Promoting Search Diversity in Ant Colony Optimization with Stubborn Ants

Ashraf M. Abdelbar
Donald C. Wunsch
Missouri University of Science and Technology, dwunsch@mst.edu

Follow this and additional works at: https://scholarsmine.mst.edu/ele_comeng_facwork

Part of the Electrical and Computer Engineering Commons

Recommended Citation
A. M. Abdelbar and D. C. Wunsch, "Promoting Search Diversity in Ant Colony Optimization with Stubborn Ants," Procedia Computer Science, vol. 12, pp. 456-462, Elsevier, Jan 2012.
The definitive version is available at https://doi.org/10.1016/j.procs.2012.09.104

This work is licensed under a Creative Commons Attribution-No Derivative Works 3.0 License.

This Article - Conference proceedings is brought to you for free and open access by Scholars' Mine. It has been accepted for inclusion in Electrical and Computer Engineering Faculty Research & Creative Works by an authorized administrator of Scholars' Mine. This work is protected by U. S. Copyright Law. Unauthorized use including reproduction for redistribution requires the permission of the copyright holder. For more information, please contact scholarsmine@mst.edu.
Promoting search diversity in ant colony optimization with stubborn ants

Ashraf M. Abdelbar<sup>a</sup>*, Donald C. Wunsch II<sup>b</sup>

<sup>a</sup>American University in Cairo, Cairo, Egypt
<sup>b</sup>Missouri University of Science & Technology, Rolla, Missouri, USA

Abstract

In ant colony optimization (ACO) methods, including Ant System and MAX-MIN Ant System, each ant stochastically generates its candidate solution, in a given iteration, based on the same pheromone $\tau$ and heuristic $\eta$ information as every other ant. Stubborn ants is an ACO variation in which if an ant generates a particular candidate solution in a given iteration, then the components of that solution will have a higher probability of being selected in the candidate solution generated by that ant in the next iteration. In previous work, we evaluated this variation with the MMAS Ant System model and the Traveling Salesman Problem (TSP), and found that it can both improve solution quality and reduce execution-time. In this paper, we evaluate stubborn ants with Ranked Ant System, and find that performance also improves in terms of solution quality and execution time.

Keywords: Swarm intelligence.

1. Overview

Ant colony optimization (ACO) [8] is an active area of research concerned with using problem solving mechanisms observed in nature in social insects in solving computational problems. In most ACO methods for discrete combinatorial optimization, including Ant System [6], MAX-MIN Ant System [11], and others [5,7], each ant stochastically generates its solution, in a given iteration, based on the same pheromone $\tau$ and heuristic $\eta$ information as every ant. In a given iteration, the probability that a given candidate solution will be generated by a given ant $k$ is identical to the probability that it will be generated by any other given ant $k$. The solutions that ant $k$ generated in iterations 1,...,$t$-1 have no effect on the probability distribution that ant $k$ uses to generate solutions in iteration $t$. 

1877-0509 © 2012 Published by Elsevier B.V. Selection and/or peer-review under responsibility of Missouri University of Science and Technology.
Stubborn ants [1,2] are a variation in which if a given ant generated a particular candidate solution $S^{t-1}$ in iteration $t-1$, then the solution components of $S^{t-1}$ will have a higher probability of being selected in the candidate solution $S^t$ generated by that ant in iteration $t$.

In previous work, we evaluated this variation in the context of the MAX-MIN Ant System model and the Traveling Salesman Problem (TSP), and found that it can both improve solution quality and reduce execution-time. In this paper, we explore the generality of the effectiveness of the stubborn ants variation by evaluating it in the context of Ranked Ant System, and find that performance also improves in terms of solution quality and execution time.

2. Background

Ant colony optimization (ACO) [8,9] is a general-purpose biologically-motivated population-based discrete optimization paradigm that can be applied to a wide variety of problems. In this section, we will describe the main computational features of ACO, without focusing heavily on the biological motivations. In our presentation, we will use the traveling salesman problem (TSP) for illustration.

An instance of TSP consists of a fully-connected weighted graph, whose nodes are called "cities," and where the weights on the edges represent the "distances" between the cities. The objective is to find the minimum-weight Hamiltonian cycle, where a Hamiltonian cycle is a simple cycle that visits every node exactly once.

