Extended Trail Reinforcement Strategies for Ant Colony Optimization

Nikola Ivkovic[1], Mirko Malekovic[1], and Marin Golub[2]

[1] Faculty of Organization and Informatics, University of Zagreb
{nikola.ivkovic,mirko.malekovic}@foi.hr
[2] Faculty of Electrical Engineering and Computing, University of Zagreb
marin.golub@fer.hr

Abstract. Ant colony optimization (ACO) is a metaheuristic inspired by the foraging behavior of biological ants that was successfully applied for solving computationally hard problems. The fundamental idea that drives the ACO is the usage of pheromone trails for accumulating experience about the problem that is been solved. The best performing ACO algorithms typically use one, in some sense "the best", solution to reinforce trail components. Two main trail reinforcement strategies are used in ACO algorithms: iteration best and global best strategy. This paper extends the reinforcement strategies by using the information from an arbitrary number of previous iterations of the algorithm. The influence of proposed strategies on algorithmic behavior is analyzed on different classes of optimization problems. The conducted experiments showed that using the proposed strategies can improve the algorithm's performance. To compare the strategies we use the Mann–Whitney and Kruskal – Wallis statistical tests.

Keywords: reinforcement strategy, pheromone trail, MAX-MIN ant system, Ant colony optimization, Swarm intelligence, combinatorial optimization, parameter settings.

1 Introduction

Ant colony optimization (ACO) [1], [2] is a class of algorithms inspired by a foraging behavior of biological ants. The colony of ants searches a surrounding area for food sources. The ants that found a food leave a pheromone trail on its way back. This way, the ants communicate indirectly with the rest of the colony by modifying a surrounding environment. The other ants then follow the pheromone trails to a food source, and leave its own trails on the way back to the colony. The shorter paths are reinforced more often, and this attracts more ants, causing an autocatalytic effect. After some time, the most of ants use the shortest path. The pheromone trails laid on the longer paths eventually evaporates.

The ACO algorithms use artificial ants to construct solutions using solution components. The solution components are linked with artificial pheromone trails that affect the solution construction process. The trails encompass the collective knowledge, based on the experience of the colony about the problem. At the end of iteration the components that constitute good solutions are reinforced.

The ACO metaheuristic is successfully applied on a variety of hard computational problems. The first ant based algorithm, the Ant system (AS) [3], uses all solutions constructed in the previous iteration to reinforce the trails, but the trails associated with better solutions are reinforced more than lower quality solutions. After the AS, a number of ACO algorithms with improved performance on different combinatorial problems were published, and the most of them use only one solution, in some sense the best one, for trails reinforcement. The Ant colony system (ACS) [4] uses relatively greedy strategy and reinforces the trails of the global best solution (also known as best-so-far), i.e. the best solution that was constructed from the beginning of the algorithm. One of the most successful and popular [5] ACO algorithm, MAX-MIN ant system (MMAS) [6] usually uses the best solution constructed in the current iteration, but for bigger problems it is recommended to use the global best strategy. Considering that standard strategies are diametrically opposite, in this paper we propose new strategies that allow a finer control between explorativity of the iteration best and greediness of the global best strategy. The proposed strategies are not concerned with the pheromone values to be added on the trails and are also applicable when the amount of additional pheromone changes as the algorithm progresses [7].

This paper is organized as following. Section 2 briefly explains the ACO algorithm. In the section 3, the new strategies for selecting a solution for trails reinforcement procedure are introduced and compared. The section 4 briefly presents optimization problems used in the experimental evaluations of the new strategies and explains conditions under which the experiments were conducted. The section 5 presents and analyzes the results of the experimental researches, and section 6 gives final conclusions.

2 Ant Colony Optimization and MAX-MIN Ant System

Ant colony optimization can be described with the preudocode written in the table 1. In the `Initialize()` procedure the MMAS sets all trails to a maximum value and also the parameters of the algorithm are set. After that, the algorithm runs iteratively until satisfactory good solution is found or predefined time or number of iterations elapses. Solutions are constructed in the `ConstructSolutions()` procedure by adding solution components in the list of components until entire solution is constructed. The probability of selecting *i-th* component $c(i)$ from the set of components L_i is given for MMAS by expression (1). Parameters α and β balance between pheromone trail $\tau_c(i)$ and heuristic value $\eta_c(i)$. Update procedure includes trails evaporation using (2) for all trails, and trails reinforcement (3) for components included in the iteration best or global best solution. In the MMAS trails are maintained within a minimum and a maximum limits. The parameter ρ is a trail evaporation rate, and the $f(S^{best})$ gives a goodness of the solution used in the trails reinforcement procedure.

