Proceedings of the International Multiconference on Computer Science and Information Technology pp. 887–891
ISBN 978-83-60810-22-4
ISSN 1896-7094

ACO with semi-random start applied on MKP

Stefka Fidanova
and Pencho Marinov
Institute for Parallel Processing
Bulgarian Academy of Sciences
Acad. G. Bonchev str. bl 25A
1113 Sofia, Bulgaria
Email: {stefka, pencho }@parallel.bas.bg

Krassimir Atanassov
Central Laboratory for Bio-Medical Engineering
Bulgarian Academy of Sciences
Acad. G. Bonchev str. bl 25A
1113 Sofia, Bulgaria
Email: krat@bas.bg

***Abstract*—Ant Colony Optimization (ACO) is a stochastic search method that mimics the social behavior of real ants colonies, which manage to establish the shortest route to feeding sources and back. Such algorithms have been developed to arrive at near-optimal solutions to large-scale optimization problems, for which traditional mathematical techniques may fail. On this paper semi-random start is applied. A new kind of estimation of start nodes of the ants is made and several start strategies are prepared and combined. The idea of semi-random start is better management of the ants. This new technique is tested on Multiple Knapsack Problem (MKP). Benchmark comparison among the strategies is presented in terms of quality of the results. Based on this comparison analysis, the performance of the algorithm is discussed. The study presents ideas that should be beneficial to both practitioners and researchers involved in solving optimization problems.**

I. Introduction

Many combinatorial optimization problems are fundamentally hard. This is the most typical scenario when it comes to realistic and relevant problems in industry and science. Examples of optimization problems are Traveling Salesman Problem [10], Vehicle Routing [12], Minimum Spanning Tree [8], Multiple Knapsack Problem [5], etc. They are NP-hard problems and in order to obtain solution close to the optimality in reasonable time, metaheuristic methods are used. One of them is Ant Colony Optimization (ACO) [3].

ACO algorithms have been inspired by the real ants behavior. In the nature, ants usually wander randomly, and upon finding food return to their nest while laying down pheromone trails. If other ants find such a path, they are likely not to keep traveling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food. However, as time passes, the pheromone starts to evaporate. The more time it takes for an ant to travel down the path and back again, the more time the pheromone has to evaporate and the path to become less prominent. A shorter path, in comparison will be visited by more ants and thus the pheromone density remains high for a longer time.

ACO is implemented as a team of intelligent agents which simulate the ants behavior, walking around the graph representing the problem to solve using mechanisms of cooperation and adaptation. ACO algorithm requires to define the following [1], [4]:

- The problem needs to be represented appropriately, which would allow the ants to incrementally update the solutions through the use of a probabilistic transition rules, based on the amount of pheromone in the trail and other problem specific knowledge. It is also important to enforce a strategy to construct only valid solutions corresponding to the problem definition.
- A problem-dependent heuristic function, that measures the quality of components that can be added to the current partial solution.
- A rule set for pheromone updating, which specifies how to modify the pheromone value.
- A probabilistic transition rule based on the value of the heuristic function and the pheromone value, that is used to iteratively construct a solution.

The structure of the ACO algorithm is shown by the pseudocode below (Figure 1). The transition probability $p_{i,j}$, to choose the node j when the current node is i, is based on the heuristic information $\eta_{i,j}$ and the pheromone trail level $\tau_{i,j}$ of the move, where $i, j = 1, \ldots, n$.

$$p_{i,j} = \frac{\tau^a_{i,j}\eta^b_{i,j}}{\sum_{k\in allowed} \tau^a_{i,k}\eta^b_{i,k}}, \tag{1}$$

The higher the value of the pheromone and the heuristic information, the more profitable it is to select this move and resume the search. In the beginning, the initial pheromone level is set to a small positive constant value τ_0; later, the ants update this value after completing the construction stage. ACO algorithms adopt different criteria to update the pheromone level.

The pheromone trail update rule is given by:

$$\tau_{i,j} \leftarrow \rho\tau_{i,j} + \Delta\tau_{i,j}, \tag{2}$$

where ρ models evaporation in the nature and $\Delta\tau_{i,j}$ is new added pheromone which is proportional to the quality of the solution.

