Control and Cybernetics

vol. **43** (2014) No. 3

New evaluations of ant colony optimization start nodes*

by

Stefka Fidanova[1], Pencho Marinov[1] and Krassimir Atanassov[2]

[1]Institute of Information and Communication Technologies,
Bulgarian Academy of Sciences
Acad. G. Bonchev St. bl. 25A, 1113 Sofia, Bulgaria
stefka@parallel.bas.bg, pencho@parallel.bas.bg
[2]Institute of Biophysics and BioMedical Engineering,
Bulgarian Academy of Sciences
Acad. G. Bonchev St. bl. 105, 1113 Sofia, Bulgaria
krat@bas.bg

Abstract: Ant Colony Optimization (ACO) is a stochastic search method that mimics the social behavior of real ant colonies, managing to establish the shortest route to the 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. In this paper, the semi-random start procedure is applied. A new kind of evaluation of start nodes of the ants is developed and several starting strategies are prepared and combined. The idea of semi-random start is related to a better management of the ants. This new technique is tested on the Multiple Knapsack Problem (MKP). A Comparison among the strategies applied is presented in terms of quality of the results. A comparison is also carried out between the new evaluation and the existing one. Based on this comparative analysis, the performance of the algorithm is discussed. The study presents the idea that should be beneficial to both practitioners and researchers involved in solving optimization problems.

Keywords: combinatorial optimization, ant algorithms, start nodes evaluation, semi random start

1. 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 (Stutzle and Dorigo, 1999), Vehicle Routing (Zhang et al., 2006), Minimum Spanning Tree (Reiman and Laumanns, 2004), Multiple Knapsack

*Submitted: January 2011; Accepted: September 2014

Problem (Fidanova, 2002), etc. They are NP-hard problems and it is unpractical to apply exact or traditional numerical methods to them, because they need huge amounts of computational resources. In order to obtain solutions close to the optimal ones in reasonable time, metaheuristic methods are used. One of them is Ant Colony Optimization (ACO) (Dorigo and Gambardella, 1997).

ACO algorithms have been inspired by the real ant behavior. In nature, ants usually wander randomly, and upon finding food, they 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, relative to the others, 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 behavior of the ants, walking around the graph representing the problem to solve, using mechanisms of cooperation and adaptation. ACO algorithm requires having following definitions (Bonabeau, Dorigo and Theraulaz, 1999; Dorigo and Stutzle 2004):

- The appropriate representation of the problem, so as to allow the ants to incrementally update the solutions through the use of 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, which measures the quality of components that can be added to the current partial solution;
- A set of rules 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 problem is represented by a graph and the solutions are represented by paths in a graph. The method is constructive and does not need the initial solutions. In every iteration ants begin to create their solutions starting from random nodes of the graph. Random start is a kind of diversification of the search. Then, the subsequent nodes are included in the solution by applying the probabilistic rule called transition probability. At the end of every iteration the quantity of the pheromone is updated. The main rule is that the elements of better solutions should receive more pheromone than others and thus become more desirable in the next iteration. It is a kind of intensification of the search around good solutions. The pheromone can be deposited on the nodes of the graph or on the arcs of the graph. The choice between the two is problem dependent.

The structure of the ACO algorithm is shown by the pseudo-code given in Fig. 1. The transition probability $p_{i,j}$, i.e. the one of choosing 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 $i, j = 1, \ldots, n$:

$$p_{i,j} = \frac{\tau_{i,j}^a \eta_{i,j}^b}{\sum\limits_{k \in allowed} \tau_{i,k}^a \eta_{i,k}^b}. \tag{1}$$

