPS No. 667/ November 2010

Cross Entropy based Neighborhood Reduction and Initial Tour Formation for Tabu Search on the Asymmetric Traveling Salesman Problem

by

Sumanta Basu
Assistant Professor, IIM Calcutta, Diamond Harbour Road, Joka P.O., Kolkata 700 104 India

&

Diptesh Ghosh
Associate Professor, IIM Ahmedabad, Vastrapur, Ahmedabad Pin-380015, India

Cross Entropy based Neighborhood Reduction and Initial Tour Formation for Tabu Search on the Asymmetric Traveling Salesman Problem

Sumanta Basu Diptesh Ghosh[y]

Abstract

generated. At the end of the iteration, we update P as follows. Let n_{ij} be the number of times that arc (v_i, v_j) appears in the tours in E. Then $p_{ij} = n_{ij}/(e \cdot n)$. After *iter* iterations get over, a sparse graph $G' = (V, A')$ is formed where $A' = \{(v_i, v_j) : p_{ij} > 0\}$.

From preliminary experiments we observed that at the end of the specified number of iterations, the output graph became too sparse and as a result some of the tours did not have any neighboring tour. In such graphs, tabu search was unable to better the best tour in the E list after *iter* iterations. So motivated by Toth and Vigo (1998), we added a step in our algorithm in which we chose a threshold , and added those arcs in A whose costs were less than the threshold to A'. The result of this operation is a denser G' but one in which some neighboring tours do exist for most tours.

Based on preliminary experiments with randomly generated ATSP instances of sizes 100 and 250, we chose k, e, and *iter* as 50000, $15n$ and 20 respectively. We chose the threshold value as 1.5 times the average of the costs of arcs in E weighed by the frequency of their appearance.

3 Generation of Initial Tours

Performance of any local search heuristic like tabu search is often critically dependent on the quality of the solution used as a starting point for the algorithm. As the problem size increases, the solution space increases exponentially, and the choice of initial solutions becomes increasingly important. Tabu search is implemented for large sized problems in the multi-start mode. In this mode, tabu search is started from several initial tours which are widely separated in the solution space. The best tour obtained in all the runs is output by the algorithm. The preprocessing algorithm described in Section 2 can be easily tweaked to generate initial tours. To do this, the required number of initial tours are obtained from the tours present in the elite set E at the end of the preprocessing algorithm.

4 Computational Experience

In this section, we first describe the tabu search implementations that we create to test our preprocessing and initial tour generation schemes. We then report the experiment design and test beds of problems that we use for our experiments. Finally we present the results of our computational experiments.

Tabu search implementations: We combined our preprocessing method and initial tour generation method into three tabu search implementations, labeled A through C. Each implementation is defined as a combination of the preprocessing method used, method used to generate initial tours, and the implementation of tabu search used. Two implementations of tabu search described in Basu et al. (2008) were considered; the TS-CI implementation which is the conventional implementation designed for tabu search on instances de ned on complete graphs, and the TS-SAG implementation designed for tabu search on instances de ned on sparse asymmetric graphs. In conventional implementations, non-existent arcs in a graph are represented as in nite cost arcs. Hence even though the neighborhood of a tour is much smaller for an ATSP instance de ned on a sparse graph than one de ned on a complete graph with the same number of nodes, conventional implementations of tabu search actually search a complete graph in the both cases. TS-SAG uses special data structures to eliminate the need for in nite cost arcs and so tabu search actually searches a much smaller neighborhood. This speeds up the TS-SAG algorithm signi cantly compared to TS-CI on ATSPs de ned on sparse graphs. Table 1 describes the implementations that we use for our computational experiments. All the implementations

Table 1: Details of implementations

Implementation Scheme	Preprocessing Scheme	Initial Solution	Tabu Search Implementation
A	None	Generated randomly	TS-CI
B	None	From Section 3	TS-CI
C	From Section 2	From Section 3	TS-SAG

were coded in C, were run on a computer with an Intel Quad Core 2.4GHz processor and 3 GB of RAM. The length of the tabu list in all tabu search implementations was xed at 8.

Comparisons between di erent implementations allow us to comment on the usefulness of the preprocessing method and initial tour generation method. A comparison of qualities of the tours obtained by implementations A and B shows us the e ectiveness of the use of special methods to generate initial tours for multi-start tabu search. A comparison between implementations B and C shows us the usefulness of the preprocessing scheme combined with the use of special tabu search implementation designed for ATSPs de ned on sparse graphs.