Our novelty is to use estimations of start nodes with respect to the quality of the solution and thus to better manage the search process. On the basis of the estimations we offer several start strategies and their combinations. Like a benchmark problem is used Multiple Knapsack Problem (MKP) because a

```
Ant Colony Optimization
Initialize number of ants;
Initialize the ACO parameters;
while not end-condition do
    for k=0 to number of ants
        ant k choses start node;
        while solution is not constructed do
            ant k selects higher probability node;
        end while
    end for
    Update-pheromone-trails;
end while
```

Fig. 1. Pseudocode for ACO

lot of real world problems can be represented by it and MKP arises like a subproblem in many optimization problems.

The rest of the paper is organized as follows: in section 2 several start strategies are proposed. In section 3 the MKP is introduced. In section 4 the strategies are applied on MKP and the achieved results are compared and strategies are classified. At the end some conclusions and directions for future work are done.

II. Start Strategies

The known ACO algorithms create a solution starting from random node. But for some problems, especially subset problems, it is important from which node the search process starts. For example if an ant starts from node which does not belong to the optimal solution, probability to construct it is zero. Therefore we offer several start strategies.

Let the graph of the problem has m nodes. We divide the set of nodes on N subsets. There are different ways for dividing. Normally, the nodes of the graph are randomly enumerated. An example for creating of the nodes subsets, without loss of generality, is: the node number one is in the first subset, the node number two is in the second subset, etc. the node number N is in the $N-th$ subset, the node number $N+1$ is in the first subset, etc. Thus the number of the nodes in the subsets are almost equal. We introduce estimations $D_j(i)$ and $E_j(i)$ of the node subsets, where $i \geq 2$ is the number of the current iteration. $D_j(i)$ shows how good is the j^{th} subset and $E_j(i)$ shows how bad is the j^{th} subset. $D_j(i)$ and $E_j(i)$ are weight coefficients of $j-th$ node subset $(1 \leq j \leq N)$, which we calculate by the following formulas:

$$D_j(i) = \phi.D_j(i-1) + (1-\phi).F_j(i), \tag{3}$$

$$E_j(i) = \phi.E_j(i-1) + (1-\phi).G_j(i), \tag{4}$$

where $i \geq 1$ is the current process iteration and for each j $(1 \leq j \leq N)$:

$$F_j(i) = \begin{cases} \frac{f_{j,A}}{n_j} & \text{if } n_j \neq 0 \\ F_j(i-1) & \text{otherwise} \end{cases}, \tag{5}$$

$$G_j(i) = \begin{cases} \frac{g_{j,B}}{n_j} & \text{if } n_j \neq 0 \\ G_j(i-1) & \text{otherwise} \end{cases}, \tag{6}$$

$f_{j,A}$ is the number of the solutions among the best $A\%$, $g_{j,B}$ is the number of the solutions among the worst $B\%$, where $A+B \leq 100$, $i \geq 2$ and

$$\sum_{j=1}^{N} n_j = n, \tag{7}$$

where n_j $(1 \leq j \leq N)$ is the number of solutions obtained by ants starting from nodes subset j, n is the number of ants. Initial values of the weight coefficients are: $D_j(1) = 1$ and $E_j(1) = 0$. The parameter ϕ, $0 \leq \phi \leq 1$, shows the weight of the information from the previous iterations and from the last iteration. When $\phi = 0$ only the information from the last iteration is taken in to account. If $\phi = 0.5$ the influence of the previous iterations versus the last is equal. When $\phi = 1$ only the information from the previous iterations is taken in to account. When $\phi = 0.25$ the weight of the information from the previous iterations is three times less than this one of the last iteration. When $\phi = 0.75$ the weight of the previous iterations is three times higher than this one of the last iteration. The balance between the weights of the previous iterations and the last is important. At the beginning when the current best solution is far from the optimal one, some of the node subsets can be estimated as good. Therefore, if the value of the parameter ϕ is too high the estimation can be distorted. If the weight of the last iteration is too high then information for good and bad solutions from previous iterations is ignored, which can distort estimation too.

