Chapter XXXIII
Ant Colony Optimization and Multiple Knapsack Problem

Stefka Fidanova
Bulgarian Academy of Science, Bulgaria

ABSTRACT

The ant colony optimization algorithms and their applications on the multiple knapsack problem (MKP) are introduced. The MKP is a hard combinatorial optimization problem with wide application. Problems from different industrial fields can be interpreted as a knapsack problem including financial and other management. The MKP is represented by a graph, and solutions are represented by paths through the graph. Two pheromone models are compared: pheromone on nodes and pheromone on arcs of the graph. The MKP is a constraint problem which provides possibilities to use varied heuristic information. The purpose of the chapter is to compare a variety of heuristic and pheromone models and different variants of ACO algorithms on MKP.

INTRODUCTION

Combinatorial optimization is a process of finding the best or optimal solution for problems with a discrete set of feasible solutions. Applications occur in numerous settings involving operations management and logistics. The economic impact of combinatorial optimization is profound, affecting diverse sections. While much progress has been made in finding exact solutions to some combinatorial optimization problems (COPs), many hard combinatorial problems (NP-problems) are still not exactly solved in a reasonable time and require good meta-heuristic methods. The aim of meta-heuristic methods for COPs is to produce quickly good-quality solutions. In many practical problems they have proved to be effective and efficient approaches, being flexible to accommodate variations in problem structure and in the objectives considered for the evaluation of solutions (Lonnstedt, 1973). For all these reasons, meta-heuristics has probably been one of the most stimulated research topics in optimization for the last two decades. Examples are decision making problems.

The ant colony optimization (ACO) is a new meta-heuristic method. ACO algorithms are applied in real life and industrial problems for which a good solution for a short time is required. ACO achieves good results for problems with restrictive constraints like multiple knapsack problem. It represents a multi-agent system where low-level interaction between single agents results in a complex behavior of the whole ant colony. It imitates the behavior shown by real ants when searching for food. Ants are social insects that live in colonies and whose behavior is aimed more to the survival of the colony as a whole than to that of a single individual component of the colony. An important and interesting aspect of ant colonies is how ants can find the shortest path between the food sources and their nest. Ants communicate information about food sources via a chemical substance called pheromone, which the ants secrete as they move along.

Analogously, ACO is based on the indirect communication of a colony of simple agents, called "artificial" ants, mediated by "artificial" pheromone trails. The pheromone trails in ACO algorithms serve as distributed numerical information, which ants use to probabilistically construct solutions to the problem to be solved and which ants adapt during the algorithm's execution to reflect their search experience. Artificial ants not only imitate the behavior described, but also apply additional problem-specific heuristic information. The idea is developed by Moyson and Manderick (1988). The first example of ant algorithm is Ant System (Dorigo, Maniezzo, & Colorni, 1996), and it has been applied to and provided solutions for various hard combinatorial optimization problems. Recently, different versions of the ACO algorithms such as the ant colony system (Dorigo, 1999, pp. 53-66), the ant algorithm with elitist ants (Dorigo, 1999, pp. 137-172), the max-min ant system (Stützle & Hoos, 2000), the ant algorithm with additional reinforcement (Fidanova, 2002), and the best-worst ant system (Cordón, Fernàndez de Viana, & Herrera, 2002) have been applied to many optimization problems. Examples are the traveling salesman problem (Dorigo, 1999, pp. 53-66), the quadratic assignment (Gambardella, 1999, pp.167-176), the vehicle routing (Gambardella, 1999, pp. 63-76), and the multiple knapsack problem (Fidanova, 2003).

The multiple knapsack problem (MKP) is a hard combinatorial optimization problem with wide applications which enlists many practical problems from different domains like financial and other management. It is an interesting problem of both practical and theoretical point of view: practical because of its wide application; theoretical because it is a constraint problem and gives various possibilities for heuristic constructions.

The aim of this chapter is to introduce ACO and its application on MKP.

