Ant Colony Optimization Start Strategies: Two Case Studies

Stefka Fidanova and Pencho Marinov

1 Introduction

The ability of ant colonies to form paths for carrying food is rather fascinating. The problem is solved collectively by the whole colony. This ability is explained by the fact that ants communicate in an indirect way by laying trails of pheromone. The higher the pheromone trail within a particular direction, the higher the probability of choosing this direction. The collective problem solving mechanism has given rise to a metaheuristic referred to es Ant Colony Optimization (ACO).
The ACO algorithm uses a colony of artificial ants that behave as cooperative agents in a mathematical space where they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. The problem is represented by graph and the ants walk on the graph to construct solutions. The solutions are represented by paths in the graph. After the initialization of the pheromone trails, the ants construct feasible solutions, starting from random nodes, and then the pheromone trails are updated. At each step the ants compute a set of feasible moves and select the best one (according to some probabilistic rules) to continue the rest of the tour. The structure of the ACO algorithm is shown by the pseudo-code below. 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 $i, j = 1, \ldots, n$.

$$p_{i,j} = \frac{\tau_{i,j}^a \eta_{i,j}^b}{\sum_{k \in Unused} \tau_{i,k}^a \eta_{i,k}^b},$$

where $Unused$ is the set of unused nodes of the graph. 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:

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

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

Figure 1: Pseudocode for ACO

where ρ models evaporation in the nature and $\Delta\tau_{i,j}$ is the new added pheromone which is proportional to the quality of the solution.
As other metaheuristics, ACO algorithm is applied on hard combinatorial optimization problems coming from real life and industry. It is unpractical to apply exact methods or traditional numerical methods on this kind of problems, because they need huge amount of computational resources, time and memory. Examples of optimization problems are Traveling Salesman Problem [6], Vehicle Routing [7], Minimum Spanning Tree [5], Multiple Knapsack Problem [4], etc.
In this paper we propose two kind of estimation of start nodes with respect to the quality of the solutions and thus to better menage the search process. We offer various start strategies and their combinations.

2 Subset Estimations

The essential part of ACO algorithm is starting from random node when ants create solutions. It is a kind of diversification of the search and leads to using small number of ants, which means less computational resources. 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 ($1 \leq j \leq N$). We propose two ways to estimate the nodes subsets:

Estimation 1

$$D_j(i) = \frac{i.D_j(i-1) + F_j(i)}{i},$$

$$E_j(i) = \frac{i.E_j(i-1) + G_j(i)}{i},$$

Estimation 2

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

$$E_j(i) = \phi.E_j(i-1) + (1-\phi).G_j(i), \tag{2}$$

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

$$F_j(i) = \begin{cases} \frac{f_{j,A}}{n_j} & \text{if } n_j \neq 0 \\ F_j(i-1) & \text{otherwise} \end{cases}, \tag{3}$$

$$G_j(i) = \begin{cases} \frac{g_{j,B}}{n_j} & \text{if } n_j \neq 0 \\ G_j(i-1) & \text{otherwise} \end{cases}, \tag{4}$$

$f_{j,A}$ 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

$$\sum_{j=1}^{N} n_j = n, \tag{5}$$

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 [3]. 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 $\{j\ |j$ 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 $\{j\ |j$ 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 $\{j\ |j$ 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 $\{j\ |D_j(i) > D\}$, 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 $\{j\ |D_j(i) > D\}$, 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 $St2 = \{strategy4,\ strategy5\}$. 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 ($strategy1$), ($strategy2$; $strategy5$), ($strategy3$; $strategy4$). 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 $St2$

3 Computational Results

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. 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/**

Table 1: Estimation 1 and Estimation 2

number nodes	10	5	4	2	1	number nodes	10	5	4	2	1
random	32	32	32	32	32	random	32	32	32	32	32
strat. 1	83	90	86	86	94	strat. 1	64	86	83	90	93
strat. 2	35	31	38	61	83	strat. 2	63	87	83	91	91
strat. 3	65	86	86	86	94	strat. 3	62	89	83	90	88
strat. 4	84	84	84	84	84	strat. 4	84	82	84	87	92
strat. 5	84	84	84	84	84	strat. 5	85	85	86	90	87
strat. 1-4	85	92	87	87	**96**	strat. 1-4	68	89	85	93	**95**
strat. 1-5	85	92	87	87	**96**	strat. 1-5	62	90	81	92	92
strat. 2-4	35	37	37	62	84	strat. 2-4	65	89	86	90	93
strat. 2-5	35	37	37	62	84	strat. 2-5	59	88	84	92	93
strat. 3-4	71	85	86	93	94	strat. 3-4	66	90	84	94	93
strat. 3-5	71	85	86	93	94	strat. 3-5	61	86	81	94	92

mastjjb/jeb/orlib/, with 100 objects and 10 constraints. 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 [4] 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. 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 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.

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 tables the rate of the start strategies when the node subsets consist one node, with bold is the best rate. We compare the achieved results by the estimation method 2 with results by the estimation method 1 when $\phi = 0.5$ because it is the best value for ϕ.

Comparing the tables me observe that rates of the estimation method 1 are higher than rates of estimation method 2 in most of the cases with exeption of strategy 2 and its combinations with strategies 4 and 5. The best rate with estimation 1 is

96 while the best rate estimation 2 is 95. The both are achieved wen strategy 1 is combined with strategy 4. We can conclude that the estimation method 1 achieves better results than estimation method 2.

4 Conclusion

In this paper we address the modeling of the process of ant colony optimization method by using two kind of nodes estimations, combining 5 start strategies. So, the start node of each ant depends of the goodness of the respective region. In a future we will focus on parameter settings which manage the starting procedure. We will investigate on influence of the parameters to algorithm performance.

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

References

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

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

[3] Dorigo M., Stutzle T., Ant Colony Optimization, MIT Press, 2004.

[4] Fidanova S., 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.

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

[6] Stutzle T. Dorigo M., 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, 163-183, 1999.

[7] Zhang T., Wang S., Tian W., Zhang Y., 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.