We try to use the experience of the ants from previous iteration to choose the better starting node. Other authors use this experience only by the pheromone, when the ants construct the solutions [4]. Let us fix threshold E for $E_j(i)$ and D for $D_j(i)$, than we construct several strategies to choose start node for every ant, the threshold E increases every iteration with $1/i$ where i is the number of the current iteration:

1 If $E_j(i)/D_j(i) > E$ then the subset j is forbidden for current iteration and we choose the starting node randomly from $\{j \mid j \text{ is not forbidden}\}$;
2 If $E_j(i)/D_j(i) > E$ then the subset j is forbidden for current simulation and we choose the starting node randomly from $\{j \mid j \text{ is not forbidden}\}$;
3 If $E_j(i)/D_j(i) > E$ then the subset j is forbidden for K_1 consecutive iterations and we choose the starting node randomly from $\{j \mid j \text{ is not forbidden}\}$;
4 Let $r_1 \in [0.5, 1)$ is a random number. Let $r_2 \in [0, 1]$ is a random number. If $r_2 > r_1$ we randomly choose node from subset $\{j \mid D_j(i) > D\}$, otherwise we randomly chose a node from the not forbidden subsets, r_1 is chosen and fixed at the beginning.
5 Let $r_1 \in [0.5, 1)$ is a random number. Let $r_2 \in [0, 1]$ is a random number. If $r_2 > r_1$ we randomly choose node

from subset $\{j \mid D_j(i) > D\}$, otherwise we randomly chose a node from the not forbidden subsets, r_1 is chosen at the beginning and increase with r_3 every iteration.

Where $0 \leq K_1 \leq$"number of iterations" is a parameter. If $K_1 = 0$, than strategy 3 is equal to the random choose of the start node. If $K_1 = 1$, than strategy 3 is equal to the strategy 1. If $K_1 =$"maximal number of iterations", than strategy 3 is equal to the strategy 2.

We can use more than one strategy for choosing the start node, but there are strategies which can not be combined. We distribute the strategies into two sets: $St1 = \{strategy1,\ strategy2,\ strategy3\}$ and $St2 = \{strategy5,\ strategy6\}$. The strategies from same set can not be used at once. Thus we can use strategy from one set or combine it with strategies from the other set. Exemplary combinations are $(strategy1)$, $(strategy2;\ strategy5)$, $(strategy3;\ strategy6)$. When we combine strategies from $St1$ and $St2$, first we apply the strategy from $St1$ and according it some of the regions (node subsets) become forbidden, and after that we choose the starting node from not forbidden subsets according the strategy from $St2$

III. Multiple Knapsack Problem

We test the ideas for controlled start on MKP. MKP is a real world problem and is a representative of the class of subset problems. The MKP has numerous applications in theory as well as in practice. It also arises as a subproblem in several algorithms for more complex problems and these algorithms will benefit from any improvement in the field of MKP. The following major applications can be mentioned: problems in cargo loading, cutting stock, bin-packing, budget control and financial management. Sinha and Zoltner [9] proposed to use the MKP in fault tolerance problem and in [2] is designed a public cryptography scheme whose security realize on the difficulty of solving the MKP. Martello and Toth [7] mention that two-processor scheduling problems may be solved as a MKP. Other applications are industrial management, naval, aerospace, computational complexity theory.

The MKP can be thought as a resource allocation problem, where there are m resources (the knapsacks) and n objects and every object j has a profit p_j. Each resource has its own budget c_j (knapsack capacity) and consumption r_{ij} of resource i by object j. The aim is maximizing the sum of the profits, while working with a limited budget.

The MKP can be formulated as follows:

$$\max \textstyle\sum_{j=1}^{n} p_j x_j$$

$$\text{subject to} \textstyle\sum_{j=1}^{n} r_{ij} x_j \leq c_i \quad i = 1, \ldots, m \tag{8}$$

$$x_j \in \{0, 1\} \quad j = 1, \ldots, n$$

x_j is 1 if the object j is chosen and 0 otherwise.

