Two-Stage Ant Colony System for Solving the Traveling Salesman Problem

Amilkar Puris[1], Rafael Bello[1], Ann Nowe[2] and Yailen Martínez[1]

[1] Department of Computer Science, Universidad Central de Las Villas, Cuba.
[2] Comp Lab, Department of Computer Science, Vrije Universiteit Brussel, Belgium.
{rbellop, ayudier, yailenm}@uclv.edu.cu, ann.nowe@vub.ac.be

Abstract. In this paper we propose a multilevel approach of Ant Colony Optimization to solve the Traveling Salesman Problem. In this case we use the Ant Colony System algorithm. The basic idea is to split the heuristic search performed by ants into two stages. Also, the effect of using local search was analyzed. We have studied the performance of this new algorithm for several Traveling Salesman Problem instances. Experimental results obtained show the Two-stage approach significantly improves the Ant Colony System in terms of the computation time needed.

1 Introduction

Ant Colony Optimization (ACO) is a metaheuristic used to guide other heuristics in order to obtain better solutions than those that are generated by local optimization methods. In ACO a colony of artificial ants cooperates to look for good solutions to discrete problems. Artificial ants are simple agents that incrementally build a solution by adding components to a partial solution under construction. This computational model was introduced in 1991 by M. Dorigo and co-workers [12] and [13]. Information about this metaheuristic can be found in [1], [3] and [5].

Ant System (AS) is the first ACO algorithm; it was introduced using the Traveling Salesman Problem (TSP) [2] and [4]. In TSP, we have a set of N fully connected cities $\{c_1, \ldots, c_n\}$ by arcs (i,j); each arc is assigned a weight d_{ij} which represents the distance between cities i and j, the goal is to find the shortest possible tour visiting each city once before returning to initial city. When ACO is used to solve this problem, pheromone trails (τ_{ij}) are associated to arcs which denote the desirability of visiting city j directly from city i. Also, the function $\eta_{ij}= 1/d_{ij}$ indicates the heuristic desirability of going from i to j. Initially, ants are randomly associated to cities. In the successive steps ant k applies a random proportional rule to decide which city to visit next according to expression (1):

$$p_{ij}^{k} = \frac{(\tau_{ij})^{\alpha} * (\eta_{ij})^{\beta}}{\sum_{l \in N_i^k} (\tau_{il})^{\alpha} * (\eta_{il})^{\beta}} \qquad if \quad j \in N_i^k \quad (neighborhood \quad of \quad ant\ k) \tag{1}$$

Received 02/06/06
Accepted 03/10/06
Final version 09/10/06

where α and β are two parameters to point out the relative importance of the pheromone trail and the heuristic information, respectively. After all ants have built their tours the values τ_{ij} are updated in two stages. First, τ_{ij} values are decreased by evaporation, $\tau_{ij}=(1-\alpha)*\tau_{ij}$, using the parameter α, where $0<\alpha<1$. This to avoid unlimited accumulation of pheromone. Secondly, all ants increase de value of τ_{ij} on the arcs they have crossed in their tours, $\tau_{ij}=\tau_{ij}+Inc_{ij}$, where Inc_{ij} is the amount of pheromone deposited by all ants which included the arc (i,j) in their tour. Usually, the amount of pheromone deposited by ant k is equal to $1/C_k$, where C_k is the length of the tour of ant k.

Some direct successor algorithms of Ant Systems are: Elitist AS, Rank-based AS and MAX-MIN AS. A more different ACO algorithm is Ant Colony System (ACS). ACS uses the following pseudorandom proportional rule to select the next city j from city i.

$$j=\begin{cases}\arg\max_{l\in N_i^k}\left\{\tau_{ij}*\left(\eta_{il}\right)^{\beta}\right\} & if \quad q\le q_0\\ random\ selection\ according\ to\ (1) & otherwise\end{cases}. \qquad (2)$$

where q is a random variable uniformly distributed in [0,1], q_0 which is a parameter taken in the interval [0,1], controls the amount of exploration, and $\alpha=1$ in the random selection (expression 1). In ACS, ants have a local pheromone trail update ($\tau_{ij}=(1-\xi)*\tau_{ij}+\xi*\tau_{ij}(0)$) applied after crossing an arc(i,j), where $\tau_{ij}(0)$ represents the initial value for the pheromone, and a global pheromone trail update ($\tau_{ij}=(1-\alpha)*\tau_{ij}+\alpha*Inc_{ij}$) executed only by the best-so-far ant.

