Ant Colony Optimization with a Genetic Restart Approach toward Global Optimization

G. Hossein Hajimirsadeghi[1], Mahdy Nabaee[1], Babak N. Araabi[1]

[1]Center of Excellence for Control and Intelligent Processing,
Electrical and Computer Engineering Faculty,
College of Engineering, University of Tehran, Tehran, Iran
{h.hajimirsadeghi, m.nabaee}@ece.ut.ac.ir, araabi@ut.ac.ir

Abstract. Ant Colony Optimization (ACO) a nature-inspired metaheuristic algorithm has been successfully applied in the traveling salesman problem (TSP) and a variety of combinatorial problems. In fact, ACO can effectively fit to discrete optimization problems and exploit pre-knowledge of the problems for a faster convergence. We present an improved version of ACO with a kind of Genetic semi-random-restart to solve Multiplicative Square Problem which is an ill-conditioned NP-hard combinatorial problem and demonstrate its ability to escape from local optimal solutions. The results show that our approach appears more efficient in time and cost than the solitary ACO algorithms.

Keywords: Ant Colony Optimization, NP-hard combinatorial problems, Multiplicative Squares, Heuristic Optimization, Random-Restart Algorithms, Genetic Algorithms, Ill-conditioned Problems.

1 Introduction

Ant algorithms are a class of population-based metaheuristic algorithms for solving Combinatorial Optimization problems. Ant Colony Optimization (ACO) is biologically inspired from the foraging behavior of real ants. ACO is an iterative process in which repeatedly, probabilistic candidate solutions are constructed by heuristic knowledge of the problem and pheromone trails as communication mediums. The main points of ACO are distributed computation, positive feedback and greedy construction heuristics. After the first ACO algorithm proposed by Dorigo (1992) [1], different types of ACO have been developed, most pursuing new ways of exploration and exploitation. Moreover, the combination of ACO and local search algorithms has led to successful results and obtained better performance on variety of problems. To date, ACO has been applied in many combinatorial problems, including Traveling Salesman Problem (TSP), quadratic assignment, vehicle routing, graph coloring, routing for telecommunication networks, sequential ordering, scheduling, data mining, and so on.

In this paper, we introduce an improved version of ACO to maximize the score of Multiplicative Squares (MS). The maximum version of MS is a Square such that sum of the products of its rows, columns, diagonals, and broken diagonals is maximum.

It's a complicated problem, because a precision of 20+ digits is needed for the dimensions greater than 10. So a very defiant and crafty algorithm must be applied to the problem. A genetic inspired random-restart is added to the ACO algorithm as a survivor approach to refrain from local maxima.
In section 2, Multiplicative Square is introduced, while optimization algorithms, including ACO and local search methods are described in section 3. Next in section 4, our methodology is explained and the results are summarized in section 5. Finally, conclusions are drawn in section 6.

2 Problem Description of Multiplicative Squares

The Multiplicative Squares are a class of squares filled by the numbers 1 to n^2 that the products of their rows, columns diagonals, and broken diagonals have a special feature. The most well-known type of MS is Kurchan square posed by Rodolfo Kurchan (1989), which is originated from magic square. The maximum product minus the minimum one is as small as possible in a Kurchan Square. However, in the MAX version of MS, sum of the following products is maximum. The score function of a MS of dimension 3 is illustrated in Fig. 1.

Rows: 5*1*8 = 40, 3*9*4 = 108, 7*2*6 = 84
Columns: 5*3*7 = 105, 1*9*2 = 18, 8*4*6 = 192
Diagonals: 5*9*6 = 270, 1*4*7 = 28, 8*3*2 = 48
Anti-diagonals: 8*9*7 = 504, 1*3*6 = 18, 5*4*2 = 40
MAXMS: SF= 40+108+84+105+18+192+270+28+48+504+18+40= 1455
Kurchan MS: SF= 504-18 = 486

5	1	8
3	9	4
7	2	6

Fig. 1. An example of a 3*3 multiplicative square and Score Function (SF) evaluation for

MS problem is an ill-conditioned NP-hard problem in which a small change of indexes may cause larger errors. Therefore, escaping from local optimums seems to be hard in larger dimensions that more precision and more exploration are needed.

