COMPREHENSIVE STUDY OF ANT COLONIES FOR THE TRAVELING SALESMAN PROBLEM AND COMPARE EFFICIENCY AND PERFORMANCE WITH BASIC ALGORITHM

Maulik V. Dhamecha *

Krishna H. Hingrajiya **

ABSTRACT

The travelling salesman problem (TSP) is an NP-hard problem in combinatorial optimization studied in operations research and theoretical computer science. Given a list of cities and their pair wise distances, the task is to find the shortest possible tour that visits each city exactly once.

It is used as a benchmark for many optimization methods. Even though the problem is computationally difficult, a large number of heuristics and exact methods are known, so that some instances with tens of thousands of cities can be solved.

The TSP has several applications even in its purest formulation, such as planning, logistics, and the manufacture of microchips.

The ant colony optimization is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. In ACO, a set of software agents called artificial ants search for good solutions to a give optimization problem. The traveling salesman problem (TSP) is the problem of finding a shortest closed tour which visits all the cities in a given set. In this article we will restrict attention to TSPs in which cities are on a plane and a path (edge) exists between each pair of cities (i.e., the TSP graph is completely connected). Ant colony optimization is a metaheuristic in which a colony of artificial ants cooperates in finding good solutions to difficult discrete optimization problems.

* Assistant Professor, Department of CE, School of Engineering, RK University, Rajkot, Gujarat, India.

** Assistant Professor, Department of CE, School of Engineering, RK University, Rajkot, Gujarat, India.

INTRODUCTION

Ant algorithms [18, 14, 19] are a recently developed, population-based approach which has been successfully applied to several NP-hard combinatorial optimization problems [6, 13, 17, 23, 34, 40, 49]. As the name suggests, ant algorithms have been inspired by the behavior of real ant colonies, in particular, by their foraging behavior. One of the main ideas of ant algorithms is the indirect communication of a colony of agents, called (artificial) ants, based on pheromone trails (pheromones are also used by real ants for communication). The (artificial) pheromone trails are a kind of distributed numeric information which is modified by the ants to react their experience while solving a particular problem. Recently, the Ant Colony Optimization (ACO) met heuristic has been proposed which provides a unifying framework for most applications of ant algorithms [15, 16] to combinatorial optimization problems. In particular, all the ant algorithms applied to the TSP ¯t perfectly into the ACO meta-heuristic and, therefore, we will call these algorithms also ACO algorithms.

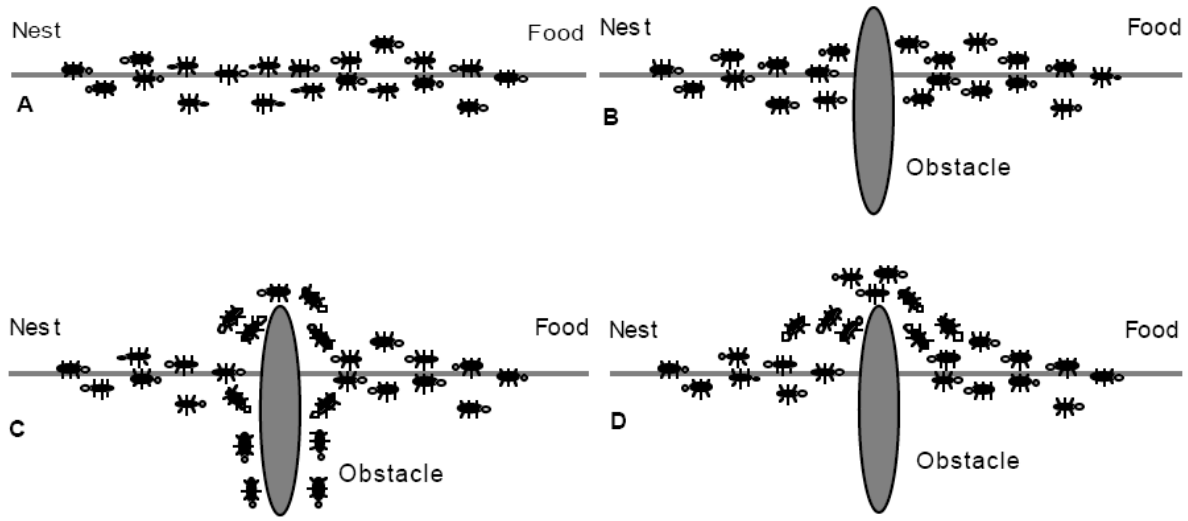


Fig.1. (A) Real ants follow a path between nest and food source. (B) An obstacle appears on the path: Ants choose whether to turn left or right with equal probability. (C) Pheromone is deposited more quickly on the shorter path. (D) All ants have chosen the shorter path.