In this paper, we propose a new approach to ACO in which the search process developed by ants is splitted into two stages. We have studied the performance of this proposal using the ACS algorithms. In the following, we analyze some related works and introduce the new algorithm. After that, the performance of it is studied in the case of the Travelling Salesman Problem (TSP). Finally we conclude our findings.

2 Related Work

The problem of finding low cost tours in reasonable time rather than solving the problem to optimality in the case of the TSP using a multilevel approach was addressed in [8]. The multilevel idea was first proposed by Bernard and Simon [9] as a method in speeding-up the recursive spectral bisection algorithm partitioning unstructured problems. It has been recognized that an effective way of accelerating search algorithms is to use multilevel techniques. The approach presented in [8] progressively coarsens the TSP, initializes a tour and then employs either the Lin-Kernighan or the Chained Lin-Kernighan algorithms to refine the solution on each coarsened problem; the resulting multilevel algorithm is shown to considerably enhance the quality of tours.

On the other hand, authors in [10] present a multilevel approach to ACO, for solving the mesh partitioning problem in the finite element methods. The multilevel ant colony algorithm performs very well and it is shown to be superior to several classical mesh partitioning methods. Their studies show that ACO was successful to solve the graph-partitioning problem in the case of graphs of smaller size as a result of this, they enhanced the basic ACO with a multilevel technique. That is, a set of the largest independent subgraph is created from the original graph, these are optimized using ACO, and then the optimized partition is expanded.

In the model proposed in this paper we use a multilevel approach to solve the TSP motivated by the fact that it was shown that a multilevel strategy is beneficial in order solve the TSP [14]. In this case, we introduce the multilevel approach to ACO. However, this model differs from the previous approach [10] in some aspects: (i) the search process developed by the ants is divided in two stages instead of on a grouping of cities; (ii) the search process developed by ants is partitioned into two more simple search processes instead of partitioning the problem into more simple subproblems; (iii) the partial solution obtained in the first level is used as initial state for the search of the ants in the second level.

3 Two-Stage ACS Algorithm

The Two-Stage Ant Colony System algorithm (TS-ACS) proposed in this investigation is based on the following idea: to divide the search process made by the ants in two stages, so that, in the first stage preliminary results are reached (partial solutions) that serve as an initial state for the search made by the ants in the second stage. In the case of TSP, this means that tours containing a subset of cities are generated in the first stage, in the second stage, these routes will serve like an initial state for the ants.

The determination of the initial state in which the search process starts has been an interesting problem in heuristic search. It is well known that the initial state has an important effect on the search process. The aim is to be able to approach the initial state to the goal state. Of course, it is necessary to consider an adequate balance between the computational cost of obtaining that initial state and the total cost; in other words, the sum of the cost of approaching the initial state to the goal state plus the cost of finding the solution from that "improved" initial state should not be greater than the cost of looking for the solution from a random initial state.

More formally, the purpose is the following. Let E_i be an initial state randomly generated, or obtained by any other method without a significant computational cost, E_i^* is an initial state generated by some method M that approaches it to the goal state, $C_M(E_i^*)$ denotes the cost of obtaining the state E_i^* from E_i using the method M, and $CC_{HSA}(x)$ is the computational cost to find a solution from state x using a Heuristic Search Algorithm (HSA). Then, the objective is that $C_M(E_i^*) + CC_{HSA}(E_i^*) < CC_{HSA}(E_i)$.

In the TS-ACS proposed here, the procedure to generate E_i^* and the HSA are both the ACS algorithm, so the objective is $C_{ACS}(E_i^*) + CC_{ACS}(E_i^*) < CC_{ACS}(E_i)$. As ACS is used in both stages, the difference between the 2 stages is obtained by giving different values to some parameters of the model in each stage.

A ratio (r) is introduced in order to establish the relative setting of the values of the algorithm's parameters in both stages; the ratio indicates which proportion of the complete search is given to the first stage. For instance, if r=0.3, means that the first stage will cover 30% of the search process and the second stage the remaining 70% (see an example of the application of this ratio in next section).