Table 1. The ACO algorithm and the formulas used in the MMAS

The ACO preudocode	Formulas used in the MMAS

`Initialize()`
`DO until stop conditions are met`
`  ConstructSolutions()`
`  UpdateTrails()`
`NEXT ITERATION.`
	$p_{c(i)} = \frac{\tau_{c(i)}^{\alpha} \cdot \eta_{c(i)}^{\beta}}{\sum_{k \in L_i} \tau_k^{\alpha} \cdot \eta_k^{\beta}}$ (1)
	$\tau_{c(j)} = (1-\rho) \cdot \tau_{c(j)}$ (2)
	$\tau_{c(k)} = \tau_{c(k)} + \frac{1}{f(s^{best})}$ (3)

3 Extended Reinforcement Strategies

The first extended strategy, the κ-best strategy, is defined for any natural number *κ*. The best solution constructed in the previous *κ* iterations is used for reinforcement of the trails. Formally, the κ-best solution for iteration *i* can be defined (for minimization problems) with expression (4).

$$KappaBestSolution_i = \min_{0 \le k \le \min\{i,\kappa\}} \{IterationBestSolution_{i-k}\} . \quad (4)$$

Before the algorithm reaches the iteration *κ*, the best solution of all constructed solutions is used for the reinforcement. For implementation of this strategy it is necessary to save up to *κ* solutions from the previous iterations. In the every iteration, it is necessary to save one new solution, and to delete the oldest saved solution. Therefore, it is convenient to use queue – like structure, which also allows reading all solution in the structure, when searching for the best solution.

The second strategy, the κ-best strategy, is a kind of approximation of the κ-best strategy that uses solutions from at most *κ* previous iterations. Initially, the best solution from the first iteration is set as a κ-max-best solution. In an every following iteration, the iteration best solution is saved as the κ-max-best solution if it is better than the previously saved one or if the κ-max-best solution has not been updated for previous *κ* iterations. The method of selecting a κ-max-best solution is described with the preudocode:

```
counter = counter + 1
IF ib "is better than" kappaMaxBest OR counter >= kappa DO
      counter = 0
      kappaMaxBest = ib
END IF
```

Initially, the `counter` variable is set to 0 and the best solution constructed in the first iteration is saved in the `kappaMaxBest` variable. The iteration best solution is saved in the `ib` variable.

A time complexity for the κ-best strategy is *O*(*κ*), since it is necessary to search the list of up to *κ* elements, and *κ*-max-best strategy has *O*(1) complexity. It can be shown that κ-best and κ-max-best strategies are generalization of standard strategies. For κ-best strategy, if the κ is equal or greater than the maximum iteration (*MAXITER*), then the solutions from all iterations are considered when searching for the best one, which is equivalent to the global best strategy. Also, for the κ-max-best strategy `counter` cannot exceed the *MAXITER*, so only a global best solution can be stored in the `kappaMaxBest` variable, if kappa>=*MAXITER*. Also, at the beginning of the algorithm, when kappa<= current iteration, all κ-best and κ-max-best exhibit an equivalent behavior to the *gb* strategy. The table 2 summarizes general equivalences between different strategies.

Table 2. The special cases equivalences between strategies

Standard strategy	κ-best strategy	κ-max-best strategy
iteration best	1-best	1-max-best
global best	∞-best	∞-max-best

4 Experimental Settings

The experimental evaluations of proposed strategies were conducted on Quadratic Assignment Problem (QAP), Travelling Salesman Problem (TSP) and Asymmetric Travelling Salesman Problem (ATSP), all well known NP-hard optimization problems that arise in many practical applications. The TSP has a set of cities (nodes) and all distances (edges) between cities are known. The problem is to find a closed tour with a minimum total distance. If the edges have directions the problem is called asymmetrical, otherwise the problem is called symmetrical. Is general TSP term should include asymmetrical and symmetrical variants, but more often TSP denotes symmetrical variant. For QAP there is a set of facilities and an equally sized set of locations with defined flow weights between the facilities and distances between the locations. The problem is to allocate the facilities to the locations in a way that the sum of products of flow weights and distances are minimized. In this paper all ATSP and TSP problems used in the empirical studies are from the *TSPLIB* library, publicly accessible at *http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/* and *VLSI Data Set* accessible at *http://www.tsp.gatech.edu/vlsi/*.

All used QAP problems are from the *QAPLIB* library publicly accessible at *http://www.seas.upenn.edu/qaplib/*.