The first ACO algorithm, called Ant System (AS) [18, 14, 19], has been applied to the Traveling Salesman Problem (TSP). Starting from Ant System, several improvements of the basic algorithm have been proposed [21, 22, 17, 51, 53, 7]. Typically, these improved algorithms have been tested again on the TSP. All these improved versions of AS have in common a stronger exploitation of the best solutions found to direct the ants' search process; they mainly differ in some aspects of the search control. Additionally, the best performing

ACO algorithms for the TSP [17, 49] improve the solutions generated by the ants using local search algorithms.

In this paper we give an overview on the available ACO algorithms for the TSP. We first introduce, in Section 1.2, the TSP. In Section 1.3 we outline how ACO algorithms can be applied to that problem and present the available ACO algorithms for the TSP. Section 1.4 briefly discusses local search for the TSP, while Section 1.5 presents experimental results which have been obtained with MAX{MIN Ant System, one of the improved versions of Ant System. Since the first application of ACO algorithms to the TSP, they have been applied to several other combinatorial optimization problems. On many important problems ACO algorithms have proved to be among the best available algorithms. In Section 1.6 we give a concise overview of these other applications of ACO algorithms.

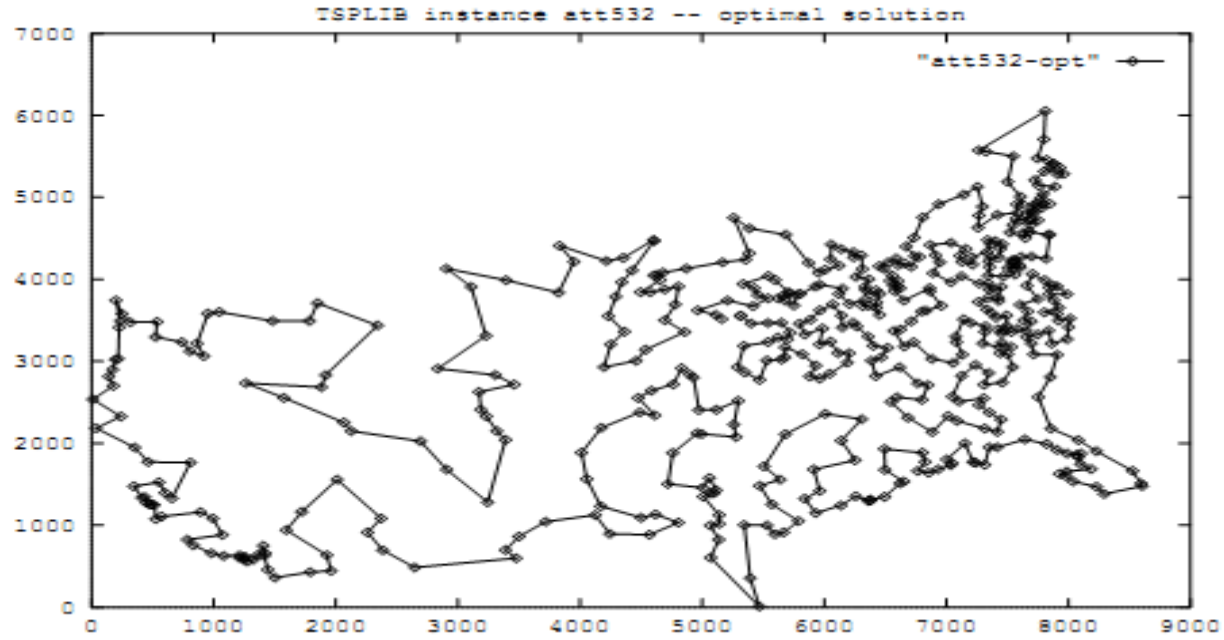
The traveling salesman problem (TSP) is the problem of finding a shortest closed tour which visits all the cities in a given set. In this article we will restrict attention to TSPs in which cities are on a plane and a path (edge) exists between each pair of cities (i.e., the TSP graph is completely connected).