The setting of the ratio r has a high influence on the overall performance of the algorithm. A high value of r, say almost 1, causes the state E_i^* to be closer to the goal state, by doing so the value of $C_{ACS}(E_i^*)$ may increase and the value of $CC_{ACS}(E_i^*)$ will decrease. But, in addition to this balance between the costs of $C_{ACS}(E_i^*)$ and $CC_{ACS}(E_i^*)$, we have the problem about how much the space search is explored; while more greater is the rate r, the search in the second stage decreases for several reasons: (I) there are less ants working, (II) the amount of cycles decreases, and (III) although the quantity of possible initial states for the second stage must grow when r grows, that amount is already limited by the result of the previous stage.

Therefore, a key point is to study what value of rate r is the best in order to obtain the best balance between the searches in both stages. This value must allow:

- To minimize the value of $C_{ACS}(E_i^*) + CC_{ACS}(E_i^*)$.
- To allow an exploration of the search space that guarantees to find good solutions.

4 Two-Stage ACS in the Travelling Salesman Problem

When applying the ACS algorithm to the TSP, the ants begin the search starting from random initial states; that is, in each cycle an ant begins its tour in a randomly selected city, and chooses the next city to visit using rule (2). In the beginning no pheromone information is available to guide the search, only the heuristic information is present. On the contrary, the TS-ACS constructs partial tours (they do not include all the cities) in the first stage; this information serves as initial state for the ants in the second stage of the search process. In other words, instead of restarting the search from scratch every cycle, ants use the partial tours built in the first stage as the starting point in the second stage.

In TSP the parameters whose values are set depending on the ratio r are: the quantity of ants (m) to use in each stage, the number of cycles to execute in each stage (nc), and the amount of cities (cc) that must be included in the tour in each stage. In the experiments reported below we compare the conventional single stage ACS with the proposed Two Stages ACS. The setting of the parameters is done in the following way. Suppose a setting of the parameters for the conventional ACS algorithm as follows: m=100, nc=100, and cc=30, and a setting of the ratio as r=0.3, then the values of these parameters for the Two Stage ACS were set as follows m_1=100*0.3=30, nc_1=100 * 0.3 = 30 and cc_1=30 * 0.3 = 9 for the first stage; and m_2=100 * 0.7 = 70, nc_2=100 * 0.7 = 70 and cc_2= 30. Meaning that 30 ants execute the ACS algorithm during 30 cycles starting from random cities and constructing tours of 9 cities. In the second stage, 70 ants will execute ACS algorithm during 70 cycles forming tours of 30 cities. This means that in the first stage 30% of the ants, search for solutions which is reduced in size (tours including only 30% of the cities), and this in 30% of the total number of cycles. In the second stage the remaing 70% of the ants are used, they get 70% of the total number of the cycles, in order to find solutions to the complete problem. When the first stage is finished, we select a subset of solutions (denoted by EI) containing a quantity (cs) of the best solutions (tours with shortest distance) found in the first stage.

A refinement algorithm based on a local search strategy is introduced to improve the results of the first stage which explores small regions of the solution space. This means, the partial solutions in the set EI resulting from stage 1 are improved using the well known 2-opt procedure [11]. The TSP 2-exchange neighborhood of a candidate solution s consist of the set of all solutions s^* that can be obtained from s by exchanging two pairs of arcs in any possible way [5].

This improved EI subset, provides the initial states for the second stage. This means that in each cycle of the second stage each ant chooses, in a random way, an element of EI is a tour and positions itself in a randomly selected city belonging to the tour. After that, the ant will add other nc2 cities to this tour using the ACS algorithm. The initial values of the pheromones in the second stage are those attained at the end of the first stage.

The TS-ACS-TSP algorithm is given below:

Given the parameters (beta, rho, epsilon, cc, factor r, number of solutions in EI (cs))

```
P0: Define the quantity of ants (m) either like an
input data or by using some method depending on the
number of cities.

P1: Stage 1.
P1.1: Calculate the parameters for the first stage:
        $m_1$= r*m
        $nc_1$= r*nc
        $cc_1$= r*cc
P1.2: Apply the ACS algorithm, which in the first stage
develops $nc_1$ cycles.
P1.3: Set of tours ← Tours generated by ACS algorithm
in the first stage.

P2: Stage 2.
P2.1: Calculate the parameters for the second stage:
        $m_2$= m-$m_1$
        $nc_2$= nc-$nc_1$
        $cc_2$= cc
        EI ← Selecting the best cs solutions from Set
              of tours.

P2.2: Apply a local search to improve partial solutions
in EI set.
P2.3: Apply the ACS algorithm, which in the second
stage develops $nc_2$ cycles, using the elements of EI like
initial states for the ants in the second stage.
```

5 Experimental Results

A comparative study of ACS and TS-ACS algorithms in the TSP was done by using public available (or benchmark) data base for this problem [7].