ANT COLONY OPTIMIZATION ALGORTHM

All ACO algorithms can be applied to any COP. They follow specific algorithmic scheme. After the initialization of the pheromone trails and control parameters, a main loop is repeated until the stopping criteria are met. The stopping criteria can be a certain number of iterations, a given CPU time limit, or a time limit without improving the result or if some lower (upper) bound of the result is known and the achieved result is close enough to this bound. In the main loop, the ants construct feasible solutions, and then the pheromone trails are updated. More precisely, partial problem solutions are seen as states: each ant starts from random state and moves from a state i to another state *j* of the partial solution. At each step, ant *k* computes a set of feasible expansions to its current state and moves to one of these expansions, according to a probability distribution specified as follows. For ant *k*, the probability p_{ij}^k to move from a state *i* to a state *j* depends on the combination of two values:

$$p_{ij}^k = \begin{cases} \dfrac{\tau_{ij}.\eta_{ij}}{\sum\limits_{l \in allowed_k} \tau_{il}.\eta_{il}} & if \ \ j \in allowed_k \\ 0 & otherwise \end{cases} \qquad (1)$$

where:

- η_{ij} is the attractiveness of the move as computed by some heuristic information, indicating a priori desirability of that move;
- τ_{ij} is the pheromone trail level of the move, indicating how profitable it has been in the past to make that particular move (it represents therefore a posterior indication of the desirability of that move); and
- $allowed_k$ is the set of remaining feasible states.

Thus, the higher the value of the pheromone and the heuristic information are, the more profitable it is to include state *j* in the partial solution. In the beginning, the initial pheromone level is set to τ_0, which is a small positive constant. In nature there is not any pheromone on the ground at the beginning, or the initial pheromone is $\tau_0 = 0$. If in ACO algorithm the initial pheromone is $\tau_0 = 0$, then the probability to chose next state will be $p_{ij}^k = 0$ and the search process will stop from the beginning. Thus it is important that the initial pheromone value is positive.

The pheromone level of the elements of the solutions is changed by applying the following updating rule:

$$\tau_{ij} \leftarrow \rho.\tau_{ij} + \Delta\tau_{ij} \qquad (2)$$

where the rule $0 < \rho < 1$ models evaporation and $\Delta\tau_{ij}$ is an added pheromone. The ACO algorithms differ in pheromone updating. There exist various versions of ACO algorithms such as the ant system (Dorigo et al., 1996), the ant colony system (Dorigo, 1999, pp. 53-66), ACO with elitist ants (Dorigo, 1999, pp. 137-172), the max-min ant system (Stützle & Hoos, 2000), the ant algorithm with additional reinforcement (Fidanova, 2002, pp. 292-293), the best-worst ant system (Cordón et al., 2002), and so on. The main difference between them is pheromone updating.

Ant System

The first ant algorithm is ant system. In this algorithm all pheromone is decreased, and after that every ant adds a pheromone corresponding to the quality of the solution. More precisely:

$$\Delta\tau_{ij}^k = \begin{cases} (1-\rho) f(S^k) & \text{if it is maximization problem} \\ (1-\rho)/f(S^k) & \text{if it is minimization problem} \end{cases}$$

where:

- $\Delta\tau_{ij}^k$ is the pheromone added by the ant *k*;
- S^k is the solution achieved by ant *k*;
- *f*(S^k) is the value of the objective function.

In any ant system, better solutions and elements used by more ants receive more pheromone.

Ant Colony System (ACS)

The main features of the ACS algorithm are as follows: to use the best found solution, after each iteration the ants—which construct the best solution from the beginning of the trials—add pheromone; to avoid stagnation of the search, the pheromone on other solutions is decreased. Local pheromone updating and global pheromone updating in ACS are applied. In the local pheromone updating the value of the pheromone on used elements decreases and comes between initial pheromone τ_0 and the old value of the pheromone. It is a kind of diversification of the search in the search space.

$$\tau_{ij} \leftarrow \rho.\tau_{ij} + (1-\rho).\tau_0$$

In the global pheromone updating, the ant that constructs the best solution adds another pheromone depending on the quality of the solution.

$$\tau_{ij} \leftarrow \alpha.\tau_{ij} + (1-\alpha).\Delta\tau_{ij}$$

$$\Delta\tau_{ij}^k = \begin{cases} f(S^k) & \text{if it is maximization problem} \\ 1/f(S^k) & \text{if it is minimization problem} \end{cases}$$

The main idea of ACS is to enforce the pheromone of the best found solution and at the same time to diversify the search.