There are m constraints in this problem, so MKP is also called m-dimensional knapsack problem. Let $I = \{1, \ldots, m\}$ and $J = \{1, \ldots, n\}$, with $c_i \geq 0$ for all $i \in I$. A well-stated MKP assumes that $p_j > 0$ and $r_{ij} \leq c_i \leq \sum_{j=1}^{n} r_{ij}$ for all $i \in I$ and $j \in J$. Note that the $[r_{ij}]_{m \times n}$ matrix and $[c_i]_m$ vector are both non-negative.

In the MKP one is not interested in solutions giving a particular order. Therefore a partial solution is represented by $S = \{i_1, i_2, \ldots, i_j\}$ and the most recent elements incorporated to S, i_j need not be involved in the process for selecting the next element. Moreover, solutions for ordering problems have a fixed length as one search for a permutation of a known number of elements. Solutions for MKP, however, do not have a fixed length. The graph of the problem is defined as follows: the nodes correspond to the items, the arcs fully connect nodes. Fully connected graph means that after the object i one can chooses the object j for every i and j if there are enough resources and object j is not chosen yet.

IV. Computational Results

The computational experience of the ACO algorithm is shown using 10 MKP instances from "OR-Library" available within WWW access at **http://people.brunel.ac.uk/mastjjb/jeb/orlib/**, with 100 objects and 10 constraints. To provide a fair comparison for the above implemented ACO algorithm, a predefined number of iterations, $k = 100$, is fixed for all the runs. Thus we can observe which strategy reaches good solutions faster. If the value of k (number of iterations) is too high, the achieved results will be very close to the optimal solution and will be difficult to appreciate different strategies. We apply strategies on MMAS [11], because it is one of the best ACO approach. The developed technique has been coded in C++ language and implemented on a Pentium 4 (2.8 Ghz). The parameters are fixed as follows: $\rho = 0.5$, $a = 1$, $b = 1$, number of used ants is 20, $A = 30$, $B = 30$, $D = 1.5$, $E = 0.5$, $K_1 = 5$, $r_3 = 0.01$. The values of ACO parameters (ρ, a, b) are from [6] and experimentally is found that they are best for MKP. The tests are run with 1, 2, 4, 5 and 10 nodes within the nodes subsets and values for ϕ are 0, 0.25, 0.5 and 0.75. For every experiment, the results are obtained by performing 30 independent runs, then averaging the fitness values. The computational time which takes start strategies is negligible with respect to the computational time which takes solution construction.

Tests with all possible combinations of strategies and with random start (12 combinations at all), four value for ϕ and five kind of node subsets are run and every test 30 times. Thus the all runs are 72 000. One can observe that sometimes all nodes subsets become forbidden and the algorithm stops before performing all iterations (strategies 1, 2, 3 and combinations with them). So if all nodes subsets become forbidden the algorithm performs several iterations without any strategy with random start till some of the subsets become not forbidden. Then the algorithm continue to apply the chosen strategy.

The problem which arises is how to compare the achieved solutions by different strategies and different node-devisions. Therefore the difference (interval) d between the worst and best average result for every problem is divided to 10. If the

TABLE I
ESTIMATON OF STRATEGIES AND NODES DEVISIONS FOR $\phi = 0$

number nodes	10	5	4	2	1
random	32	32	32	32	32
strat. 1	84	84	87	83	83
strat. 2	33	31	36	53	74
strat. 3	79	86	86	88	86
strat. 4	86	86	86	86	86
strat. 5	86	86	86	86	86
strat. 1-4	83	89	84	81	89
strat. 1-5	83	89	84	81	89
strat. 2-4	33	36	35	53	82
strat. 2-5	33	36	35	63	82
strat. 3-4	69	89	88	87	**90**
strat. 3-5	69	89	88	87	**90**