The experiments were conducted on 18 ATSP, 19 TSP and 55 QAP problems. For every problem instance, 16 different reinforcement strategies were tested, and the experiment was repeated 100 times. Algorithm was executed with parameters: ρ=1, ρ=0.02, and number of ants was set equal to a problem size. The parameter β was set to 4 for TSPs and ATSPs, and for QAPs it was set to 0. For ATSP and TSP problems with size less than 100, the algorithm was allowed to perform the maximal 1000 iterations. For greater sized problems and all QAP problems, 10000 iterations were allowed. The experiments were performed in parallel on different processors inside *Beowulf* type cluster.

5 Results and Discussion

In the field of computational intelligence it is common to use parametric statistics, which relay on the assumption that data came from some known distribution although these assumptions are often violated [8]. By examining distributions for particular experiments we found that in some cases the distributions are very different from the standard distribution. Consequently, we use nonparametric statistics for data analysis and median instead of mean as a measure of centrality. One convenient characteristics of median is that at least 50% of samples are less than or equal to the median. It can be expected that the algorithm will find a solution with at most median value with at least 50% probability if the algorithm is executed once or at least with 75% probability if it is executed twice, etc.

By examining the functional dependencies of a median solution in the particular iterations for all tested problems (92 problems) we noticed following. The choice of the optimal strategy depends on a problem that is being solved an also on the maximal allowed number of iterations. The typical examples are presented on Fig. 1 (a)-(c).

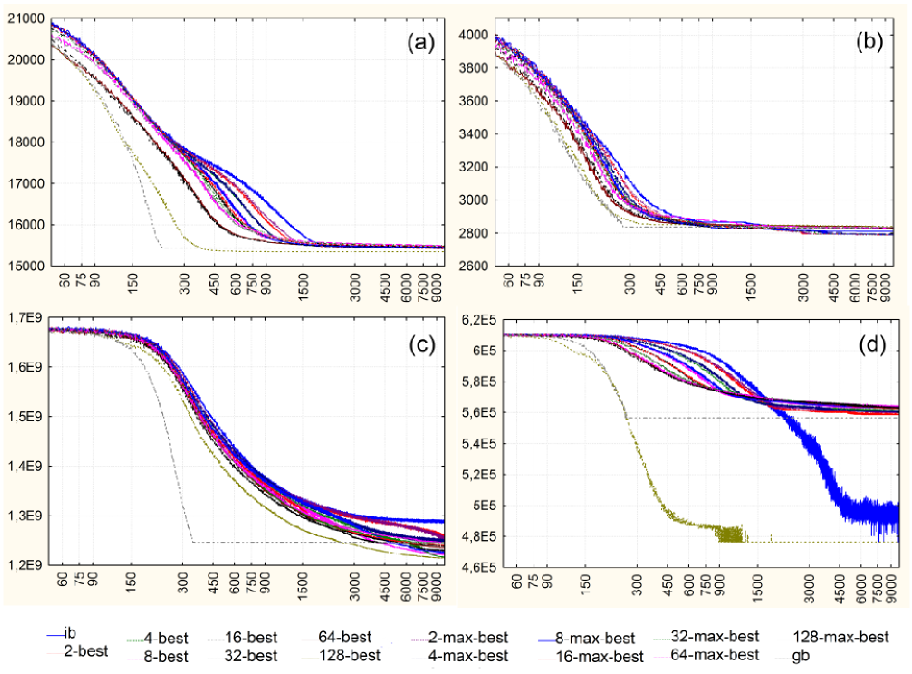

Fig. 1. Median iteration best solution on the axis ordinate, as a function of iteration, for different strategies for: rd400 TSP problem (a), ftv17 ATSP problem (b), and QAP problems tai100b (c) and lipa40b (d). The scale on the abscissa axis is logarithmic.

At the beginning, all strategies cause similar algorithmic behavior because differences in the pheromone trails are still small. Also, some strategies have equivalent behavior on the beginning. After that, performance of the strategies differs more considerably. Greedier strategies typically find good solutions faster than less greedy strategies. For problems that are relatively easy for MMAS algorithm, greedier strategies often find an optimal solution relatively fast and less greedy strategies catch with them later. In that case, greedier strategies are the optimal choice regardless of the iteration. For harder problems, more often less greedy strategies outrun the greedier strategies after some number of iterations, so the choice of the optimal strategy depends on the number of iterations.