For these experiments we use the following parameter setting: α=0.1, ξ=0.1, a quantity of ants m=10 and a number of cycles nc=1000. Firstly, we studied the values {1, 3, 5} and {0.3, 0.6, 0.9} for β and q_0 respectively. Table 1 shows the different values of β and q_0 with which the best solution was obtained for each database. These values β and q_0 were selected to develop the next experiments. The column "Best solution" indicates the best solution reported back for that problem in the corresponding database, and the column "Best solution with ACS" contains the best solution found by our implementation of the ACS algorithm.

Table 1: Best solutions obtained using ACS algorithm.

Data base	**Best solution (BS)**	**Best solution with ACS**	**Time (ms)**	**β**	**q_0**
bays29.tsp	2020	2048	1062	5	0.6
berlin52.tsp	7542	7650	3801	5	0.6
st70.tsp	675	762	7632	5	0.6
rd100.tsp	7910	8670	18140	5	0.6
ch150.tsp	6942	6867	54001	5	0.6
kroA200.tsp	29368	32712	112982	5	0.6
tsp225.tsp	3919	4285	148351	3	0.9
a280.tsp	2579	2904	271179	3	0.9
lin318.tsp	42029	47328	398572	3	0.9
pcb442.tsp	50778	58265	973212	3	0.9
rat783.tsp	8806	12255	5081400	3	0.9

Secondly, we developed experiments to determine an adequate value for the ratio. We run the TS-ACS-TSP algorithm using several values for the ratio r = {0.2, 0.25, 0.3, 0.4, 0.5}. Tables 2 and 3 show the results averaged over 6 runs. The quality of the solution and the time cost were measured. Moreover, we made the comparison without using local search and with using the 2-opt procedure to improve the solution generated by ACS and TS-ACS algorithms.

The comparison with respect to the quality of the solution is presented in Table 2. The column "Best solution with ACS plus 2-opt" contains the best solution found by the ACS algorithm plus a local search developed with 2-opt procedure; the column "Best solution with TS-ACS" contains the best solution found by TS-ACS algorithm and the corresponding ratio with which it was found; the column "Best solution with TS-ACS plus 2-opt after first stage" contains the best solution found by TS-ACS algorithm plus a local search developed with 2-opt procedure applied to the solutions obtained after stage 1, and the column "Best solution with TS-ACS plus 2-opt after both stages" contains the best solution found by TS-ACS algorithm plus a local search developed with 2-opt procedure applied to the solutions obtained after the two stage. The time cost to obtain these solutions is presented in Table 3 (measured in seconds).

These experimental results can be summarized in the following way:

- In most of the cases TS-ACS algorithm reaches better solutions than ACS. In case ACS yields the better result, the difference to the results obtained by TS-ACS is not relevant.
- The time in which TS-ACS algorithm gets those results is between 40% and 50% of the time needed by the ACS algorithm, that is, the cost in time is reduced by half.
- A value of rate r between 0.2 and 0.35 produces the best results.
- Greater values for the ratio r decrease the time cost but also the quality (because the search space is not sufficiently covered).