ACO is based on a number of primitive processing elements, each operating in parallel with little centralized control. In ACO, the processing elements are called ants and the collection of processing elements is called a colony. In each iteration, each ant $k$ generates a candidate solution $x_k$, and the set of solutions generated by all ants is used to update a central data structure, conventionally called $\tau$, that can be thought of as representing the collective wisdom of the group. In generating its solution in a given iteration, each ant makes use of the $\tau$ data structure, and also makes use of a problem-dependent heuristic function $\eta$.

In the case of TSP, $\tau$ would be a two-dimensional $n \times n$ array, where $n$ represents the number of cities; the entry $\tau_{ij}$ represents the extent to which the collective wisdom of the swarm is inclined towards edge $e_{ij}$. A common choice for the heuristic $\eta_{ij}$ for the TSP is the reciprocal of the distance $d$ associated with the edge $e_{ij}$, $\eta_{ij} = 1/d(e_{ij})$.

A number of different algorithms [4,5,7] have been introduced within the ACO paradigm. The following steps describe most of these algorithms for a static combinatorial optimization problem such as TSP:

Initialization
while (termination criteria not reached) do
   SolutionConstruction
   LocalSearch // optional
   PheromoneUpdate

The different ACO algorithms that have been studied are generally similar in the SolutionConstruction step, and mostly differ in the PheromoneUpdate step. Below, we expand the algorithmic skeleton given above for Ranked Ant System (RAS) in the context of TSP.

Outline of RAS: The following steps are repeated until some termination criteria is reached:

1. SolutionConstruction: For each ant $k$, construct a tour $S_k$ by adding one edge at a time until a full tour is constructed. Because of the Hamiltonian cycle constraint, each choice of the next edge to add to the tour must be made in the context of the partial tour that has already been constructed. Therefore, we always maintain a current city $i$, and, repeatedly, we choose the next edge $e_{ij}$ to add to $S_k$ from the feasible domain
set \( D(i) \), where \( D(i) \) is the set of edges in which city \( i \) participates, excluding any edge that is already a member of \( S_k \). Once an edge \( e_{ij} \) is added to the solution, the city \( j \) then becomes the new current city and the process repeats. For the current city \( i \):

\[
\text{Pr}[e_{ij} \in S_k] = \frac{[\tau_{ij}]^\alpha [\eta_{ij}]^\beta}{\sum_{k \in D(i)} [\tau_{ik}]^\alpha [\eta_{ik}]^\beta}
\]

(1)

Thus, the probability that the edge \( e_{ij} \) is included in \( S_k \) is proportional to the product of the two terms of the numerator. The value \( \tau_{ij} \) represents the extent to which the past experience of the colony indicates it is a good idea to include the edge \( e_{ij} \), and \( \eta_{ij} \) represents a problem-dependent heuristic function that indicates the intrinsic "goodness" of the edge \( e_{ij} \). The two parameters \( \alpha \) and \( \beta \) are used to adjust the relative emphases of the two terms.

A commonly-used [8] implementation tradeoff is to pre-compute for each city, the set of its \( r \) nearest neighbors (where \( r \) is typically a value between 15 and 40), and then to restrict the feasible domain \( D_i(k) \) in Equation (1) to city \( i \)'s \( r \) nearest neighbors. If all \( r \) nearest neighbors have already been visited, the choice is made from the entire set of unvisited set. The reader is directed to [8] for more information.

2. **LocalSearch**: Once each ant has constructed a complete tour \( S_k \), it can optionally run a local-search procedure to improve the tour before proceeding to the next step. A commonly-used local search procedure for TSP [3] is 2-opt, in which all tours that can be obtained from \( S_k \) by exchanging 2 edges are exhaustively considered.

3. **PheromoneUpdate**: The \( \tau \) array is then updated to integrate the experience gained from the solutions constructed into the collective wisdom of the colony.

a. **Evaporation**: In a step called evaporation, every entry in the \( \tau \) array is reduced by an evaporation parameter \( \rho \):

\[
\tau_{ij} = \tau_{ij} (1 - \rho)
\]

where \( 0 < \rho < 1 \). Evaporation allows the colony to gradually "forget" its inclination towards a solution component unless that component continues to receive reinforcement in the pheromone deposit step (see below).

b. **Pheromone Deposit**: At the end of each iteration, each ant is given a rank \( r \) according to the cost of the tour constructed in that iteration, with the ant constructing the best tour having rank 1. Only the \( (w-1) \) best-ranked ants are allowed to deposit pheromone, where \( w \) is a parameter of the algorithm. In addition, the best-so-far solution \( S^* \) is used to deposit an additional amount of pheromone, weighted by \( w \). For every ant \( k \) with rank \( r \), where \( 1 \leq r \leq (w-1) \), we carry out the following: for every \( e_{ij} \in S_k \),

\[
\tau_{ij} = \tau_{ij} + (w - r) \frac{1}{f(S_k)}
\]

