Two Step Ant Colony System to Solve the Feature Selection Problem

Rafael Bello¹, Amilkar Puris¹, Ann Nowe²,
Yailen Martínez¹, and María M. García¹

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, mmgarcia}@uclv.edu.cu,
ann.nowe@vub.ac.be

Abstract. In this paper we propose a new model of ACO called Two-Step Ant Colony System. The basic idea is to split the heuristic search performed by ants into two stages. We have studied the performance of this new algorithm for the Feature Selection Problem. Experimental results obtained show the Two-Step approach significantly improves the Ant Colony System in term of 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 by M. Dorigo. Information about this metaheuristic can be found in [6], [8] and [10].

Ant System (AS) is the first ACO algorithm; it was introduced using the Travelling Salesman Problem (TSP) [7] and [9]. In TSP, we have a set of N fully connected cities $\{c_1, \dots, c_n\}$ 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 trip visiting each city once before returning to initial city. When ACO is used to solve this problems, 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 , where d_{ij} is the distance between cities i and 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):

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

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, $\tau_{ij}=(1-\rho)*\tau_{ij}$, using the parameter ρ , where $0<\rho<1$. This to avoid unlimited accumulation of pheromone. Second, all ants increase the 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 .

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

where q is a random variable uniformly distributed in $[0,1]$, q_0 which is a constant taken in the interval $[0,1]$, controls the amount of exploration. 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 ($\tau_{ij}=(1-\rho)*\tau_{ij} + \rho*Inc_{ij}$) 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 algorithm. In the following, we first introduce the new algorithm. After that, the performance of it is studied in the case of the Feature Selection Problem (FSP). Finally we conclude our findings.

2 Two-Step ACS Algorithm

The Two-Step Ant Colony System algorithm 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 FSP, this means that subsets containing a subset of features are generated in the first stage; in the second stage, these subsets 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 in 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, $C_M(E_i, E_i^*)$ denotes the cost of obtaining E_i^* from state E_i using the method M , and $CC_{HSA}(x)$ is the computational cost of finding a solution from state x using a Heuristic Search Algorithm (HSA). Then, the objective is that $C_M(E_i, E_i^*) + CC_{HSA}(E_i^*) < CC_{HSA}(E_i)$.

In the TS-ACS approach proposed here, the procedure to generate E_i^* and the HSA are both the ACS algorithm, so the objective is $C_{ACS}(E_i, E_i^*) + CC_{ACS}(E_i^*) < CC_{ACS}(E_i)$. 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 step. A ratio (r) is introduced in order to establish the relative setting of the values of the parameters of the algorithm in both stages; the ratio indicates which proportion of the complete search is given to the first stage. For example, if $r=0.3$, means that the first step will cover 30% of the search process and the second step the remaining 70%.

The setting of the ratio r has a high influence on the overall performance of the algorithm. A high value of r , that is, about value 1, causes the state E_i^* to be closer to the goal state, then the value of $C_{ACS}(E_i, E_i^*)$ may increase and the value of $CC_{ACS}(E_i^*)$ will decrease. But, in addition to this balance between the costs of $C_{ACS}(E_i, E_i^*)$ and $CC_{ACS}(E_i^*)$, we have the problematic 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 diminish the value of $C_{ACS}(E_i, E_i^*) + CC_{ACS}(E_i^*)$.
- To allow an exploration of the search space that guarantees to find good solutions.

3 Two-Step ACS in the Feature Selection Problem

In this section we discuss some results we obtained when applying the two stage approach to the feature selection problem. Feature selection is useful in different computational tasks, for instance, in machine learning processes. An appropriate representation space for learning by selecting relevant attributes to the problem domain is a crucial issue for learning systems [11], [16] and [17]. Usually, not all features describing the examples are relevant to the classification process and some of them are irrelevant or redundant. Too many irrelevant features increase the complexity of learning process and decrease the accuracy of induced knowledge. Feature selection is useful to reduce the dimensionality problem; it results not only in improving the speed of data manipulation [14], but even in improving the classification rate by reducing the influence of noise [5] and [15].

The FSP can be viewed as a particular case of a more general subset selection problem in which the goal is to find a subset maximizing some adopted criterion. Feature selection methods search through the subsets of features and try to find the

best subset among the competing 2^N-1 candidate subsets according to some evaluation measure, where N denotes the total number of features.

Each state represents a subset of features in the search space. All feature selection methods contain two important components: an Evaluation function used to evaluate a candidate feature subset and a Search algorithm to search through the feature space. Search strategies are important because the feature selection process may be time consuming and an exhaustive search for the “optimal” subset is impractical for even moderate sized problems [18]. Examples of search strategies are heuristic search, probabilistic methods and hybrid algorithms.