In MAX MS, it can be concluded that the greater numbers should be in the same row, column, diagonal, or broken diagonal, so that the products and at last the score will be greater. We use this feature in our approach that will be described in section 4.

3 Optimization Algorithms

3.1 Ant System

The first ACO algorithm called Ant System applied to Traveling Salesman Problem (TSP) by Dorigo. AS makes up the main framework of other ACO algorithms and is considered as a prototype. In TSP each of m artificial ants generates a complete tour by a probabilistic rule (1), which is the probability that ant k in city i visits city j.

$$p_{i,j}^{k} = \begin{cases} \frac{[\tau_{i,j}]^{\alpha}.[\eta_{i,j}]^{\beta}}{\sum_{l \in N_i^k}[\tau_{i,l}]^{\alpha}.[\eta_{i,l}]^{\beta}}, & \forall j \in N_i^k \\ 0, & otherwise \end{cases} \quad (1)$$

Where τ is pheromone, $\eta_{i,j}$ is heuristic function and is equal to $\frac{1}{d_{i,j}}$ the inverse of the difference between city i and j, N_i^k is the set of cities that haven't been visited by ant k, α and β are parameters which shows the relative importance of pheromone versus heuristic or exploitation versus exploration.

Equation (1) shows that ants prefer paths with shorter length and higher amount of pheromone, so they independently generate tours by pre-knowledge of the problem and cooperative informative communication. Once all the ants complete their tours the pheromone trails updates, using (2) and (3).

$$\tau_{i,j} = (1-\rho).\tau_{i,j} + \sum_{k=1}^{m} \Delta\tau_{i,j}^{k} \quad (2)$$

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{Q}{L_k}, & (i,j) \in \text{tour done by ant k} \\ 0, & \text{otherwise} \end{cases} \quad (3)$$

Where ρ is evaporation rate, L_k is the length of tour taken by ant k, Q is a constant, and m is the number of ants.

3.2 ACO Algorithms

After Ant System, Researchers started to improve the performance of ACO. A first improvement of ACO was elitist strategy (AS_{elite}) [2], which was simply considered more emphasis on the global-best tour. Another improvement was AS_{rank} as an offspring of AS_{elite}, proposed by Bullnheimer, Hartl and Strauss [7]. It sorts the ants and then the trails are updated by only the first $\omega - 1$ ants according to (4).

$$\tau_{i,j} = (1-\rho).\tau_{i,j} + \sum_{k=1}^{\omega-1}(\omega-1).\Delta\tau_{i,j}^{k} + \omega.\ \Delta\tau_{i,j}^{gb} \quad (4)$$

Where $\Delta\tau_{i,j}^{k} = \frac{1}{L_k}$ and $\Delta\tau_{i,j}^{gb} = \frac{1}{L_{gb}}$.

Stüzle and Hoos introduced MAX-MIN Ant System (MMAS) [8]. In MMAS trails are limited to an interval $[\tau_{min}, \tau_{max}]$, so it help ants not to converge to local optimum. Further, in MMAS, only the best ant (iteration-best or global-best) is allowed to deposit pheromone. Sometimes, for more exploration an additional mechanism called Pheromone Trail Smoothing is applied to MMAS.

Gambardella and Dorigo in 1996 proposed Ant Colony System (ACS) [11], [3], which was a simplified version of Ant-Q. Ant-Q is a link between reinforcement learning and Ant Colony Optimization. However, ACS simply and more efficiently describes the same behavior as Ant-Q. Two strategies are used in ACS to increase the previous information exploitation. At first, trails are updated by the best ant, like

MMAS, and secondly, ants select the next city, using a *pseudo-random proportional rule* [11]. The rule states that with probability q_0 the city j is selected, where $j = \arg\max_{j \in N_i^k}\{[\tau_{i,j}]^{\alpha}.[\eta_{i,j}]^{\beta}\}$, while with the probability $1 - q_0$ a city is chosen using (1). Furthermore, there is a distinct difference between ACS and other ACO algorithms and that is trails are updated, while the solutions are built. It's similar to ant-quantity and ant-density approaches that update pheromone trails synchronic to making tours. However, in ACS ants eat portion of the trails as they walk on the path. So the probability that the same solutions are constructed in an iteration decreases.