THE TRAVELING SALESMAN PROBLEM

The TSP is extensively studied in literature [29, 31, 45] and has attracted since a long time a considerable amount of research effort. The TSP also plays an important role in Ant Colony Optimization since the first ACO algorithm, called Ant System [18, 14, 19], as well as many of the subsequently proposed ACO algorithms [21, 17, 52, 53, 7] have initially been applied to the TSP. The TSP was chosen for many reasons: (i) it is a problem to which ACO algorithms are easily applied, (ii) it is an NP-hard [26] optimization problem, (iii)it is a standard test-bed for new algorithmic ideas and a good performance on the TSP is often taken as a proof of their usefulness, and (iv) it is easily understandable, so that the algorithm behavior is not obscured by too many technicalities.

Intuitively, the TSP is the problem of a salesman who wants to ¯nd, starting from his home town, a shortest possible trip through a given set of customer cities and to return to its home town. More formally, it can be represented by a complete weighted graph G =(N;A) with N being the set of nodes, representing the cities, and A the set of arcs fully connecting the nodes N. Each arc is assigned a value dij , which is the length of arc (i; j) 2 A, that is, the distance between cities i and j, with i; j 2 N. The TSP is the problem of finding a minimal length Hamiltonian circuit of the graph, where an Hamiltonian circuit is a closed tour visiting exactly once each of the n = jNj nodes of G. For symmetric TSPs, the distances between the

cities are independent of the direction of traversing the arcs, that is, dij = dji for every pair of nodes. In the more general asymmetric TSP (ATSP) at least for one pair of nodes i; j we have dij = dji. In case of symmetric TSPs, we will use Euclidean TSP instances in which the cities are points in the Euclidean space and the inter-city distances are calculated using the Euclidean norm. All the TSP instances we use are taken from the TSPLIB Benchmark library [44] which contains a large collection of instances; these have been used in many other studies or stem from practical applications of the TSP.



LOCAL SEARCH FOR THE TSP

Local Search starts from some initial assignment and repeatedly tries to improve the current assignment by local changes. If in the neighborhood of the current tour T a better tour T0 is found, it replaces the current tour and the local search is continued from T0. The most widely known iterative improvement algorithms for the TSP are certainly 2-opt [12] and 3-opt [32]. They proceed by systematically testing whether the current tour can be improved by replacing 2 or at most 3 arcs, respectively. Both local search algorithms are widely studied in the literature [45, 29] and have been shown empirically to yield good solution quality. Local search algorithms using k> 3 arcs to be exchanged are not used commonly, due to the high computation times involved and the low additional increase in solution quality. Already for 2-opt and 3-opt a straightforward implementation would require O(n2)or O(n3) exchanges to be examined. This is clearly infeasible when trying to solve instances with several hundreds of cities in reasonable computation time.

Fortunately, there exist quite a few speed-up techniques [1, 45, 29] achieving, in practice, run-times which grow sub quadratic ally. This effect is obtained by examining only a small part of the whole neighborhood. We use three techniques to reduce the run-time of 2-opt and 3-opt implementations. One, consists in restricting the set of moves which are examined to those contained in a candidate list of the nearest neighbors ordered according to no decreasing distances [1, 33, 45]. Using candidate lists, for a given starting node i we only consider moves which add a new arc between i and one of the nodes in its candidate list. Hence, by using a neighborhood list of bounded length, an improving move can be found in constant time.

An additional speed-up is achieved by performing a fixed radius nearest neighbor search [1]. For 2-opt at least one newly introduced arc has to be shorter than any of the two removed arcs (i; j) and (k; l). Due to symmetry reasons, it success to check whether dij >dik. A similar argument also holds for 3-opt [1].

To yield further reductions in run-time we use don't look bits associated with each ode. Initially, all don't look bits are turned o® (set to 0). If for a node no improving move can be found, the don't look bit is turned on (set to 1) and the node is not considered as a starting node for finding an improving move in the next iteration. In case an arc incident to a node is changed by a move, the node's don't look bit is turned o® again. For asymmetric TSPs, 2-opt is not directly applicable because one part of the tour has to be traversed in the opposite direction. In case a sub-tour is reversed, the length of this sub-tour has to be computed from scratch. Yet, one of the three possible 3-opt moves does not lead to a reversal of a sub-tour. We call reduced 3-opt this special form of 3-opt. The implementation of reduced 3-opt uses the same speed-up techniques as described before. The local search algorithm producing the highest quality solutions for symmetric TSPs is the Lin-Kernighan heuristic (LK) [33]. The Lin-Kernighan