The higher the value of the pheromone and the heuristic information, the more profitable it is to select the given 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 adapt different criteria to update the pheromone level.

```
Ant Colony Optimization
Initialize the 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

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 the rate of evaporation of the pheromone, while $\Delta\tau_{i,j}$ is the newly added pheromone, which is proportional to the quality of the solution. There exist several variants of ACO algorithms, which differ with respect to pheromone updating.

In the ACO algorithms, in every iteration the ants start to create their solutions from a random node of the graph of the problem. It is a kind of diversification of the search. Our research is focused on semi random start and its influence on algorithm performance, regardless of the variant of ant algorithm that is used. With the semi random start we keep the diversification and, on the other hand, we try to avoid bad starting nodes.

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

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

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

2. The starting strategies

The known ACO algorithms create a solution starting from a 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 a node, which does not belong to the optimal solution, the probability of constructing the optimal solution is zero. On the other hand, the random start is a kind of diversification of the search process, implying that we might need less ants, which means less computational resources. Therefore, we offer several starting strategies, by which we keep the random start in some sense, and at the same time we force the ants to partially avoid bad solutions.

Let the graph of the problem have m nodes. We divide the set of nodes into N subsets. There are different ways for performing this division. Normally, the nodes of the graph are randomly enumerated. An example for creating the node 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 numbers of nodes in the subsets are almost equal. We introduce evaluation $D_j(i)$ 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 the j^{th} node subset $(1 \leq j \leq N)$, which we calculate by the following formulas:

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

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

where $i \geq 2$ is the current process iteration and for each j $(1 \leq j \leq N)$:

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

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

$f_{j,A}$ 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 the node subset j, n being the total number of ants. Initial values of the weight coefficients are: $D_j(1) = 1$ and $E_j(1) = 0$ $(1 \leq j \leq N)$. 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 into account. If $\phi = 0.5$, the influence of the previous iteration is equal to that of the last one. 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 the one from the last iteration. When $\phi = 0.75$, the weight of the information from previous iterations is three times bigger than the one from the last iteration. The balance between the weights of the information from previous iterations and from the last one is important. At the beginning, when the current best solution is far from the optimal one, some of the node subsets can be evaluated as good. Therefore, if the value of the parameter ϕ is too high, the evaluation can be distorted. If the weight of the last iteration is too high, then information concerning good and bad solutions from previous iterations is ignored, which can distort evaluation, too.

We try to use the experience of the ants from previous iterations to choose the better starting node. Other authors use this experience only by the pheromone, when the ants construct the solutions (Dorigo and Stutzle, 2004). Let us fix the threshold E for $E_j(i)$ and D for $D_j(i)$, then we construct several strategies for choosing the start node for every ant. The threshold E increases at every iteration with $1/i$, where i is the number of the current iteration. We propose the following starting strategies:

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

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

K_1 is a parameter ($0 \leq K_1 \leq$"number of iterations"). If $K_1 = 0$, then strategy 3 is equal to the random choice of the start node. If $K_1 = 1$, then strategy 3 is equal to strategy 1. If $K_1 =$"maximal number of iterations", then strategy 3 is equal to strategy 2.