AS Local Best Tour (AS-LBT) [6], is another improved kind of AS, in which only local information is used to reinforce trails. It means that each ant updates its trail by the best tour it has found to date. This approach shows more diversity than AS.

Some other improvements in the field of ACO are the Multiple Ant Colonies Algorithms [9], which exploits interactions between colonies, Population-based ACO (P-ACO), which makes up a special population of good solutions, and Omicron ACO (OA), which is inspired by MMAS and elitist strategy.

In addition a number of hybrid algorithms have been developed that use good features of ACO. For example the combination of Genetic Algorithm (GA) and ACO, called Genetic Ant Colony Optimization (GACO) have been used to solve different combinatorial problems [4], [5].

Moreover, ACO algorithms often exploit Local Search to improve their performance which is explained in the next section.

3.3 Local Search Algorithms

Local search is a greedy algorithm for solving optimization problems by exploring among candidate solutions. It starts with an initial solution and iteratively moves to neighbor solutions. If a better solution is found it will be replaced by the previous one and the procedure is repeated until no improving solution can be found.

The very simple version of local search is Hill Climbing, in which the first closest neighbor is selected to move. The other well-known local search algorithms are 2-opt and 3-opt.

There exist two crucial problems with local search algorithms that they are easily get trapped in local optimum, and finally their results are strictly dependent on initial solutions [10]. For the first problem some solutions have been devised, like random-restart strategies, while for the latter, heuristic and evolutionary algorithms like ACO can be used to generate appropriate initial solutions for local search algorithms [11].

4 Methodology

In this section we introduce our approach to solve the MAX MS problem. In ACO metaheuristic the main task is to find a graph representation for the problem so that ACO searches for a minimum cost path over the graph.

In each iteration, ants construct a candidate solution individually, going from the first layer to the last one. In Each layer a number between 1 to n^2 is selected which

has not been selected with the same ant before. These numbers are used as indices for feasible Multiplicative Squares. It means that the numbers n^2 to 1 are placed in the square respectively, according to indices generated by the ants.

Once the tours are completed ants deposit pheromones on the edges, using (5).

$$\Delta\tau_{i,j}^{k} = \begin{cases} \frac{FS_k}{Q}, & (i,j) \in tour\ done\ by\ ant\ k \\ 0, & otherwise \end{cases} \quad (5)$$

Where FS_k is some of the product of rows, columns, diagonals, and broken diagonals (or Function Score) of the square, constructed by ant k. Note that the ratio $\frac{FS_k}{Q}$ is inverse to one suggested in (3) because of the maximization case.

Here we suggest a heuristic function based on the feature, introduced in section 2. The heuristic function applied to each index according to a defined rule. When an ant is in an arbitrary index the heuristic function for the next step to any index in the same row, column, diagonal, or broken diagonal is λ and to other indices is μ which is less than λ. In this respect there is more probability for greater numbers to be multiplied and a larger FS is obtained. For example the mechanism has been shown for a 4 by 4 square in Fig. 2.

μ	λ	λ	λ
λ	λ	*	λ
μ	λ	λ	λ
λ	μ	λ	μ

(a)

λ	μ	λ	λ
λ	λ	λ	*
λ	μ	λ	λ
μ	λ	μ	λ

(b)

Fig. 2. Heuristic function is illustrated for two sample conditions. The current position of the ant is displayed by "*".

For ACO algorithm we used MMAS with a little difference. Actually In our method, the best ants, both iteration-best and global-best deposit pheromone and the heuristic function changes as the iteration increases. Adding iteration-best ants as the communicative elements are for more escape from so many local optimums in the problem, and global-best ants speed up the convergence. Parameter β decreases by the iterations and then increases in the last part of the process for modulating exploration during the search algorithm. Moreover, eating ants are used in the case of local optimum trap which are the components of ACS algorithm.