EXPERIMENTAL RESULTS FOR SYMMETRIC TSPS

In this section we report on experimental results obtained with MMAS, on some symmetric TSP instances from TSPLIB. Most of the instances were proposed as benchmark instances in the First International Contest on Evolutionary Computation (1st ICEO) [2]. We compare the computational results obtained with MMAS to a standard iterated local search algorithm [38, 37, 29] for the TSP using the same 3-opt implementation and the same maximal computation time tmax for each trial as MMAS.

Iterated local search (ILS) [38, 37, 29] is well known to be among the best algorithms for the TSP. In ILS a local search is applied iteratively to starting solutions which are obtained by mutations of a previously found local optimal solution, typically the best solution found so far. In particular, the ILS algorithm we use for comparison first locally optimizes an initial tour produced by the nearest neighbor heuristic. Then it applies iteratively 3-opt to initial solutions which are obtained by the application of random, sequential 4-opt moves (called double-bridge moves in [38]) to the best solution found so far. The runs are performed on a Sun UltraSparc II Workstation with two UltraSparc I 167MHz processors with 0.5MB external cache. Due to the sequential implementation of the algorithm only one processor is used. The computational results show that MMAS is able to ¯nd very high quality solutions for all instances and on most instances MMAS achieves a better average solution quality than ILS. This result is very encouraging, since ILS is cited in the literature as a very good algorithm for the TSP [37, 29].

Table 1 Comparison of MMAS with iterated 3-opt (ILS) applied to some symmetric TSP instances (available in TSPLIB). Given are the instance name (the number in the name gives the problem dimension, that is, the number of cities), the algorithm used, the best solution, the average solution quality (its percentage deviation from the optimum in parentheses), the worst solution generated, the average time tavg to ¯nd the best solution in a run, and the maximally allowed computation time tmax. Averages are taken over 25 trials for n< 1000, over 10 trials on the larger instances. Best average results are in boldface.

Instance	Algorithm	Best	Average	Worst	t_{avg}	t_{max}
d198	MMAS	15780	15780.4 (0.0)	15784	61.7	170
	ILS	15780	**15780.2 (0.0)**	15784	18.6	
lin318	MMAS	42029	**42029.0 (0.0)**	42029	94.2	450
	ILS	42029	42064.6 (0.09)	42163	55.6	
pcb442	MMAS	50778	**50911.2 (0.26)**	51047	308.9	600
	ILS	50778	50917.7 (0.28)	51054	180.7	
att532	MMAS	27686	**27707.9 (0.08)**	27756	387.3	1250
	ILS	27686	27709.7 (0.09)	27759	436.2	
rat783	MMAS	8806	**8814.4 (0.10)**	8837	965.2	2100
	ILS	8806	8828.4 (0.34)	8850	1395.2	
u1060	MMAS	224455	224853.5 (0.34)	225131	2577.6	3600
	ILS	224152	**224408.4 (0.14)**	224743	1210.4	
pcb1173	MMAS	56896	**56956.0 (0.11)**	57120	3219.5	5400
	ILS	56897	57029.5 (0.24)	57251	1892.0	
d1291	MMAS	50801	**50821.6 (0.04)**	50838	1894.4	5400
	ILS	50825	50875.7 (0.15)	50926	1102.9	
fl1577	MMAS	22286	**22311.0 (0.28)**	22358	3001.8	7200
	ILS	22311.0	22394.6 (0.65)	22505	3447.6	

Table 1.2 Comparison of MMAS with iterated 3-opt (ILS) applied to some asymmetric TSP instances (available in TSPLIB). Given are the instance name (the number in the name gives the problem dimension, that is, the number of cities; an exception is instance kro124p with 100 cities), the algorithm used, the best solution, the average solution quality (its percentage deviation from the optimum in parentheses), the worst solution generated, the average time tavg to ¯nd the best solution in a run, and the maximally allowed computation time tmax. Averages are taken at least over 25 trials. Best average results are in boldface.