In addition, for the best-so-far solution \( S^* \), we carry out the following: for every \( e_{ij} \in S^* \),

\[
\tau_{ij} = \tau_{ij} + (w - r) \frac{1}{f(S^*)}
\]

3. **Stubborn Ants and Search Diversity**

In standard ACO models, all ants employ Equation (1), using the same \( \tau \) array and \( \eta \) heuristic function, to probabilistically generate their candidate solutions in each iteration. In stubborn ants [1], we replace Equation (1) to become the following:
where \( J \) is a stubbornness parameter that determines the degree to which an ant is biased towards its past solution. Of course, if \( J = 1 \), then our model reduces to the standard model.

The idea is that the components of the solution constructed by a given ant will have an amplified probability of being selected by the same ant in the next iteration, with the degree of amplification determined by the stubbornness parameter \( J \). Thus, instead of all ants being motivated by the same \( \tau \) and \( \eta \) information, each ant will have its own bias. This is likely to increase the diversity of the tours constructed by the colony in a given iteration, although of course there will be less diversity in the tours constructed by one particular ant from iteration to the next. However, we would argue that, on the whole, the effect of stubbornness is to promote search space exploration, especially in multimodal landscapes.

In general, the tour \( S_{t-1} \) that an ant generates in an iteration \( t-1 \) will be similar to the tour \( S_t \) that it generates in iteration \( t \), with the degree of similarity increasing as \( J \) increases. The tour \( S_t \) will then serve as an attractor in the construction of the next tour \( S_{t+1} \), which will be similar to both \( S_{t-1} \) and \( S_t \), but more similar to \( S_t \) than to \( S_{t-1} \). The tour that a particular ant is biased towards will therefore evolve gradually over time.

4. Experimental Results

We used a 3,038-city TSP instance (\( \text{pcb3038} \)) obtained from TSPLIB, with the parameter settings shown in Table 1. We ran 13 experiments, in each experiment using a different value of the \( J \) parameter. We used values of \( J \) ranging from 1 to 800, as indicated in Table 2, where the setting \( J = 1 \), of course, corresponds to the standard RAS model. Each experiment was repeated for 30 trials, and the solution cost and execution time were recorded for each trial.

Table 2 shows the results of these experiments. The columns \( \mu_c \) and \( \sigma_c \) indicate the mean and standard deviation, respectively, of solution cost, expressed as the percentage by which the cost exceeds the optimal. The columns \( \mu_t \) and \( \sigma_t \) indicate the mean and standard deviation, respectively, of CPU time, as measured by the Linux \text{getrusage} function on a 2.83 GHz Intel processor. Fig. 1 shows a plot of mean solution cost \( \mu_c \) versus the value of \( J \).

We can observe from the table and the figure that all values of \( J \) produce better results, in both solution cost and CPU time, than the standard model (\( J = 1 \)), although the best results are obtained with \( J \) in the range 200 to 400.

To determine the level of statistical significance of the solution cost results, we apply a non-parametric (two-tailed) Wilcoxon rank-sum test to the results of each experiment. For each experiment with a non-unity value of \( J \), the null hypothesis is that there is no difference between the non-unity setting of \( J \) and the setting \( J = 1 \) (which corresponds to the standard model). The results of these tests are reported in the last two columns of Table 2. One of these columns indicates the computed \( W \) statistic, and the other indicates whether the computed \( W \) statistic is significant (for \( p = 0.001 \)). We can see from the table that the results are significant for all values of \( J > 1 \).
Table 1: Parameter values used in experiments

| Parameter                              | Value |
|----------------------------------------|-------|
| $\alpha$                               | 1     |
| $\beta$                                | 2     |
| $\rho$                                 | 0.2   |
| length of nearest-neighbor list        | 25    |
| type of local search                   | 2-opt |
| number of ants                         | 50    |
| number of iterations                   | 200   |
| number of trials                       | 30    |
| $W$                                    | 6     |