ACO has been used as a search algorithm for the FSP. Methods which combine ACO and Rough Set Theory (RST) to find reducts with promising results were proposed in [1], [2], [3] and [12]. They are based on the reduct concept.

A Decision System is a pair $DS=(U, A \cup \{d\})$, where U is a non-empty finite set of objects called the Universe, A is a non-empty finite set of features, and $d \notin A$ is the decision feature. The basic concepts of RST are the lower and upper approximations of a subset $X \subseteq U$ [13]. These were originally introduced with reference to an indiscernibility relation $IND(B)$, where objects x and y belong to $IND(B)$ if and only if x and y are indiscernible from each other by features in B .

Let $B \subseteq A$ and $X \subseteq U$. It can be proved that B defines an equivalence relation. The set X can be approximated using only the information contained in B by constructing the B -lower and B -upper approximations of X , denoted by B_*X and B^*X respectively, where $B_*X = \{x : [x]_B \subseteq X\}$ and $B^*X = \{x : [x]_B \cap X \neq \emptyset\}$, and $[x]_B$ denotes the class of x according to B -indiscernible relation. The objects in B_*X are sure members of X , while the objects in B^*X are possible members of X . If $B^*X - B_*X$ is not empty, then X is a rough set.

RST offers several measures about a Decision System. Among them is the quality of the approximation of classification (expression 3). It expresses the percentage of objects which can be correctly classified into the given classes $Y = \{Y_1, \dots, Y_n\}$ employing only the set of features in B .

$$\gamma_B(Y) = \frac{\sum_{i=1}^n |B_*Y_i|}{|U|} \quad (3)$$

An important issue in the RST is feature reduction based on the reduct concept. A reduct is a minimal set of features that preserves the partitioning of universe and hence the ability to perform classifications. The subset B is a reduct if $IND(A)=IND(B)$; that is, $\gamma_A(Y)=\gamma_B(Y)$. The concept of reduct is one of the most important concepts in the Rough Set Theory.

However, this beneficial alternative is limited because of the computational complexity of calculating reducts. The problem of finding a globally minimal reduct for a given information system is NP-hard. For that reason, methods for calculating reducts have been developed using heuristic search, such as is studied in [12].

When ACO is used to solve the feature selection problem the graph representation is lightly different to the TSP case. This problem can be modeled in the following way. Let $A=\{a_1, a_2, a_{na}\}$ be a set of features. You can view this set as a network in

which nodes represent features, and all nodes are connected by bi-directional links. Pheromone values τ_i are associated to nodes a_i , τ_i represents the absolute contribution of the feature a_i to a reduct. In the first step, each ant k is assigned to one node, it can move to all nodes in the network ($b_k \leftarrow \{a_i\}$, where b_k is the subset ant k has to build). Ants perform a forward selection in which each ant k expands its subset b_k step-by-step by adding new features; for performing it, each ant k looks for all features in the set $A-b_k$ and selects the next feature among them to include in b_k according to the ACO rule. This rule is the pseudorandom proportional rule (expression 2) in the case of ACS. The quality of the approximation of classification measure of RST (expression (3)) is used like a heuristic function (η) in the ACO model ($\eta(B)=\gamma_B(Y)$).

The ACS algorithm showed the best performance when a comparative study about several ACO algorithms was developed to solve the feature selection problem, see [2] and [3]. For this reason, we use the Ant Colony System algorithm to study the two stage approach.

The Two-step ACS algorithm in the FSP is based in the following key idea: to split the process of finding reducts in two stages. In the first stage, the algorithm builds candidate subsets, which are used in the second stage as initial subsets for each ant. In order to apply the two stage approach we need to decide how to set some parameters. The number of ants and the number of cycles per stage is calculated according to the ratio r ; that is, let m the quantity of ants to use and NC_{max} the number of cycles to perform, these values are divided for each stage in the following form: First stage $\{m_1=m*r \text{ and } NC_{max_1}=NC_{max}*r\}$, and Second stage $\{m_2=m*(1-r) \text{ and } NC_{max_2}=NC_{max}*(1-r)\}$. Also, we must define what a partial solution is in the first stage. In this case, the reference value is the quality of the approximation of classification measure $\gamma_A(Y)$, and ants have to build subset of features with a quality of the approximation of approximation equal to $r*\gamma_A(Y)$. In the second stage we are looking for a solution to the original problem, so the subset has to obtain the value $\gamma_A(Y)$.

In other words, in the first stage m_1 ants during NC_{max_1} cycles will be looking for subset of features with a quality of the approximation of classification equal to $r*\gamma_A(Y)$. In the second stage, m_2 ants will perform NC_{max_2} cycles using the subsets resulting from the first stage as initial states in order to generate reducts.