We can use more than one strategy for choosing the start node, but there are strategies, which can not be combined. We divide the strategies into two sets: $St1 = \{strategy1,\ strategy2,\ strategy3\}$ and $St2 = \{strategy4,\ strategy5\}$. The strategies from the set $St1$ are prohibitive strategies and the strategies from the set $St2$ are incentive strategies. The strategies from the same set can not be used simultaneously. Thus, we can use a strategy from one set or combine it with a strategy from the other set. Exemplary combinations are ($strategy1$), ($strategy2$; $strategy4$), ($strategy3$; $strategy5$). When we combine strategies from $St1$ and $St2$, we first apply the strategy from $St1$ and, according to it, some of the regions (node subsets) become forbidden, and after that we choose the starting node from the not forbidden subsets, according the strategy from $St2$.

3. The Multiple Knapsack Problem

We test the idea for the 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 (1979) proposed to use the MKP in fault tolerance problem, and in Diffie and Hellman (1976) a public cryptography scheme is designed, whose security is funded on the difficulty of solving the MKP. Martello and Toth (1984) mention that the two-processor scheduling problem may be solved as an MKP. Other applications are industrial management, naval and aerospace problems, and computational complexity theory.

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

The MKP can be formulated as follows:

$$\begin{gathered} \max \textstyle\sum_{j=1}^{n} p_j x_j \\ \text{subject to} \sum_{j=1}^{n} r_{ij} x_j \leq c_i \quad i = 1, \ldots, m \\ x_j \in \{0, 1\} \quad j = 1, \ldots, n \end{gathered} \tag{8}$$

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

This problem has m constraints, and so MKP is also called the 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 $r_{ij} \leq c_i \leq \sum_{k=1}^{n} r_{ik}$ for all $i \in I$ and $j \in J$. Note that the $[r_{ij}]_{m \times n}$ matrix and the $[c_i]_m$ vector are both non-negative.

In the MKP one is not interested in solutions giving a particular order. Therefore, the partial solution is represented by $S = \{i_1, i_2, \ldots, i_j\}$ and the most recent element incorporated in 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 searches 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. The fully connected graph means that after object i one can choose any object j, if there are enough resources and the object j has not been chosen yet.

Leguizamon and Michalewicz (1999) solve the MKP by applying the ACO algorithm with dynamic heuristic information. Fidanova (2008) compared several kinds of heuristic information, including dynamic, and found that the ACO algorithm with some of the static heuristic information performs better, than with the dynamic one. Min Kong (1999) proposed various pheromone intensification strategies and compared the algorithm performance. All of these algorithms use random start of the ants in every iteration. Our research is focused on semi random start and its influence on algorithm performance, regardless of the variant of the ant algorithm that is used.

4. Computational results

The computational experience with the ACO algorithm is shown using 10 MKP instances from the "OR-Library" available at **http://people.brunel. ac.uk/ mastjjb/jeb/orlib/**, with 100 objects and 10 constraints. To provide a fair comparison for the here constrained 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 obtained results will be very close to the optimal solution and it will be difficult to appreciate the difference between the strategies. We apply

strategies proposed on MMAS (Stutzle and Hoos, 2000), because it is one of the best among the ACO approaches. The developed technique has been coded in C++ language and implemented on a Pentium 4 (2.8 GHz). The ACO parameters are fixed as follows: $\rho = 0.5$, $a = 1$, $b = 1$, the 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 Fidanova (2006) and it was found experimentally that they are best for MKP. The tests are run with 1, 2, 4, 5 and 10 nodes in the node 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, taken by the start strategies, is negligible with respect to the computational time, which is taken by solution construction.

Tests with all possible combinations of strategies and with random start (12 combinations), four values for ϕ and five kinds of node subsets, are run, and every test is run 30 times. Thus, the total number of runs is 72 000. One can observe that sometimes all the node subsets become forbidden and the algorithm stops before performing all iterations (strategies 1, 2, 3 and combinations involving them). So, if all node 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 continues by applying the chosen strategy.