For rd400 problem (Fig. 1a), the *gb* strategy gives the best results for roughly 150-350 iterations and then it is outrun with 128-best strategy that gives the best solutions from 350-10000 iterations. For ftv170 (Fig. 1b) at first, *gb* and 128-best give the best results, but at the end, the best results are achieved by *ib*, 2-best, 4-best and 2-max-best strategies, while the most greedy strategies achieves the worst results. The *gb* strategy greatly outstands as the best strategy for tai100b (Fig 1c) at the beginning of the algorithm. This is typical for QAP problems and much more noticeable than with TSPs and ATSPs, possibly because the MMAS does not use the heuristic values for QAPs. After about 2500 iterations 128-best strategy outrun the *gb* strategy and stay the best one until the end of 100000 iterations. At the end, 4-best strategy comes close to 128-best as the second best strategy.

In the case when the difference between strategies is the most noticeable, the bigger κ parameter means more successful κ-best strategy: The same applies for the κ-max-best strategy. For lower κ values, the performance of κ-best and κ-max-best strategies is similar, but for bigger κ, i.e. 64 and 128, the κ-best is noticeably better than the κ-max-best.

As an exception from other tested problems, for lipa40b (Fig 1d) the MMAS algorithm shows rather unexpected behavior. At first, the *gb* and 128-best achieves the best solutions, but after about 300 iterations the *gb* strategy stagnates and 128-best keeps finding better solutions relatively fast. After while, *ib* strategy starts finding better solutions relatively fast, and at the end, the both 128-best and *ib* strategies have median solutions equal to the optimal one, while the other strategies achieve considerably poorer solutions.

To compare κ-best and κ-max-best strategies for all tested problems at ones, we divided values of the achieved solutions with the values of the best known solutions for particular problems and run Mann–Whitney U test. The test was run for κ-best and κ-max-best pairs after 300, 1000, 3000 and 10000 iterations (for TSP and ATSP only for 300 and 1000 iterations, since for problems with size up to 100 maximum number of iteration was limited to 1000). For QAP problems, the Mann–Whitney test confirmed with very high significance ($p<0.000001$) that 128-best have a better performance than the 128-max-best strategy. Also, it was confirmed with the high significance ($p<0.01$) that 64-best is a better than 64-max-best. For other κ values, the test could not confirm that κ-best and κ-max-best exhibit the significant difference in the performance ($p>0.5$ and is often rather close to 1). The test confirmed with the high significance ($p<0.01$) that the κ-best is better than the κ-max-best for the ATSP for all κ values (except 1 and ∞), and for the TSP for κ values greater than 8.

To compare the performance of different strategies we use Kruskal-Wallis *H* test. The strategies with highest rank score are presented with scatterplots on Fig. 2.