Table 2: Experimental results

| $\gamma$ | $\mu_e$  | $\sigma_e$ | $\mu_t$  | $\sigma_t$ | $W$ | sig.? |
|----------|----------|------------|----------|------------|-----|-------|
| 1        | 6.22     | 0.16       | 32.1     | 0.11       | --- | ---   |
| 2        | 5.62     | 0.15       | 30.9     | 0.15       | 468 | yes   |
| 5        | 4.72     | 0.17       | 29.2     | 0.25       | 465 | yes   |
| 10       | 4.11     | 0.19       | 28.1     | 0.37       | 465 | yes   |
| 50       | 2.24     | 0.38       | 22.9     | 1.03       | 465 | yes   |
| 100      | 2.12     | 0.55       | 21.0     | 1.77       | 465 | yes   |
| 200      | 1.35     | 0.61       | 17.1     | 2.73       | 465 | yes   |
| 300      | 1.48     | 0.67       | 17.0     | 3.01       | 465 | yes   |
| 400      | 1.32     | 0.83       | 17.3     | 3.12       | 465 | yes   |
| 500      | 1.41     | 0.77       | 17.4     | 3.19       | 465 | yes   |
| 600      | 1.59     | 0.75       | 17.3     | 3.31       | 465 | yes   |
| 700      | 1.42     | 0.75       | 17.3     | 3.15       | 465 | yes   |
| 800      | 1.55     | 0.79       | 17.7     | 3.35       | 465 | yes   |
Fig. 1: Plot of mean solution cost (y-axis) versus value of $\gamma$ (x-axis)

Fig. 2: Plot of mean solution cost (y-axis) versus iteration number (x-axis), for several selected values of $\gamma$
In order to examine the evolution of solution cost over time, we computed the following. For each iteration \( t \), we computed the mean solution cost at iteration \( t \), aggregated over the 30 trials. This is shown in Fig. 2 for selected values of \( \gamma \). In this figure, the x-axis represents the iteration number, and the y-axis represents the mean solution cost at that iteration. We can see from the figure that performance improves consistently as \( \gamma \) increases from 1 up to 100, but there is not a large difference in performance for \( \gamma \) ranging from 200 to 800.

5. Concluding Remarks

Stubborn ants are an ACO variation that can be applied in combination with most ACO models. In previous work, we showed that stubbornness can significantly improve the performance of MAX-MIN Ant System on the TSP. In this paper, we combined stubborn ants with Ranked Ant System, and found that performance improved to a statistically significant extent. We also found that the performance gain is fairly robust over a wide range of settings of the stubbornness \( \gamma \) parameter.

Acknowledgements

Partial support for this work was provided by the National Science Foundation, the Missouri S&T Intelligent Systems Center, and the Mary K. Finley Missouri Endowment. We would like to thank Thomas Stützle for valuable advice and guidance, and for useful discussions.

References

1. A.M. Abdelbar, Stubborn ants, *Proceedings IEEE Swarm Intelligence Symposium (SIS-08)*, 2008, pp. 1-5.
2. A.M. Abdelbar, and D.C. Wunsch, Improving the performance of MAX-MIN ant system on the TSP with stubborn ants, *Proceedings Genetic and Evolutionary Computation Conference (GECCO-12)*, 2012, to appear.
3. J.L. Bentley, Fast algorithms for geometric traveling salesman problems, *ORSA Journal on Computing*, Vol. 4, No. 4, 1992, pp. 387-411.
4. B. Bullnheimer, R.F. Hartl, and C. Strauss, An improved ant system algorithm for the vehicle routing problem, *Annals of Operations Research*, Vol. 89, 1999, pp. 319-328.
5. O. Cordón, I.F. de Viana, and F. Herrera, Analysis of the best-worst ant system and its variants on the TSP, *Mathware and Soft Computing*, Vol. 9, No. 2-3, 2002, pp. 177-192.
6. M. Dorigo, V. Maniezzo, and A. Colomni, Ant system: optimization by a colony of cooperative agents, *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 26, 1996, pp. 29-41.
7. M. Dorigo, and L.M. Gambardella, Ant colony system: a cooperative learning approach to the traveling salesman problem, *IEEE Transactions on Evolutionary Computation*, Vol. 1, No. 1, 1997, pp. 35-66.
8. M. Dorigo and T. Stützle, *Ant Colony Optimization*, MIT Press, Cambridge, MA, USA, 2004.
9. M. Dorigo, and T. Stützle, Ant colony optimization: overview and recent advances, IRIDIA Technical Report 2009-013, Université Libre de Bruxelles, Brussels, Belgium, 2009.
10. T. Stützle, ACOTSP, Version 1.0, Available from http://www.aco-metaheuristic.org/aco-code, 2004.
11. T. Stützle, and H.H. Hoos, MAX-MIN ant system, *Future Generation Computer Systems*, Vol. 16, 2000, pp. 889-914.