Table 2. A comparisona between ACS and TS-ACS in TSP (quality solution)

Data base	**Best solution**	**BS with ACS plus 2-opt**	**BS with TS-ACS**	**BS with TS-ACS plus 2-opt after first stage**	**BS with TS-ACS plus 2-opt after both stages**
bays29.tsp	2020	2041	2058 (0.25)	2045 (r=0.2)	2026 (0.2)
berlin52.tsp	7542	7564	8034 (0.25)	7863 (0.25)	7542 (0.25)
st70.tsp	675	739	785 (0.2)	763 (0.25)	745 (0.25)
rd100.tsp	7910	8345	8668 (0.2)	8658 (0.3)	8159 (0.2)
ch150.tsp	6528	6673	6932 (0.3)	6896 (0.3)	6670 (0.2)
kroA200.tsp	29368	31598	32994 (0.2)	32889 (0.2)	31970 (0.25)
tsp225.tsp	3919	4115	4357 (0.25)	4204 (0.25)	4102 (0.2)
a280.tsp	2579	2827	3002 (0.2)	2944 (0.25)	2878 (0.25)
lin318.tsp	42029	46006	47370 (0.25)	47003 (0.3)	46086 (0.2)
pcb442.tsp	50778	56861	58420 (0.2)	58369 (0.3)	56790 (0.2)
rat783.tsp	8806	11917	12181 (r=0.3)	12202 (0.25)	11931(0.3)

Table 3. A comparison between ACS and TS-ACS in TSP (time cost)

Data base	**BS with ACS plus 2-opt**	**BS with TS-ACS**	**BS with TS-ACS plus 2-opt after first stage**	**BS with TS-ACS plus 2-opt after both stages**
bays29.tsp	1231	468	606	651
berlin52.tsp	3937	1672	1719	1734
st70.tsp	7851	3484	3609	3687
rd100.tsp	18721	10609	12492	14202
ch150.tsp	57803	20122	21984	31874
kroA200.tsp	123852	64588	66781	74873
tsp225.tsp	155151	74255	76120	78677
a280.tsp	295464	126823	135987	146898
lin318.tsp	430637	154963	172492	190220
pcb442.tsp	1040741	396250	425612	478066
rat783.tsp	5315734	2262624	2490555	2983343

6 Conclusion

We have presented an improvement of the Ant Colony Optimization based in a multilevel approach. It consists on splitting the search process developed by ants into two stages. The study was developed using the Ant Colony System algorithm. In this approach the values of some parameters (number of ants, quantity of cycles, number of cities included in each stage, etc.) are assigned a different value in each stage according to a ratio which indicates what proportion of the complete search corresponds to each stage.

We studied the performance using different ratio values in the Traveling Salesman Problem. The best results were obtained when this value is about 0.3.

This new approach to ACO produces an important reduction of the computation time cost, yet preserving the solution quality.

References

1. Dorigo, M. et al. The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man and Cybernetics-Part B, vol. 26, no. 1, pp. 1-13. 1996.
2. Dorigo, M. and Gambardella, L.M.. Ant colonies for the traveling salesman problem. BioSystems no. 43, pp. 73-81, 1997.
3. Dorigo, M. et al. Ant algorithms for Discrete optimization. Artificial Life 5(2), pp. 137-172. 1999.
4. Dorigo, M. and Stützle, T.. ACO Algorithms for the Traveling Salesman Problem, 1999, Evolutionary Algorithms in Engineering and Computer Science: Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications, John Wiley & Sons., EEUU.
5. Dorigo, M. and Stutzle, T.. Ant Colony Optimization. MIT Press. 2004.
6. Stefanowski, J.. An experimental evaluation of improving rule based classifiers with two approaches that change representations of learning examples. Engineering Applications of Artificial Intelligence 17, pp. 439-445. 2004.
7. The TSPLIB Symmetric Traveling Salesman Problem Instances, http://elib.zib.de/pub/Packages/mp-testdata/tsp/tsplib/index.html
8. Walshaw, C.. A multilevel approach to the traveling salesman problem. Operations Research 50(5), 862-877, 2002.
9. Bernard, S.T. and Simon, H.D.. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. Concurrency: Practice and Experience 6 (2), pp. 101-117. 1994.
10. Korosec, P. et al.. Solving the mesh-partitioning problem with an ant-colony algorithm. Parallel Computing 30, pp. 785-801. 2004.
11. Michalewicz, Z. and Fogel, D.B.. How to solve it: modern heuristics. Springer-Verlag. 2004.
12. Dorigo, M., Maniezzo, V., Colorni, A.: Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy (1991).
13. Dorigo, M.: Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale. PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy (1992).
14. Walshaw, C. et al.. The multilevel paradigm: a generic Meta-heuristic for Combinatorial Optimization Problems?. Technical Report 00/IM/63, Univ. Greenwich, London SE10 9LS, UK. 2000.