In the next step, ACO algorithms are accompanied by local search algorithms. In fact pheromone trails are updated by the local search solutions. In our approach the best tours (g-b and i-b) which are obtained in that iteration get improved by 2-opt local search.

As we said in section 3, local search algorithms may get stuck at a local optimum, so we pose a genetic semi-random restart that runs in an outer loop and endows new survivor initial solution to commence ACO and local search algorithms again.

In our genetic restart process, 2 parents reproduce 3 different children by a kind of cross over operator and each of the parents grants a child by mutation.

In cross over two break cuts are selected from 2 to $n^2 - 1$ randomly. Next, the block between these two numbers are chosen from the first parent and then moved to the right corner, left corner, or the same place of a new tour to devote 3 distinct children. The remaining vertices are filled with the other parent. An example of our cross over is depicted in Fig. 3.

Parent 1	1	**3**	**4**	2	5
Parent 2	4	5	1	2	3
Child 1	3	4	5	1	2
Child 2	5	1	2	3	4
Child 3	5	3	4	1	2

Fig. 3. An example of two cut cross over with 3 children.

In mutation, the block is remained for the parent 2 and the remainder is filled by random permutation. Just the same, the complement block of parent 1 is constant and the remnant is built randomly (Fig. 4). Hence, the two new children are different from the three previous ones, reproduced by cross over as much as possible.

Parent 1	1	**3**	**4**	2	5
Parent 2	4	**5**	**1**	2	3
Child of parent 1	1	4	3	2	5
Child of parent 2	2	5	1	4	3

Fig. 4. An example of a two cut mutation.

5 Experimentation and Results

To verify the efficiency of the proposed algorithm, it was employed on MS7 (7*7 grid) and MS8. Experimentally, We used parameter settings, $\alpha = 1$, initial value of $\beta = 3$, $\rho = .4$, $Q =$ the best SF found up to that iteration [6], eat rate = .9, $\lambda = 1$, and $\mu = .5$ for all experiments. In the case of MS7, the population size of about 50 (equal to the number of variables [2], [12]) ants was used, and the trails were set to interval, [0.002 , 2], with an initial value of 2. While for MS8, population size was set to 64, and trails were limited to $[\tau_{min}, \tau_{max}] = [.001\ ,\ 2]$ and initial value of 2.

10 trials were conducted, and all the tests were carried out for 600 iteration. The results are presented in Table 1. Flexible heuristic is our complete algorithm with β modulation described in previous section and genetic random restart, while fixed heuristic is the same as the first one without β modulation, and finally, No GA restart represents the algorithm without any restart process.

Table 1 shows a good performance of our algorithm specially compared with the same procedure without restart algorithm. Furthermore, the average number of incidents that the introduced Genetic restart algorithm granted new initial survivor solution to ACO algorithm is stated in Table 2.

Table 1. Experiment results

(a) MS7

Method	Best	Avg.	Std. Dev.	Std. Dev %	Best err.%	Avg. err.%
Flexible heuristic	836927418654	836545183884.3	310273380.3	0.037	0	0.046
Fixed heuristic	836864383934	836387896300.2	282729277	0.034	0.0075	0.064
No GA restart	836590536598	835890051299.2	472719981.5	0.057	0.0403	0.124

(b) MS8

Method	Best	Avg.	Std. Dev.	Std. Dev %	Best err.%	Avg. err.%
Flexible heuristic	402702517088866	402397450057731	410397887424.8	0.102	0	0.076
Fixed heuristic	402693316462602	396228893243407	12487304223038.1	3.15	0.0023	1.608
No GA restart	402672245516278	379411679729931	27191910644358.2	7.17	0.0075	5.784

Table 2. Genetic Semi-Random-Restart Performance

Method	Avg. number of successive genetic restart (MS7)	Avg. number of successive genetic restart (MS8)
Fixed heuristic	1.6	2.4
Flexible heuristic	1.3	2.3

To illustrate the operation of our Genetic semi-random-restart algorithm, trace of a particular run is demonstrated in Fig. 5. It shows that the posed restart mechanism improves the robustness and precision of the whole algorithm and efficiently helps to come out of the local optimums.