The problem, which arises, is how to compare the solutions obtained with the use of different strategies and different node-divisions. For this purpose, the difference (interval) d between the worst and best average result for every problem is divided into 10 equal segments. If the average result for some strategy, node division and ϕ is in the first interval, with borders being the worst average result and the worst average plus $d/10$, it gets the score equal to 1. If it is in the second interval, with borders being the worst average plus $d/10$ and worst average plus $2d/10$, it gets the score equal to 2 and so on. If it is in the 10th interval, with borders being the best average minus $d/10$ and the best average result, it gets the score equal to 10. Thus, for a test problem, the obtained results for every strategy, every node division and every ϕ get the score ranging from 1 to 10. After that, the scores of all the test problems for every strategy, every node division and ϕ is summed. So, the score for the strategy/node-division/ϕ takes the value between 10 and 100, because there are 10 benchmark problems.

Table 1 shows the scores for the strategies/node-divisions when parameter $\phi = 0$, which means that only the results from the last iteration are taken into account in the node-subsets evaluation. The best score is being shown in bold. We observe that the scores of the ACO algorithm with starting strategies imply that it outperforms the traditional ACO with completely random start. Comparing the strategies, we see that the worst scores characterize strategy 2 and its combinations with strategies 4 and 5. In strategy 2, the node-subsets with high value of evaluation $E_j(i)$ become forbidden for the current simulation. At the initial iterations of the algorithm only bad solutions start from some node-subset, and the subset gets forbidden, even though it will still be possible

Table 1. Evaluation of strategies and node divisions for $\phi = 0$

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

to start good solutions from this node-subset. The best scores characterize the combinations of strategies 1 and 3 with strategies 4 and 5. This means that it is better for the node subsets, which are evaluated as bad, to be forbidden for a fixed number of iterations in order to stimulate ants to start from the other ones, which seem to be good. The worst scores, with respect of node division, are obtained when there are 10 nodes in the node-subsets. When there are too many nodes in the node subset, then it is possible to start good and bad solutions from this subset and it is difficult to appreciate this subset. The best score, with respect to the node division, are obtained when there is only one node in the node subsets.

In Tables 2, 3 and 4 the scores of the strategies/node-divisions for the values of the parameter $\phi = 0.25, 0.5$ and 0.75 are displayed. We can forward a conclusion similar to those formulated for $\phi = 0$. For all values of the parameter ϕ the best scores regarding the node division are obtained when there is only one node in the node-subsets. So, we put in Table 5 the scores of the starting strategies when the node subsets contain one node, the best score being again shown in bold.

In Table 5 we observe that the worst scores, regarding the value of the parameter ϕ, are obtained when we take into account only the solutions derived from the last iteration ($\phi = 0$). The scores of the strategies for $\phi = 0.75$ are slightly better than the scores for $\phi = 0.25$ and $\phi = 0.5$ (with, however, the scores for $\phi = 0,25$ being, interestingly, almost equivalent to those for $\phi = 0,75$). So, we can conclude that the balance between information from previous iterations and from the last iteration is very important. When comparing the

Table 2. Evaluation of strategies and node divisions for $\phi = 0.25$

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

Table 3. Evaluation of strategies and node divisions for $\phi = 0.5$

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

Table 4. Evaluation of strategies and node divisions for $\phi = 0.75$

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

Table 5. Evaluation of strategies and parameter ϕ values

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

strategies, we see that the worst scores are obtained when traditional ACO is applied with random start and with strategy 2, when the subsets stay forbidden for the current simulation. The best scores are achieved by combining strategy 3 with strategies 4 and 5 (very closely followed by the combinations of strategy 1 with 4 and 5). For a better performance of the ACO algorithm it is advisable to forbid "bad" regions for several iterations and to stimulate ants to start the construction of the solutions from "good" regions.

We compare the results obtained in this work with the results obtained in our previous work (Fidanova et al., 2009). In Fidanova et al. (2009) we used the same starting strategies as in this work, but with other evaluation functions, namely:

$$D_j(i) = \frac{i.D_j(i-1) + F_j(i)}{i}, \tag{9}$$

$$E_j(i) = \frac{i.E_j(i-1) + G_j(i)}{i}, \tag{10}$$

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

$$F_j(i) = \begin{cases} \frac{f_{j,A}}{n_j} & \text{if } n_j \neq 0 \\ F_j(i-1) & \text{otherwise} \end{cases}, \tag{11}$$