TABLE II
ESTIMATON OF STRATEGIES AND NODES DEVISIONS FOR $\phi = 0.25$

number nodes	10	5	4	2	1
random	32	32	32	32	32
strat. 1	83	88	86	90	90
strat. 2	32	31	36	61	81
strat. 3	62	86	84	84	96
strat. 4	86	86	86	86	86
strat. 5	86	86	86	86	86
strat. 1-4	84	91	87	92	96
strat. 1-5	84	91	87	92	96
strat. 2-4	34	33	35	59	85
strat. 2-5	34	33	35	59	85
strat. 3-4	69	83	86	84	**97**
strat. 3-5	69	83	86	84	**97**

average result for some strategy, node devision and ϕ is in the first interval with borders the worst average result and worst average plus $d/10$ it is appreciated with 1. If it is in the second interval with borders the worst average plus $d/10$ and worst average plus $2d/10$ it is appreciated with 2 and so on. If it is in the 10th interval with borders the best average minus $d/10$ and the best average result, it is appreciated with 10. Thus for a test problem the achieved results for every strategy, every nodes devision and every ϕ is appreciated from 1 to 10. After that is summed the rate of all test problems for every strategy, every nodes devision and ϕ. So the rate of the strategies/node-devision/ϕ becomes between 10 and 100, because the benchmark problems are 10. It is mode of result classification.

On Table I is shown the rate of the strategies/node-devision when parameter $\phi = 0$, which means that only the achieved results from the last iteration are taken in to account in the node-subsets estimation, in bold is the best rate. We observe that the rate of the ACO algorithm with start strategies outperforms the traditional ACO with completely random start. Comparing the strategies, the worst rate has strategy 3 and their combinations with strategies 4 and 5. In strategy 3 the nodes-subsets with high value of estimation $E_j(i)$ become forbidden for current simulation. So if at the beginning iterations of the algorithm from some node-subset start only bad solutions it will be forbidden, but it is possible from this node subset to start good solutions too. The best rate have the combinations of strategies 1 and 3 with strategies 4 and 5. It means that it is better the node subsets which are appreciated like bad to be forbidden for a fixed number of iterations and it is better to forbid some node-subsets and to stimulate ants to start from other which looks to be good, than to apply only one strategy (forbidden or stimulated). The worst rate with respect of node devision is when there are 10 nodes in the node-subsets. When there are to many nodes in the node subset then it is possible from this subset to start good and bad solutions and it is difficult to appreciate it. The best rate with respect to the node devision is when in the node subsets is only one node.

TABLE III
ESTIMATION OF STRATEGIES AND NODES DEVISIONS FOR $\phi = 0.5$

number nodes	10	5	4	2	1
random	32	32	32	32	32
strat. 1	78	86	88	92	96
strat. 2	34	35	38	51	78
strat. 3	61	86	88	94	**97**
strat. 4	86	86	86	86	86
strat. 5	86	86	86	86	86
strat. 1-4	79	90	87	94	**97**
strat. 1-5	79	90	87	94	**97**
strat. 2-4	35	40	44	56	83
strat. 2-5	35	40	44	56	83
strat. 3-4	68	92	88	92	96
strat. 3-5	68	92	88	92	96

On Tables II, III and IV are shown the rate of the strategies/node-devision when parameter $\phi = 0.25, 0.5$ and 0.75. We can make similar to the $\phi = 0$ conclusions. For all values of the parameter ϕ the best rate according node devision is when there is only one node in node-subsets. So we put in Table V the rate of the start strategies when the node subsets consist one node, with bold is the best rate.

On Table V we observe that the worst rate according value of the parameter ϕ is when we take in to account only the achieved solutions from the last iteration ($\phi = 0$). The rate

TABLE IV
ESTIMATION OF STRATEGIES AND NODES DEVISIONS FOR $\phi = 0.75$

number nodes	10	5	4	2	1
random	32	32	32	32	32
strat. 1	71	81	85	89	92
strat. 2	35	55	52	60	87
strat. 3	56	76	88	95	95
strat. 4	86	86	86	86	86
strat. 5	86	86	86	86	86
strat. 1-4	67	83	89	94	95
strat. 1-5	67	83	89	94	95
strat. 2-4	39	47	48	58	85
strat. 2-5	39	47	48	58	85
strat. 3-4	56	81	87	94	**97**
strat. 3-5	56	81	87	94	**97**