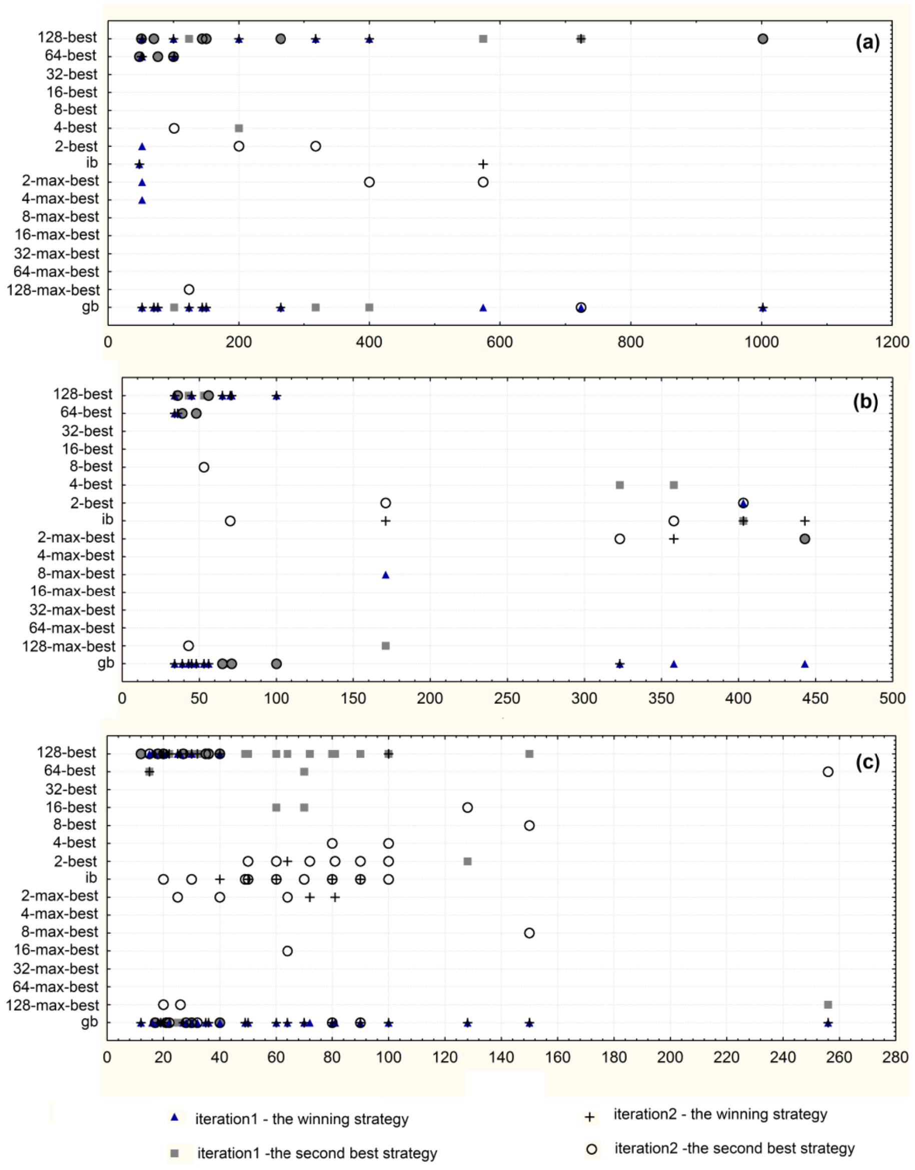

Fig. 2. The winning and the second best strategies as a result of Kruskal-Wallis statistical test for TSP (a), ATSP (b) and QAP (c). The problem size is denoted on the axis abscise.

For few easiest problems (e.g. ftv17), the algorithm found the optimal solutions very fast with all strategies. In these cases, the Kruskal-Wallis test had low H value and high p value. Consequently, these problems were excluded from the scatterplots. The *iteration1* for ATSP and TSP is equal to 300 iterations for problems with size up to 100, and 1000 iterations for bigger problems. The *iteration2* is set to 1000iterations for smaller problems and 10000 for bigger problems. This is motivated by the fact that for smaller problems the algorithm coverage faster. For the QAP problems, the size of the problem seems less related to problem's hardness and a speed of convergence, so for all problems *iteration1* and *iteration2* are set to 1000 and 10000 iterations, respectfully. The results clearly show that in many cases the extended strategies are better than the standard strategies. For smaller sized problems, 128-best and *gb* strategies often give the best results. For bigger problems it seems that 128-best and *gb* give the best results at the beginning, but if algorithm is allowed to run for a longer time, strategies with lower κ outperform the greedier strategies. The middle value κ gives moderate results regardless of the iteration, but are also rarely the worst strategies.

6 Conclusion

The proposed new strategies provide a fine control of the greediness through the κ parameter. The paper analyzes an influence of the strategies on the algorithmic behavior. The statistical tests conducted on experimental data confirmed with a high significance that the performance of the ACO algorithm can be enhanced by proposed strategies. For bigger κ values the κ-best strategy has a better performance than the κ-max-best strategy, but for lower κ values the difference in not significant.

References

1. Dorigo, M., Di Caro, G.: Ant colony optimization: A new meta-heuristic. In: Angeline, P.J., et al. (eds.) Proceedings of the Congress on Evolutionary Computation, vol. 2, pp. 1470–1477. IEEE Press (1999)
2. Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant Algorithms for Discrete Optimization. Artificial Life 5(2), 137–172 (1999)
3. Dorigo, M., Maniezzo, V., Colorni, A.: The Ant System: Optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics – Part B 26, 29–42 (1996)
4. Dorigo, M., Gambardella, L.M.: Ant Colony System: A Cooperative Learning Approach to the Travelling Salesman Problem. IEEE Transactions on Evolutionary Computation 1(1), 53–66 (1997)
5. Dorigo, M.: Ant colony optimization. Scholarpedia 2(3), 1461, revision #82084 (2007)
6. Stützle, T., Hoos, H.H.: MAX–MIN Ant System. Future Generation Computer Systems 16(9), 889–914 (2000)
7. Abraham, A., Konar, A., Samal, N.R., Das, S.: Stability analysis of the ant system dynamics with non-uniform pheromone deposition rules. IEEE Congress on Evolutionary Computation, 1103–1108 (2007)
8. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation 1(1), 3–18 (2011)