$$G_j(i) = \begin{cases} \frac{g_{j,B}}{n_j} & \text{if } n_j \neq 0 \\ G_j(i-1) & \text{otherwise} \end{cases}, \tag{12}$$

and $f_{j,A}$ is the number of solutions among the best $A\%$, while $g_{j,B}$ is the number of solutions among the worst $B\%$, where $A + B \leq 100$, $i \geq 1$ and

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

where n_j $(1 \leq j \leq N)$ is the number of solutions obtained by ants starting from the node subset j. Initial values of the weight coefficients are $D_j(1) = 1$ and $E_j(1) = 0$.

We compared the results from Fidanova et al.(2009) with the current results for $\phi = 0.75$, because it is the best value for ϕ. We calculated the scores of the strategies using the best and the worst solutions obtained with the use of both evaluation methods.

When comparing Tables 6 and 7 we observe that the scores of the evaluation method from the current work are higher than those from Fidanova et al. (2009) in most of the cases, except for strategy 2 and its combinations with strategies 4 and 5. The best score in Table 6 is 97, while the best score in Table 7 is 95. We can conclude that the evaluation method proposed in the current work leads to better results than the ones obtained in Fidanova et al. (2009).

Table 6. Evaluation of strategies and node divisions

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

Table 7. Evaluation of strategies and node divisions with the evaluation strategies from Fidanova (2009)

number of nodes	10	5	4	2	1
random	32	32	32	32	32
strategy 1	64	86	83	90	93
strategy 2	63	87	83	91	91
strategy 3	62	89	83	90	88
strategy 4	84	82	84	87	92
strategy 5	85	85	86	90	87
strategy 1-4	68	89	85	93	**95**
strategy 1-5	62	90	81	92	92
strategy 2-4	65	89	86	90	93
strategy 2-5	59	88	84	92	93
strategy 3-4	66	90	84	94	93
strategy 3-5	61	86	81	94	92

5. Conclusion

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

Acknowledgment

This work has been partially supported by the Bulgarian National Scientific Fund under the grants Ultimate/Supreme Parallel Algorithms for Large-Scale Computational Problems and Modeling of Data Mining Techniques Using Generalized Nets.

References

BONABEAU E., DORIGO M. and THERAULAZ G. (1999) *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York.

DIFFIE W. and HELLMAN M.E. (1976) New directions in cryptography. *IEEE Trans Inf. Theory*, **22** (6), 644–654.

DORIGO M. and GAMBARDELLA L.M. (1997) Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. *IEEE Transactions on Evolutionary Computation* 1, 53-66.

DORIGO M. and STUTZLE T. (2004) *Ant Colony Optimization*. MIT Press.

FIDANOVA S. (2002) Evolutionary Algorithm for Multiple Knapsack Problem. *Int. Conference Parallel Problems Solving from Nature, Real World Optimization Using Evolutionary Computing*, Granada, Spain, University of Granada.

FIDANOVA S. (2006) Ant colony optimization and multiple knapsack problem. In: J. Ph. Renard, ed., *Handbook of Research on Nature Inspired Computing for Economics ad Management*. Idea Group Inc., 498-509.

FIDANOVA S. (2008) Probabilistic Model of Ant Colony Optimization for Multiple Knapsack Problem. *Large Scale Scientific Computing. Lecture Notes in Computer Science*, **4818**, 545-552.

FIDANOVA S., ATANASSOV K., MARINOV P. and PARVATHI R. (2009) Ant Colony Optimization for Multiple Knapsack Problem with Controlled Start. *Int. Journal on BIOautomation*, **13**(4), 271-280.

KONG M., TIAN P. and KAO Y. (2008) A New Ant Colony Optimization Algorithm for the Multidimensional Knapsack Problem. *J. Computers and Operational Research*, **35**(8), 2672–2683.

LEGUIZAMON G. and MICHALEWICZ Z. (1999) A New Version of Ant System for Subset Problems. In: *Proceedings of Congress on Evolutionary Computation*. IEEE Press, 1459–1464.

MARTELLO S. and TOTH P. (1984) A mixture of dynamic programming and branch-and-bound for the subset-sum problem. *Management Science* 30, 756–771.

REIMAN M. and LAUMANNS M. (2004) A Hybrid ACO algorithm for the Capacitate Minimum Spanning Tree Problem. In: Ch. Blum, A. Roli and M. Sampels, eds., *Proceedings of First International Workshop on Hybrid Metaheuristics*, Valencia, Spain, ISBN 3-00-015331-4, 1–10.

SINHA A. and ZOLTNER A.A. (1979) The multiple-choice knapsack problem. *J. Operational Research* 27, 503–515.

STUTZLE T. and DORIGO M. (1999) ACO Algorithm for the Traveling Salesman Problem. In: K. Miettinen, M. Makela, P. Neittaanmäki, J. Periaux, eds., *Evolutionary Algorithms in Engineering and Computer Science*. Wiley, 163–183.

STUTZLE T. and HOOS H.H. (2000) MAX-MIN Ant System. In: M. Dorigo, T. Stutzle and G. Di Caro, eds., *Future Generation Computer Systems*, 16. Elsevier, 889–914.

ZHANG T., WANG S., TIAN W. and ZHANG Y. (2006) ACO-VRPTWRV: A New Algorithm for the Vehicle Routing Problems with Time Windows and Re-used Vehicles based on Ant Colony Optimization. In: *Proc. of Sixth International Conference on Intelligent Systems Design and Applications*, IEEE Press, 390-395.