Test beds and computational experiments: We performed our experiments on randomly generated ATSP instances as well as on benchmark ATSP instances. Each of the instances was taken as described on a complete digraph. The randomly generated instances consisted of ten problem instances each of size 200, 300, 400, 500, and 600. The arc costs were chosen as integers randomly in the interval [1,1000]. The benchmark instances consisted of 25 ATSP instances from Johnson et al. (2002) with 100 nodes

Table 3: Execution time in seconds required by the three implementations to complete 1000 tabu search iterations

		200	300	400	500	600
A	Avg. preproc. time	0.00	0.00	0.00	0.00	0.00
	Avg. of time for TS	43.16	127.94	277.31	497.39	821.05
	Avg. of total time	43.16	127.94	277.31	497.39	821.05
B	Avg. preproc. time	117.40	264.10	469.20	732.70	1052.50
	Avg. of time for TS	42.32	131.77	287.32	533.45	873.23
	Avg. of total time	159.72	395.87	756.52	1266.15	1925.73
C	Avg. preproc. time	117.40	264.10	469.20	732.70	1052.50
	Avg. of time for TS	3.14	7.44	12.83	19.69	29.18
	Avg. of total time	120.54	271.54	482.03	752.39	1081.68

were better than those output by implementation A. Implementation B produced slightly better tours than implementation C although the tour costs were not signi cantly di erent when tested at a signi cance level of 0.01 (in a paired-t test). From Table 3 we see that as expected, the time required by tabu search in implementation C is much less than that required by tabu search in implementations A and B. Overall, implementation C required less time than implementation B, with the di erence in the total time required increases with increasing problem size. Based on these observations, implementation C seems to dominate other implementations for randomly generated ATSP instances.

Table 4 summarizes the quality of tours output by the three implementations on the 25 benchmark ATSP instances. Since the costs of optimal tours for these problems are very di erent, we expressed the quality of tours output by the three implementations as a multiple of the Held-Karp lower bound (see Held and Karp, 1970, 1971) for that ATSP instance. Observe that implementations B and/or C produced the least cost tours in 19 out of the 25 instances. The four problems in the rgb class form a notable exception, implementation A generated the best tours to these problem instances. Table 5 presents the execution times required by the implementations on the benchmark problem instances. Between implementations B and C which provide the best quality tours in most cases, implementation C

Table 4: Costs of tours output by the implementations as a multiple of Held-Karp bound at the end of 1000 tabu search iterations

		Implementation		
Instance	Size	A	B	C
atex8	600	5.39	5.18	5.10
big702	702	5.22	5.30	5.36
dc112	112	1.01	1.01	1.01
dc126	126	1.00	1.00	1.03
dc134	134	1.01	1.01	1.02
dc176	176	1.02	1.01	1.03
dc188	188	1.01	1.01	1.02
dc563	563	1.05	1.05	1.08
dc849	849	1.05	1.05	1.05
dc895	895	1.02	1.02	1.13
dc932	932	1.01	1.01	1.11
ftv100	100	2.73	2.02	2.02
ftv110	110	2.78	2.16	2.29
ftv120	120	2.67	2.20	2.20
ftv130	130	3.11	2.25	2.25
ftv140	140	3.20	2.70	2.71
ftv150	150	3.34	2.27	2.45
ftv160	160	3.45	2.64	2.71
ftv170	170	3.64	2.89	2.85
kro124p	124	1.34	1.25	1.25
rbg323	323	2.86	3.16	3.16
rbg358	358	3.55	4.21	4.21
rbg403	403	2.11	2.48	2.48
rbg443	443	2.05	2.43	2.43
td100_1	100	1.32	1.12	1.12

the tours output by the di erent implementations over the ten randomly generated ATSP instances of a given size. The trends in the results from these experiments closely follow their counterparts in the rst set. Here too, implementations B and C produced the least cost tours output and the di erence in the quality of tours output by implementations B and C is not statistically signi cant.

For benchmark problem instances, we increased the time limit to ensure that tabu search could run from at least three initial tours for each of the instances. These allowable time limits were made proportional to the problem size. The execution times for the 25 instances are given in Table 7.

Table 8 reports the costs of tours output by the implementations as multiples of the Held-Karp bound for the corresponding problem. We see that implementations B and/or C produced the best tours in 21 of the 25 benchmark instances. Implementation A was seen to outperform implementations

Table 5: Execution time in seconds required by the three implementations

Table 7: Execution times for benchmark problems

Problem	Time Limit
atex8	2000
big702	3000
dc112	50
dc126	75
dc134	80
dc176	120
dc188	140
dc563	1500
dc849	4500
dc895	5500
dc932	7000
ftv100	50
ftv110	60
ftv120	70
ftv130	80
ftv140	90
ftv150	100
ftv160	110
ftv170	120
kro124p	50
rbg323	1000
rbg358	1100
rbg403	1200
rbg443	1300
td100_1	50

the TS-SAG implementation of tabu search produces results for moderate sized ATSP instances which are superior to conventional tabu search implementations for this problem.

5 Summary

In this paper, we suggest a tabu search implementation to solve asymmetric traveling salesman problems (ATSPs). In our implementation, we rst use a cross entropy method based preprocessing algorithm to reduce the density of the graph describing the problem instance, and to generate good starting tours for tabu search to operate in a multi-start mode. We then use a tabu search implementation specially designed to solve ATSPs de ned on sparse graphs.