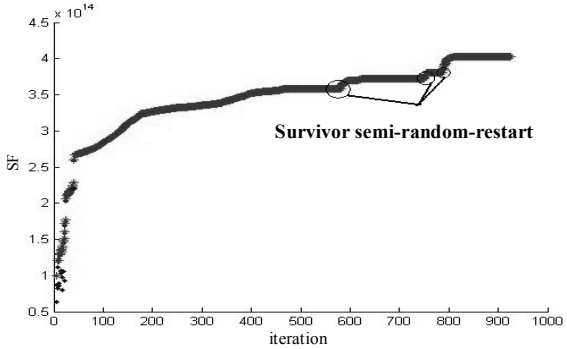

Fig. 5. Successful operation of the posed restart algorithm to evade local optimums.

6 Summary and Conclusion

This paper has introduced an improved version of ACO, with the aid of local search algorithms and specially a genetic restart algorithm, in order to global optimization. Max Multiplicative Square (MS) problem was studied as an ill-conditioned NP-hard combinatorial problem and a particular heuristic function was devised for that. Results have shown that our approach was successful to satisfy the goal of global optimization.

Further work can be in the direction of testing new random-restart techniques, in particular those which substantially differ from local search mechanism. About the MS problem, a better heuristic function can be a great step to decrease the time of evaluation. In addition, a new graph representation might be designed that more efficiently exploit the features of problem, such as symmetry and the importance of big numbers.

References

1. Dorigo, M.: Optimization, Learning and Natural Algorithms (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy (1992)
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, vol. 26, pp. 29--41 (1996)
3. Gambardella L.M. and Dorigo M.: Solving symmetric and asymmetric TSPs by ant colonies. In: Proceedings of the 1996 IEEE International Conference on Evolutionary Computation (ICEC'96), pp. 622--627. IEEE Press, Piscataway, NJ (1996)
4. Lee, Z.J.: A hybrid algorithm applied to travelling salesman problem. 2004 IEEE International Conference on Networking, Sensing and Control, vol.1, pp. 237--242 (2004)
5. Fu, T.P., Liu, Y.S., Chen, J.H.: Improved Genetic and Ant Colony Optimization Algorithm for Regional Air Defense WTA Problem. First International Conference on Innovative Computing, Information and Control (ICICIC'06), pp. 226--229 (2006)
6. White, T., Kaegi, S., Oda T.: Revisiting Elitism in Ant Colony Optimization. In: Genetic and Evolutionary Computation Conference (GECCO), LNCS, vol. 2723, pp. 122--133 (2003)
7. Bullnheimer, B., Hartl, R.F., Strauss, C.: A new rank-based version of the Ant System: A computational study. Central European Journal for Operations Research and Economics, vol. 7, pp. 25--38 (1999)
8. Stüzle, T., Hoos, H.: The MAX–MIN Ant System and local search for the traveling salesman problem. In: IEEE International Conference on Evolutionary Computation (ICEC'97), pp. 309--314. IEEE Press, Piscataway, NJ (1997)
9. Kawamura, H., Yamamoto, M., Suzuki, K., Ohuchi, A.: Multiple Ant Colonies Algorithm Based on Colony Level Interactions, IEICE Transactions, vol. E83-A, pp. 371–379 (2000)
10. Dorigo, M., Stüzle T.: The Ant Colony Optimization Metaheuristic: Algorithms, Application and Advances. Technical Report, IRIDIA-2000-32 (2000)
11. Dorigo M., Gambardella, L.M.: Ant Colony System: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, vol. 1, pp. 53--66 (1997)
12. Bonabeau, E., Dorigo, M., Theraulaz G.: Swarm Intelligence From Natural to Artificial Systems. Oxford University Press, New York NY (1999)