The TS-ACS-RST-FS algorithm has the following structure:

Given parameters α , β , NC_{max} (total number of cycles), na , $ratio$, ro and ϵ

P0: Calculate the quantity of ants (m) according to the number of features (na).
Calculate the quality of the classification using all features ($\gamma_B(Y)$ for $B=A$).

P1: Stage 1

Calculate value for parameters in the first stage:
 $NC_{max_1}=ratio*NC_{max}$

$m_1= ratio*m$; $\gamma_{B1}(Y)=ratio* \gamma_A(Y)$

Apply ACS-RST-FS algorithm.

Candidate subsets $\leftarrow$ subSetselected by the ACS-RST-FS algorithm.

P2: Stage 2

Calculate value for parameters in the second stage:

$$NCmax_2 = NCmax - NCmax_1$$

$$m_2 = m - m_1 \quad ; \quad \gamma_{B2}(Y) = \gamma_A(Y)$$

Apply ACS-RST-FS algorithm, in each cycle assign a random subset from Candidate subsets as initial subset for each ant.

The value of the ratio is an important parameter. Values near to 1 produce subsets near to the definition of reducts in the first stage.

For the experiments, we have used the following values for parameter ratio $\{0.2, 0.3, 0.36, 0.5, 0.6, \text{ and } 0.8\}$. We have studied the effect of the ratio in three aspects, the quantity of reducts, the length of reducts and the computation time. Each algorithm was executed 10 times and the results were averaged over these 10 runs.

In Table 1, we present the results obtained using a decision system that contains 20 objects which are described by 16 discrete features ($na=16$) and belongs to two classes. The first class contains 9 objects and the second one contains 11 objects, the quality of the approximation of classification is equal to 1 ($\gamma_A(Y)=1$); so, this is a consistent decision system (each pair of indiscernible objects belongs to the same class). The quantity of cycles was 21 ($NCmax=21$). In columns number 6 and 7 the information is presented in the following form: averaged number of reducts/averaged length of reducts/time.

Table 1. A comparison between algorithms ACS-RST-FS and TS-ACS-RST-FS

Algorithm	NC max ₁	NC max ₂	m ₁	m ₂	beta=5 q ₀ =0.9	beta=1 q ₀ =0.3
ACS			--	--	46.7/3.95/228s	123/4.19/274s
TS-ACS rate=0.2	4	17	3	13	32.7/4.2/82s	76.3/4.2/89.9s
TS-ACS rate=0.3	6	15	5	11	43.3/4.1/53s	71/4.2/64s
TS-ACS rate=0.36	8	13	6	10	38.7/3.9/39s	67.3/4.1/47s
TS-ACS rate=0.5	10	11	8	8	29.7/3.8/32s	43.3/4.1/44s
TS-ACS rate=0.6	13	8	10	6	20.33/3.8/41s	37/4.2/49s
TS-ACS rate=0.8	17	4	13	3	9/3.8/82s	10.67/4.2/97s

You can see the ratio about to 0.3 yields the best results. This setting obtains a quantity of reduct near to ACS-RST-FS algorithm [3] but only in the 23% of the time. Similar results were obtained using other databases. This is an expected result, because the value ratio=0.3 offers a good balance between both stages; a higher numbers of ants and cycles in the second stages allows the algorithms to

perform a larger exploration of the search space from of initial subsets having certain quality.

In Table 2, we present a similar study using Breast Cancer database from UCI Repository [4], in this database $n=9$ and we used 9 ants ($m=9$). In columns number 4 and 5 the information is presented in the following form. In the first row, the results obtained by the ACS algorithm (average number of reduces/time) are presented; in the rest of the rows we present the results yielded for the TS-ACS-RST-FS algorithm (percent of average number of reduces/percent of time) respects to the results showed in the first row. For instance, the TS-ACS-FSP algorithm with $r=0.3$, $\beta=5$ and $q_0=0.9$ obtained an average of 50.9 reduces (109% respects to 46.7 reduces) in only 71 seconds (31% respects to 228s). These results are very interesting because the Two-Step ACS approach enable us to obtain equal or greater quantity of reduces in lower time than ACS-RST-FS algorithm.

From tables 1 and 2, it is possible to see the time cost in the case of TS- ACS-RST-FS algorithm is very low. From this advantage, we propose a second idea: to increase the quantity of ants in order to yield a greater exploration of the search space. We developed experiments in which the number of ants was increased by the factors {1.33, 1.5, 1.8, 2.1}. In the last row of Table 2 the results with value 2.1 are showed. The relation is established respect the quantity of reduces and the time. For instance, when the number of ants is increased to $2.1*m$ ($m=2.1*9=19$), the algorithm TS-ACS-RST-FS obtains 124% number of reduces respect to the quantity of reduces obtained by ACS-RST-FS algorithm in only 67% of the time used by this last algorithm (for $\beta=5$ and $q_0=0.9$). We used the $rate=0.3$ because this value shows the best results in experimental results.