Ant Algorithm with Elitist Ants

In this ant algorithm only one or a fixed number (n) of ants add pheromone. The pheromone corresponding to other elements is only evaporated. Thus the pheromone of the best n solutions is forced. It is a kind of intensification of the search around the best found solutions.

Max-Min Ant System (MMAS)

The main features of MMAS algorithm are as follows:

- To exploit the best found solution—after each iteration only one ant adds a pheromone.
- To avoid stagnation of the search, the range of possible pheromone value is limited to a fixed interval $[\tau_{\min}, \tau_{\max}]$.

In MMAS algorithm the pheromone value is initialized so that after the first iteration all pheromone values are equal to $\tau_{\max}$. In the next iterations only the elements belonging to the best solution receive a pheromone; other pheromone values are only evaporated. The main idea of MMAS is to use fixed lower and upper bounds of the pheromone values. If some pheromone value is less/greater than lower/upper bound, it becomes equal to this fixed lower/upper bound and thus early stagnation of the algorithm is avoided.

Best-Worst Ant System (BWAS)

The main idea of BWAS is to use a pheromone mutation. The pheromone value of the best solution is increased, while the pheromone value of the worst solution is decreased. Thus the probability to choose elements of worst solution in the next iteration becomes lower.

Ant Algorithm with Additional Reinforcement (ACO-AR)

The main idea of ACO-AR is after pheromone updating to add additional pheromone to unused elements. Thus some elements receive additional probability to be chosen and become more desirable. Using ACO-AR algorithm the unused elements have the following features:

- They have a greater amount of pheromone than the elements belonging to poor solutions.
- They have a less amount of pheromone than the elements belonging to the best solution.

Thus the ants are forced to exploit a search space which has not been exploited yet without repeating the bad experience.

ANT ALGORITHM AND CONVERGENCE

The ACO is a meta-heuristic algorithm for approximate solution of combinatorial optimization problems. The construction of a good solution is a result of the agents' cooperative interaction. Failure in local optimum may occur when we perform the ACO algorithm. This can happen when the pheromone trail is significantly higher for one choice than for all others. This means that one of the choices has a much higher amount of pheromone than the others, and an ant will prefer this solution component over all alternatives. In this situation, ants construct the same solution over and over again, and the exploration of the search space stops. It should be avoided by influencing the probabilities for choosing the next solution component which depends directly on the pheromone trails. Various techniques exist to avoid failing into local optimum as re-initialization, smoothing of the pheromone, additional reinforcement, diversification, and intensification of the search.

Re-Initialization

When the ants repeat the same solution over and over again, the pheromone is re-initialized (Stützle & Hoos, 2000) and the algorithm starts from the beginning. The aim is to start to create solutions from other starting states and probably to construct differently from previous solutions. This technique can prevent some failing into local optimums, but the algorithm is not guaranteed to converge to an optimal solution. This technique can be applied to any ant algorithm.

Smoothing of the Pheromone Trails

The main idea of the smoothing (Stützle & Hoos, 2000) is to increase the pheromone trails according to their differences to the maximal pheromone trail as follows:

$$\tau_{ij} \leftarrow \tau_{ij} + \delta.(\tau_{\max} - \tau_{ij}),$$

where $0 < \delta < 1$ is a smoothing parameter. The above proposed mechanism has the advantage that the information gathered during the run of the algorithm is not completely lost, but merely weakened. For $\delta = 1$ this mechanism corresponds to a re-initialization of the pheromone trails, while for $\delta = 0$ pheromone trail smoothing is switched off. After the smoothing, the current lower bound of the pheromone increases.

Fixed Bounds of the Pheromone

Other method to prevent early stagnation is to fix the lower and the upper bound of the pheromone (Stützle & Hoos, 2000). Thus if the pheromone becomes less/greater than the lower/upper bound, it becomes equal to this lower/upper bound. Thus there are not choices of very high or very low amounts of pheromone.

Additional Reinforcement

The aim of additional reinforcement is to add additional pheromone on choices of pheromone low amount and thus they become more desirable (Fidanova, 2002, pp.292-293). It is a way to force the ants to look for new solutions.