TABLE V
ESTIMATION OF STRATEGIES AND PARAMETER ϕ

ϕ	0	0.25	0.5	0.75
random	32	32	32	32
strat. 1	83	93	96	92
strat. 2	74	81	78	87
strat. 3	86	96	**97**	95
strat. 4	86	86	86	86
strat. 5	86	86	86	86
strat. 1-4	89	96	**97**	95
strat. 1-5	89	96	**97**	95
strat. 2-4	82	85	83	85
strat. 2-5	82	85	83	85
strat. 3-4	90	**97**	96	**97**
strat. 3-5	90	**97**	96	**97**

of strategies when $\phi = 0.5$ is slightly better than rate when $\phi = 0.25$ and $\phi = 0.75$. So we can conclude that the balance between information from previous iterations and from last iteration is very important. According to the strategies, the worst rate is when is applied traditional ACO with random start and with strategy 2 when the subsets stay forbidden for current simulation. The best rate are combinations of strategy 3 with strategies 4 and 5. So for better performance of the ACO algorithm is advisable to forbid bad regions for several iterations and to stimulate ants to start construction of the solutions from good regions.

V. CONCLUSION

In this paper we address the modeling of the process of ant colony optimization method by using estimations, combining five start strategies. So, the start node of each ant depends of the goodness of the respective region. We focus on parameter settings which manage the starting procedure. We investigate on influence of the parameter ϕ to algorithm performance. The best solutions are achieved when "bad" regions are forbidden for several iterations and the probability the ants to start from "good" regions is higher. In future we will apply our modification of ACO algorithm on various classes of problems. We will investigate the influence of the estimations and start strategies on the achieved results.

ACKNOWLEDGMENT

This work has been partially supported by the Bulgarian National Scientific Fund under the grants Modeling Processes with fixed development rules - DID 02/29 and Effective Monte Carlo Methods for large-scale scientific problems - DTK 02/44.

REFERENCES

[1] E. Bonabeau, M. Dorigo and G. Theraulaz, *Swarm Intelligence: From Natural to Artificial Systems*, New York,Oxford University Press, 1999.

[2] W. Diffe and M.E. Hellman, *New direction in cryptography*, IEEE Trans Inf. Theory. IT-36,1976, 644-654.

[3] M. Dorigo and L.M. Gambardella, *Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem*, IEEE Transactions on Evolutionary Computation 1, 1997, 53-66.

[4] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MIT Press, 2004.

[5] S. Fidanova, *Evolutionary Algorithm for Multiple Knapsack Problem*, Int. Conference Parallel Problems Solving from Nature, Real World Optimization Using Evolutionary Computing, ISBN No 0-9543481-0-9,Granada, Spain, 2002.

[6] S. Fidanova, *Ant colony optimization and multiple knapsack problem*, in: Renard, J.Ph. (Eds.), Handbook of Research on Nature Inspired Computing for Economics ad Management, Idea Group Inc., ISBN 1-59140-984-5, 2006, 498-509.

[7] S. Martello and P. Toth, *A mixtures of dynamic programming and branch-and-bound for the subset-sum problem*, Management Science 30, 1984, 756-771.

[8] M. Reiman and M. Laumanns, *A Hybrid ACO algorithm for the Capacitate Minimum Spanning Tree Problem*, In proc. of First Int. Workshop on Hybrid Metahuristics, Valencia, Spain, 2004, 1-10.

[9] A. Sinha and A.A. Zoltner, *The multiple-choice knapsack problem*, J. Operational Research 27, 1979, 503-515.

[10] T. Stutzle and M. Dorigo, *ACO Algorithm for the Traveling Salesman Problem*, In K. Miettinen, M. Makela, P. Neittaanmaki, J. Periaux eds., Evolutionary Algorithms in Engineering and Computer Science, Wiley, 1999, 163-183.

[11] T. Stutzle and H.H. Hoos, *MAX-MIN Ant System*, In Dorigo M., Stutzle T., Di Caro G. (eds). Future Generation Computer Systems, Vol 16, 2000, 889–914.

[12] T. Zhang, S. Wang, W. Tian and Y. Zhang, ACO-VRPTWRV: A New Algorithm for the Vehicle Routing Problems with Time Windows and Re-used Vehicles based on Ant Colony Optimization, Sixth International Conference on Intelligent Systems Design and Applications, IEEE press, 2006, 390-395.