Table 2. A comparison between ACS-RST-FSP algorithm and several alternatives to TS-ACS-RST-FS algorithm ($NC_{max1}=6$, $NC_{max2}=15$)

Algorithm	m_1	m_2	$\beta=5$ $q_0=0.9$	$\beta=1$ $q_0=0.3$
ACS $m=9$	--	--	46.7/228s	123/274s
TS-ACS $rate=0.2$	2	7	60%/34%	70/49%
TS-ACS $rate=0.3$	3	6	109%/31%	73%/37%
TS-ACS $rate=0.36$	3	6	105%/25%	77%/31%
TS-ACS $rate=0.5$	4	5	100%/22%	73%/26%
TS-ACS $rate=0.6$	5	4	65%/13%	50%/20%
TS-ACS $rate=0.8$	7	2	33%/27%	31%/26%
TS-ACS $rate=0.3$ $m=2.1*m=19$	6	13	124%/67%	103%/83%

4 Conclusion

We have presented an improvement of the Ant Colony Optimization which 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 some parameters (number of ants, quantity of cycles, etc.) receive different values in each stage according to a ratio parameter which indicates what proportion of the complete search corresponds to each stage.

We studied the performance using different ratio values in the Feature selection problem. The experimental results showed the performance of ACO was increased strongly; the best results were obtained when the value of ratio is about 0.3.

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

Acknowledgments

The authors would like to thanks VLIR (Vlaamse InterUniversitaire Raad, Flemish Interuniversity Council, Belgium) for supporting this work under the IUC Program VLIR-UCLV.

References

1. Bello, P.R., Nowe, A., et al. Using ACO and Rough Set Theory to Feature Selection. WSEAS Transactions on Information Science and Applications, Issue 5, Vol. 2, May 2005, pp. 512-517.
2. Bello, P.R., Nowe, A., et al. A Model based on Ant Colony System and Rough Set Theory to Feature Selection. Proceedings of Genetic and Evolutionary Computation Conference (GECCO05), June 25-29, Washington DC, USA, 2005, pp. 275-276. ACM Press, ISBN 1-59593-038-8.
3. Bello, P.R., Nowe, A., et al. Using Ant Colony System meta-heuristic and Rough Set Theory to Feature Selection. Proceedings of The 6th Metaheuristics International Conference (MIC2005), August 22-26, Vienna, Austria. 2005. Printed by University of Vienna.
4. Blake, C.L. and Merz, C.J.. UCI repository of machine learning databases, 1998. <http://www.ics.uci.edu/~mllearn/MLRepository.html>.
5. Dong, M. and Kothari, R.. Feature subset selection using a new definition of classifiability. Pattern Recognition Letter, vol. 24, issues 9-10, pp. 1215-1225. 2003.
6. 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.
7. Dorigo, M. and Gambardella, L.M.. Ant colonies for the traveling salesman problem. BioSystems no. 43, pp. 73-81, 1997.
8. Dorigo, M. et al. Ant algorithms for Discrete optimization. Artificial Life 5(2), pp. 137-172. 1999.
9. 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.

10. Dorigo, M. and Stutzle, T.. *Ant Colony Optimization*. MIT Press. 2004.
11. Inza, I. et al.. Feature subset selection by bayesian networks based optimization. *Artificial intelligence*, vol. 123, No. 1-2, pp. 157-184. 2000.
12. Jensen, R. and Shen. Q.. Finding rough set reducts with ant colony optimization. *Proceedings of UK Workshop on Computational Intelligence*, pp. 15-22. 2003.
13. Komorowski, J. Pawlak, Z. et al.. *Rough Sets: A tutorial*. In Pal, S.K. and Skowron, A. (Eds) *Rough Fuzzy Hybridization: A new trend in decision-making*. Springer, pp. 3-98. 1999.
14. Kudo, M. and Sklansky, J.. Comparison of algorithms that select features for pattern classifiers. *Pattern Recognition* 33, pp. 25-41. 2000.
15. Somol, P. and Pudil, P.. Feature selection toolbox. *Pattern Recognition* 35, pp. 2749-2759. 2002.
16. 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.
17. Xing, H. and Xu, L.. Feature space theory –a mathematical foundation for data mining. *Knowledge-based systems* 14, pp. 253-257. 2001.
18. Zhang, H. and Sun, G.. Feature selection using tabu search method. *Pattern Recognition* 35, pp. 710-711. 2002.