Search Diversification and Intensification

In some ant algorithms, diversification and intensification techniques—like increasing the amount of the pheromone for some choices and decreasing it for others—are used. The aim is to intensify the solution search on one side and to diversify it on the other.

It is important to know whether the algorithm converges to the global optimum. Stützle and Dorigo (2002) proved that if the amount of the pheromone has a finite upper bound and a positive lower bound, then the ACO algorithm converges to the optimal solution. This means that if the probability to choose any element does not converge to zero, then the ACO algorithm converges to the global optimum. Stützle and Dorigo (2002) proved that the Ant Colony System and Max-Min Ant System satisfy the conditions for convergence and thus they converge to the global optimum when the time (number of iterations) converge to infinity.

Additional reinforcement can be applied to any ACO algorithm. Fidanova (2004) has proved that after additional reinforcement of unused elements of any ACO algorithm, it converges to optimal solution when the algorithm is run for a sufficiently large number of iterations independently whether the original ACO algorithm converges.

MULTIPLE KNAPSACK PROBLEM

The MKP has numerous applications in theory as well as in practice. It also arises as a subproblem in several algorithms for more complex COPs, and these algorithms will benefit from any improvement in the field of MKP.

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

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

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

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

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 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 $r_{ij} \leq c_i \leq \sum_{j=1}^{n} r_{ij}$ for all and. Note that the matrix and the vector are both non-negative.

We can mention the following major applications: problems in cargo loading, cutting stocks, bin-packing, budget control, and financial management may be formulated as MKP. Sinha and Zoltner (1979) propose the use of the MKP in fault tolerance problem, and Diffe and Hellman (1976) designed a public cryptography scheme whose security realizes the difficulty of solving the MKP. Matrello and Toth (1984) mention that two-processor scheduling problem may be solved as a MKP. Other applications are industrial management, team management, naval, aerospace, and computational complexity theory.

The shortest path problem in a transportation network deals with determining the subset of the connected roads that collectively comprise the shortest driving distance or the smallest driving time or the cheapest fair between two cities. The problem is: what subset of lines gives the faster response time for communication between them? Complexity theory is a part of the theory of computation dealing with the resources required during the computation time to solve a given problem. More theoretical application appears either in case a general problem is transformed to a MKP or MKP appears as a sub-problem in solving the generalized assignment problem. It again is used in solving a vehicle routing problem. In addition, MKP can be seen as a general model for any kind of binary problems with positive coefficients (Kochenberger, McCarl, & Wymann, 1974).

In solving MKP one is not interested in solutions giving a particular order. Therefore a partial solution is represented by , and the most recent elements incorporated to *S,* need not be involved in the process for selecting the next element. Also, solutions for ordering problems have a fixed length, as a permutation of a known number of elements is searched. Solutions of MKP, however, do not have a fixed length. In this chapter the solution will be represented by sequence where is 1 if the object *j* is chosen and 0 otherwise.

ACO ALGORITHM FOR MKP

The MKP is an interesting problem from a practical and theoretical point of view: practical, because it involves a lot of real-life and industrial problems; theoretical, because it gives several possibilities for pheromone and heuristic models. One of the basic elements of the ACO algorithm is the mapping of the problem onto a graph. We decide which elements of the problem should correspond to the nodes and the ones to the arcs. The solutions of the problem are represented by paths through the graph.

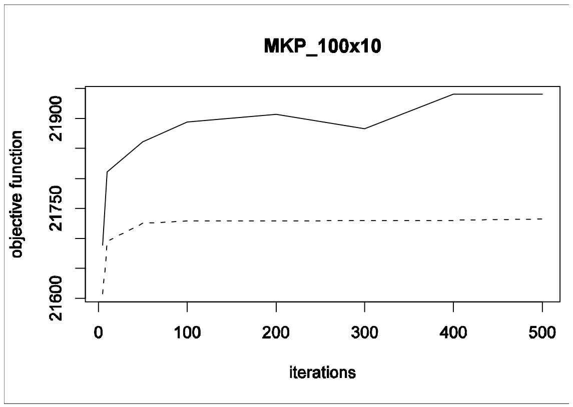
We define the graph of the MKP as follows: the nodes correspond to the objects and the arcs fully connect nodes. Fully connected graph means that after the object *I*, the object *j* might be chosen if there are enough resources and if the object *j* is not chosen yet. At each iteration, every ant constructs a solution. It first randomly chooses the initial object (node in the graph) and then iteratively adds new objects (nodes in the graph) that can be selected without violating resource constraints. Once each ant has constructed its solution, pheromone trails are updated. The pheromone model and heuristic information connected with MKP will be described in detail in the following subsections. Ants start to construct their solution from a random node. Therefore a small number of ants can be used. By experiment, it is found that between 10 and 20 ants are enough to achieve good result, without increasing the number of iterations. Thus the used computer resources such as time and memory are decreased.