We created three implementations, one without any of the enhancements

Table 8: Costs of tours output by the implementations as a multiple of Held-Karp bound at the end of the pre-speci ed execution time

		Implementation		
Instance	Size	A	B	C
atex8	600	5.51	5.25	5.05
big702	702	12.65	4.91	5.19
dc112	112	1.01	1.01	1.01
dc126	126	1.00	1.00	1.02
dc134	134	1.01	1.01	1.02
dc176	176	1.02	1.01	1.03
dc188	188	1.00	1.01	1.02
dc563	563	1.05	1.05	1.08
dc849	849	1.05	1.05	1.05
dc895	895	1.03	1.03	1.13
dc932	932	1.00	1.01	1.11
ftv100	100	2.65	1.90	1.90
ftv110	110	2.78	1.93	1.93
ftv120	120	2.67	2.16	2.16
ftv130	130	3.04	2.30	2.30
ftv140	140	3.20	2.42	2.52
ftv150	150	3.34	2.59	2.59
ftv160	160	3.45	2.78	2.68
ftv170	170	3.64	2.85	2.84
kro124p	124	1.34	1.25	1.25
rbg323	323	3.61	2.19	2.20
rbg358	358	859.84	2.72	2.78
rbg403	403	405.68	1.69	1.74
rbg443	443	367.65	1.74	1.79
td100_1	100	1.32	1.13	1.12

entropy method, followed by the TS-SAG tabu search implementation (see Basu et al., 2008) specially designed for performing tabu search on ATSPs de ned on sparse graphs.

References

Applegate D, Bixby R, Chvatal V and Cook W (2006). The Traveling Salesman Problem: A Computational Study. Princeton University Press.

Basu S, Gajulapalli R and Ghosh D (2008). Implementing tabu serach to exploit sparsity in atsp instances Working Paper Series, Indian Institute of Management Ahmedabad, 2008-10-02.

Basu S and Ghosh D (2008). A review of the tabu search literature on traveling salesman problems. Working Paper Series, Indian Institute of Management Ahmedabad, W.P. No. 2008-10-01.

Boer P, Kroese D, Mannor S and Rubinstein R (2005). A tutorial on the cross-entropy method. Annals of Operations Research 134: 19{67.

Chepuri K and Homem-de Mello T (2005). Solving the vehicle routing problem with stochastic demands using the cross entropy method. Annals of Operations Research 134: 193{181.

Cirassela J, Johnson D, McGeoch L and Zhang W (2001). The asymmetric traveling salesman problem: algorithms, instance generators, and tests. In: Buchsbaum A and Snoeyink J (eds). Algorithm Engineering and Experimentation. Third International Workshop, ALENEX 2001, Lecture Notes in Computer Science 2153, pp 32{59. Springer-Verlag.

Fischetti M, Lodi A and Toth P (2002). Exact methods for asymmetric traveling salesman problem. In: Gutin G and Punnen P (eds). The Traveling Salesman Problem and Its Variations 4, pp 169{206. Kluwer Academic Publisher: London.

Glover F and Laguna M (1998). Tabu Search. Kluwer Academic Publisher: London.

Goossens J and Baruah S (2001). Multiprocessor preprocessing algorithms for uniprocessor on-line scheduling. In: The 21th International Conference on Distributed Computing Systems.

Held M and Karp R (1970). The traveling salesman problem and minimum spanning trees. Operations Research 18: 1138{1162.

Held M and Karp R (1971). The traveling salesman problem and minimum spanning trees: Part II. Mathematical Programming 1: 6{25.

Johnson D, Gutin G, McGeoch L, Yeo A, Zhang W and Zverovich A (2002). Experimental analysis of heuristics for the ATSP. In: Gutin G and Punnen P (eds). The Traveling Salesman Problem and Its Variations 10, pp 445{488. Kluwer Academic Publishers: London.

Karp R (1972). Reducibility among combinatorial problems. Complexity of Computer Computations, pp 85{103. Plenum Press.

Khumawala B (1975). An e cient branch and bound algorithm for the warehouse location problem. Management Science 18: B718{B731.

Reinelt G (1991). TSPLIB { A traveling salesman problem library.

Rubinstein R (1997). Optimization of computer simulation models with rare events. European Journal of Operational Research99: 89{112.

Rubinstein R (1999). The simulated entropy method for combinatorial and continuous optimization. Methodology and Computing in Applied Probability 2: 127{190.

Rubinstein R (2001). Combinatorial Optimization, Cross-Entropy, Ants and Rare Events. In: Uryasev S and Pardalos P M (eds).Stochastic optimization: algorithms and applications, pp 445{488. Kluwer Academic Publishers: London.

Toth P and Vigo D (2003). The granular tabu search and its application to the vehicle-routing problem. INFORMS Journal on Computing 15: 333{346.