Instance	Algorithm	Best	Average	Worst	t_{avg}	t_{max}
ry48p	$\mathcal{MM}$AS	14422	**14422.0 (0.0)**	14422	2.3	120
	ILS	14422	14425.6 (0.03)	14507	24.6	
ft70	$\mathcal{MM}$AS	38673	**38673.0 (0.0)**	38673	37.2	300
	ILS	38673	38687.9 (0.04)	38707	3.1	
kro124p	$\mathcal{MM}$AS	36230	**36230.0 (0.0)**	36230	7.3	300
	ILS	36230	36542.5 (0.94)	37114	23.4	
ftv170	$\mathcal{MM}$AS	2755	**2755.0 (0.0)**	2755	56.2	600
	ILS	2755	2756.8 (0.07)	2764	21.7	

EXPERIMENTAL RESULTS FOR ASYMMETRIC TSPS

We applied the same algorithms also to the more general asymmetric TSP; the computational results are given in Table 1.2. On the asymmetric instances the performance difference between MMAS and ILS becomes more pronounced. While MMAS solves all instances in all runs to optimality, ILS gets stuck at suboptimal solutions in all the four instances tested. Among the instances tested, ry48p and kro124p are easily solved by MMAS, while on these the relatively poor performance of ILS is most striking. Only the two instances ft70 and ftv170 are somewhat harder. Computational results from other researchers suggest that in particular instance ft70 is, considering its small size, relatively hard to solve [39, 54, 17].

CONCLUSIONS

The key to the application of ACS to a new problem is to identify an appropriate representation for the problem (to be represented as a graph searched by many artificial ants), and an appropriate heuristic that defines the distance between any two nodes of the graph. Then the probabilistic interaction among the artificial ants mediated by the pheromone trail deposited on the graph edges will generate good, and often optimal, problem solutions.

There are many ways in which ACS can be improved so that the number of tours needed to reach a comparable performance level can diminish, making its application to larger problem instances feasible. First, a local optimization heuristic like 2-opt, 3-opt or Lin-Kernighan (Lin and Kernighan, 1973) can be embedded in the ACS algorithm (this is a standard

approach to improve efficiency of general purpose algorithms like EC, SA, NNs, as discussed in (Johnson and McGeoch, in press)). In the experiments presented in this article, local optimization was just used to improve on the best results produced by the various algorithms. On the contrary, each ant could be taken to its local optimum before global trail updating is performed. Second, the algorithm is amenable to efficient parallelization, which could greatly improve the performance for finding good solutions, especially for high-dimensional problems. The most immediate parallelization of ACS can be achieved by distributing ants on different processors: the same TSP is then solved on each processor by a smaller number of ants, and the best tour found is exchanged asynchronously among processors.

REFERENCES

1. Balas, E., Ceria, S. and Cornuéjols, G, "A lift-and-project cutting plane algorithm for mixed 0-1 programs",Mathematical Programming, PP 295–324, 1993
2. Dorigo, M., Maniezzo, V. and Colorni, A., "The Ant System: Optimization by a colony of cooperating agents" IEEE Transactions on Systems, Man, and Cybernetics–Part B, 26, 29–41,1996.
3. Hölldobler, B. and Wilson, E.O., "The ants" (Springer-Verlag, Berlin).1990
4. Lin, S. and Kernighan, B.W, "An effective Heuristic Algorithm for the TSP", Operations Research, 21, 498–516.1973
5. A.Colorni, M. Dorigo et V. Maniezzo, "Distributed Optimization by Ant Colonies", Elsevier Publishing, 134-142, 1991.
6. M. Dorigo, "Optimization, Learning and Natural Algorithms" PhD thesis, Politecnico di Milano, Italie, 1992.
7. S. Goss, S. Aron, J.-L. Deneubourg et J.-M. Pasteels, "The selforganized exploratory pattern of the Argentine ant"Naturwissenschaften, volume 76, pages 579-581, 1989
8. S. Salhi and M. Sari, "A multi-level composite heuristic for the multi-depot vehicle fleet mix problem," European Journal for Operations Research, vol.103, no.1, pp.95-112, 1997.
9. W. N. Chen and J. ZHANG "Ant Colony Optimization Approach to Grid Workflow Scheduling Problem with Various QoS Requirements", IEEE Transactions on Systems, Man, and Cybernetics--Part C: Applications and Reviews, Vol. 31, No. 1,pp.29-43,Jan 2009.
10. Pearl, J. "Probabilisitic Reasoning in Intelligent Systems: Networks of Plausible Inference", San Mateo, CA, Morgan Kaufmann,1998