Pheromone Model

To solve MKP with ACO algorithm, the key point is to decide which components of the constructed solutions should be rewarded and how to exploit these rewards when constructing new solutions. One can consider two different ways of laying pheromone trails:

- A first possibility is to lay pheromone trails on each selected node of the graph (object). The idea is to increase the desirability of some nodes so that these nodes will

Figure 1. Average solution quality: the thick line represents the pheromone on arcs and the dashed line represents the pheromone on nodes



be more likely to be selected in constructing a new solution.

- A second possibility is to lay pheromone trails on the arcs (*i*,*j*) of the graph. Here the idea is to increase the desirability to choose node *j* when the last selected node is *i*.

The first possibility is closely related to the nature of the problem, as MKP is an unordered problem. To solve MKP with ACO algorithm, Leguizamon and Michalevizc (1999) use the first possibility, while Fidanova (2003) uses the second possibility.

The two pheromone possibilities have been tested on 20 benchmarks of MKP, from OR Library, with 100 objects and 10 constraints (see http://mscmga.ms.ic.ac.uk/jeb/orlib). The number of iterations *K=500* is fixed for all the runs. For the tests we use ACS algorithm and 20 runs of each of the benchmarks. The initial pheromone parameter is fixed to $\tau_0=0.5$. The evaporation parameters are $\alpha=\rho=0.1$. The number of ants is set to be 10. As shown in Figure 1, there is very early stagnation of the algorithm by pheromone on nodes. This effect can be explained with large pheromone accumulation on some nodes, and thus the ants repeat the same solution over and over again. In the second case the pheromone is dispersed on the arcs. We will illustrate these phenomena with a small example with five objects and one constraint.

Example: $\max(x_1+3x_2+2x_3+x_4+2x_5)$

$$2x_1+x_2+3x_3+x_4+3x_5\leq 6$$

For heuristic information, let the profit of the objects be used. Thus the objects with greater profit are more desirable. The ACS is applied with one ant. Other parameters are $\tau_0=0.5$, $\alpha=\rho=0.5$. In a first iteration, let the ant start from node 1. Using probabilistic rule the achieved solution in both cases is (x_1,x_2,x_3), and the value of objective function is 6. After updating, the new amount of the pheromone is:

a. pheromone on nodes:
(3.25, 3.25, 3.25, 0.5, 0.5)

b. pheromone on arcs:

Non	3.25	0.5	0.5	0.5
0.5	*Non*	3.25	0.5	0.5
0.5	0.5	*Non*	0.5	0.5
0.5	0.5	0.5	*Non*	0.5
0.5	0.5	0.5	0.5	*Non*

In the second iteration, let the ant start from the node 2. Thus constructed by the ant, the solution in a both cases is (x_2, x_3, x_1). It is the same as in the first iteration, but achieved in different order. The new pheromone is:

a. pheromone on nodes:
(3.937, 3.937, 3.937, 0.5, 0.5)

b. pheromone on arcs:

Non	3.25	0.5	0.5	0.5
0.5	*Non*	3.937	0.5	0.5
3.25	0.5	*Non*	0.5	0.5
0.5	0.5	0.5	*Non*	0.5
0.5	0.5	0.5	0.5	*Non*

In the third iteration, let the ant start from the node 3. The achieved solution by both cases is (x_3, x_2, x_1). The new pheromone is:

a. pheromone on nodes:
(4.109, 4.109, 4.109, 0.5, 0.5)

b. pheromone on arcs:

Non	3.25	0.5	0.5	0.5
3.25	*Non*	3.937	0.5	0.5
3.25	3.25	*Non*	0.5	0.5
0.5	0.5	0.5	*Non*	0.5
0.5	0.5	0.5	0.5	*Non*

Heuristic Information

The second component in the transition probability is the heuristic information. The MKP is a constraint problem, and the constraints can be used for constructing heuristic information in various manners. There are two main types of heuristics: static and dynamic. Static heuristics remain unchanged during the run of the algorithm, while the dynamic heuristics correspond to the current state of the problem. The profit of the objects will be included in the heuristics because it is the most important information for objective function. A better result is expected when we include in the heuristics more information for the problem.

Static Heuristics

Two types of static heuristics are proposed, called "heuristics A" and "heuristics B" respectively.

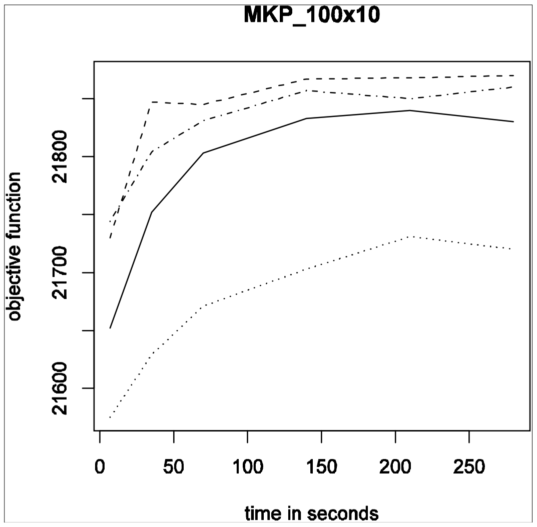
- **Heuristics A:** Let $s_j \quad {}_{i\ 1}^{m} r_{ij}$. For heuristic information we use: ${}_{ij} \quad p_j^{d_1} / s_j^{d_2}$, $0 <$ d_1 and $0 < d_1$ are parameters. The expenses of the objects are included in heuristic information. Therefore, the objects with greater profit and less average expenses are more desirable. Thus we try to have some balance between expenses and the profit for a given object.
- **Heuristics B:** Let $s_j \quad {}_{i\ 1}^{m} r_{ij} / c_i$. For heuristic information we use: ${}_{ij} \quad p_j^{d_1} / s_j^{d_2}$, $0 <$ d_1 and $0 < d_1$ are parameters. Thus the heuristic information depends on the profit, the expenses, and the budgets. The objects with greater profit, which use a lesser part of the budget, are more desirable.

Dynamic Heuristics

The third and the forth types of heuristic information are dynamic, and they correspond to the current state of the algorithm. We call them "heuristics C" and "heuristics D" respectively.

- **Heuristics C (Leguizamon & Michalevizc, 1999):** Let $b_i \quad c_i \quad {}_{j\ 1}^{n} r_{ij} x_j$ be the remainder of the budget before choosing the next object and $s_j \quad {}_{i\ 1}^{m} r_{ij} / b_i$ if $b_i \quad 0$ and $s_j \quad {}_{i\ 1}^{m} r_{ij}$ if $b_i = 0$. For heuristic information we use: ${}_{ij} \quad p_j^{d_1} / s_j^{d_2}$, where $d_1 \quad d_2$. The aim is for the heuristic information to have maximal correspondence to the current state of the algorithm and thus to achieve good result. Leguizamon and Michalevizc (1999) do not verify if $b_i \quad 0$, but because it can happen and there is division by b_i, we add this verification in the algorithm. Thus the objects with greater profit, which use less part of the available budget, will be more desirable.

Figure 2. The graphics show the average solution quality (value of the total cost of the objects in the knapsack) over 20 runs; the dash-dot line represents heuristics A, the dash line represents heuristics B, the dotted line represents heuristics C, and the thick line represents heuristics D



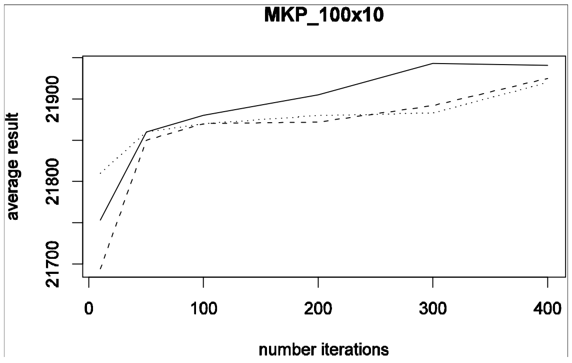
- **Heuristics D:** This is similar to heuristics C, but the parameters d_1 and d_2 can be different. By this heuristics, we can observe the influence of the parameters d_1 and d_2.

The ACO algorithms with four kind of heuristic information have been tested on 20 benchmarks of MKP, from OR Library, with 100 objects and 10 constraints. For the tests we use ACS algorithm with pheromone on arcs. The initial pheromone parameter is fixed to τ_0=0.5. The evaporation parameters are set to α=ρ=0.1. When $d_1 \neq d_2$, d_1=3 and d_2=2, otherwise d_1=d_2=1. The number of ants is set to 10. First we observe that the heuristics B shows advantage over the other tree heuristics. This means that it is more important that the expenses be a small part of the relevant budget. We expected to achieve better results by dynamic heuristics because they correspond to the current state of the problem. In spite of our expectations, we achieved weaker results by dynamic heuristics. Using dynamic heuristics, the chosen object order became important; the desirability of an object was not the same if it was chosen in the beginning of the iteration or later. The MKP is an unordered problem, and for it the order in which the objects are chosen is not important. Thus we can explain better results by static heuristics. Comparing heuristics C and D, we observe the importance of the parameters d_1 and d_2. In the case $d_1 \neq d_2$, the achieved results are better. The parameters d_1 and d_2 show the importance of the profit and the constraints in heuristic information. If d_1 is greater than d_2, then the profit is more important, and in the opposite case the constraints are more important. If both values d_1 and d_2 are great, then the heuristic information is more important than the pheromone in the transition probability.

Comparison between ACO Algorithms

The ACO algorithms are differing in pheromone updating. We compare some ACO algorithms applied on MKP. The ant colony system, the max-min ant system, and the ant algorithm with additional reinforcement have been chosen, because for them it is proven to converge to the global optimum. These ACO algorithms have been tested on 20 benchmarks of MKP with 100 objects and 10 constraints from OR Library. The reported results are average on 20 runs of each benchmark. The pheromone is laid on the arcs and the heuristics B is used. ACO-AR is applied on ant algorithm with elitist ant. The added additional pheromone is equal to the maximal added pheromone. The initial pheromone parameter is fixed to τ_0=0.5. The evapora-

Figure 3. Average solution quality: the thick line represents ACO-AR, the dotted line represents MMAS, and the dashed line represents ACS



tion parameters are $\alpha=\rho=0.1$. The minimal pheromone is set to $\tau_{min}=1000$, and the value of the maximal pheromone is equal to the approximate upper bound of the pheromone (Stützle & Hoos, 2000). The number of ants is set to 10. As shown in Figure 3, ACO-AR outperforms ACS and MMAS. By ACS and MMAS we achieve very similar results. In some of the runs, ACO-AR reaches the best found results in a literature by meta-heuristics methods.

CONCLUSION

In this chapter the ACO algorithms and their implementations on MKP are described. The MKP is represented by graph and the solutions are represented by paths through the graph. We compare two pheromone models, pheromone on the arcs of the graph of the problem and pheromone on the nodes of the graph. We observe that laying the pheromone on the arcs, the algorithm achieves better results. When the pheromone is laid on the nodes on some of them, the pheromone concentration becomes very high and ants choose them with higher probability. We compare four representations of heuristic information. Best results are achieved when the heuristic information depends on the profit, the expenses, and the budgets. The objects with greater profit, which use fewer parts of the budgets, are more desirable. We achieve better results by static heuristics than by dynamics. Using dynamic heuristics the probability to choose the same object at the beginning of the iteration is different than choosing it later, and for MKP the chosen objects order is not important. At the end we compare the results achieved by three of the ACO algorithms, proved to converge to the global optimum, ACS, ACO-AR, and MMAS. We achieve best results by ACO-AR, and in some of the runs the achieved results are equal to the best found in the literature. In the future we will investigate hybridization of the ACO algorithms, combining them with other meta-heuristic techniques and appropriate local search procedures.

ACKNOWLEDGMENT

This work is supported by the Marie Curie program of the European Community by grant MERG-CT-2004-510714.

REFERENCES

Cordón, O., Fernàndez de Viana, & Herrera, F. (2002) Analysis of the best-worst ant system ant its variations on the QAP. In M. Dorigo, G. Di Caro, & M. Sampels (Eds.), *From ant colonies to artificial ants* (pp. 228-234) (LNCS 2463). Berlin: Springer-Verlag.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). The ant system: Optimization by a colony of cooperative agents. *IEEE Transactions on*

Systems, Man and Cybernetics—Part B, 26(1), 29-41.

Dorigo, M., & Gambardella, L.M. (1999). Ant colony system: A cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computing, 1,* 53-66.

Dorigo, M., Di Caro, G., & Gambardella, M. (1999). Ant algorithms for distributed discrete optimization. *Journal of Artificial Life, 5,* 137-172.

Diffe, W., & Hellman, M.E. (1976). New direction in criptography. *IEEE Transactions in Information Theory, 36,* 644-654.

Fidanova, S. (2002). ACO algorithm with additional reinforcement. In M. Dorigo, G. Di Caro, & M. Sampels (Eds.), *From ant colonies to artificial ants* (pp. 292-293) (LNCS 2463). Berlin: Springer-Verlag.

Fidanova, S. (2002). Evolutionary algorithm for multiple knapsack problem. In D. Corn (Eds.), *Proceedings of the PPSN VII Workshops,* Granada, Spain.

Fidanova, S. (2003). ACO algorithm for MKP using various heuristic information. In I, Dimov, I. Lirkov, S. Margenov, & Z. Zlatev (Eds.), *Numerical methods and applications* (pp. 434-330) (LNCS 2542). Berlin: Springer-Verlag.

Fidanova, S. (2004). Convergence proof for a Monte Carlo method for combinatorial optimization problems. In M. Bubak, G.D. Albada, P.M.A. Sloot, & J. Dongarra (Eds.), *Computational science* (pp. 527-534) (LNCS 3039). Berlin: Springer-Verlag.

Gambardella, M. L., Taillard, E. D., & Agazzi, G. (1999). A multiple ant colony system for vehicle routing problem with time windows. In D. Corne, M. Dorigo, & F. Glover (Eds.), *New ideas in optimization* (pp. 63-76). New York: McGraw Hill.

Gambardella, M. L., Taillard, E. D., & Dorigo, M. (1999). Ant colonies for the QAP. *Journal of the Operational Research Society, 50,* 167-176.

Kochenberger, G., McCarl, G., & Wymann, F. (1974). A heuristics for general integer programming. *Journal of Decision Science, 5,* 34-44.

Leguizamon, G., & Michalevizc, Z. (1999). A new version of ant system for subset problems. *Proceedings of the International Conference on Evolutionary Computations,* Washington.

Lonnstedt, L. (1973). The use of operational research in twelve companies quoted on the Stockholm Stock Exchange. *Journal of Operational Research, 24,* 535-545.

Matrello, S., & Toth, P. A. (1984). A mixture of dynamic programming and branch-and-bound for the subset-sum problem. *Journal of Management Science, 30,* 756-771.

Moyson, F., & Manderick, B. (1988). The collective behavior of ants: An example of self-organization in massive parallelization. *Proceedings of the AAAI Spring Symposium on Parallel Models of Intelligence,* Stanford, CA.

Sinha, A., & Zoltner, A. A. (1979). The multiple-choice knapsack problem. *Journal of Operational Research, 27,* 503-515.

Stützle, T., & Hoos, H. H. (2000). Max-min ant system. In M. Dorigo, T. Stützle, & G. Di Caro (Eds.), *Future generation computer systems* (Vol. 16, pp. 889-914).

Stützle, T., & Dorigo, M. (2002). A short convergence proof for a class of ant colony optimization algorithms. *IEEE Transactions on Evolutionary Computation, 6*(4), 358-365.