# Ant Colony Optimization for Multiple Knapsack Problem and Heuristic Model

Stefka Fidanova (stefka@parallel.bas.bg)*
*IPP-BAS, Acad. G. Bonchev str. bl.25A, 1113 Sofia, Bulgaria*

**Abstract.** The Multiple Knapsack problem (MKP) is a hard combinatorial optimization problem with large application, which embraces many practical problems from different domains, including transport, like cargo loading, cutting stock, bin-packing, financial management, etc. It also arise as a subproblem in several more complex problems like vehicle routing problem and the algorithms to solve these problems will benefit from any improvement in the field of MKP. The aim of this paper is to compare different kind of heuristic models, statics and dynamics. The heuristics are used by an Ant Colony Optimization (ACO) algorithm to construct solutions to the MKP.

**Keywords:** combinatorial optimization, multiple knapsack problem, heuristics

## 1. Introduction

Combinatorial optimization is the process of finding the best, or optimal solution for problems with a discrete set of feasible solutions. Applications arise in numerous settings involving operations management and logistics. The economic impact of combinatorial optimization is profound, affecting diverse sectors. While much progress has been made in finding exact solutions to some Combinatorial Optimization Problems (COPs), many hard combinatorial problems are still not solved exactly in a reasonable time and require good heuristic methods. The aim of heuristic methods for COPs is to quickly produce good-quality solutions. Modern heuristics include simulated annealing [7], genetic algorithms [4], tabu search [5], ant colony optimization [2, 3]. 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. For all these reasons, metaheuristics have probably been one of the most stimulating research topics in optimization for the last two decades.

The ACO algorithm were inspired by the observation of real ant colonies. Ants are social insects, that is , insects that live in colonies and whose behavior is directed 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

* Supported by European Community by the grant MERG-CT-2004-510714

find the shortest path between food sources and their nest. ACO is the recently developed, population-based approach which has been successfully applied to several NP-hard COPs. One of its main ideas is the indirect communication among the individuals of a colony of agents, called "artificial" ants, based on an analogy with trails of a chemical substance, called pheromone which real ants use for communication.

ACO uses a colony of artificial ants that behaves as cooperative agents in a mathematical space where they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. Unlike the real-life case, these pathways might contain very complex information. When constructing an initial solution, at each step ants compute a set of feasible moves and select the best one according to some probabilistic rules. These rules are based on the heuristic information and pheromone trail level. The higher value of the pheromone and the heuristic information, the more profitable is to select this move and resume the search.

In this paper ACO algorithm for MKP is investigated. The MKP is an important real-life problem. Problems from different industrial fields, including transport and logistics, can be interpreted like knapsack problem. When the ACO algorithm is applied MKP gives a possibility to use varied heuristic informations. We have concentrated on the construction of static and dynamic heuristics for MKP. Our aim is to study different heuristics and their influence on the result.

The remainder of this paper is structured as follows. In section 2 we describe MKP. Section 3 investigates the application of the ACO algorithm with different types of heuristic information for solving MKP. In section 4 we show experimental results over some test problems. Finally we draw some conclusions.

## 2. Formulation of the problem

The Knapsack Problem has numerous applications in theory as well as in practice. It also arise as a subproblem in several algorithms for more complex COPs and these algorithms will benefit from any improvement in the field of MKP. We can mention the following major applications: problems in cargo loading, cutting stock, bin-packing, budget control and financial management may be formulated as MKP. In [10] is proposed to use the MKP in fault tolerance problem and in [1] is designed a public cryptography scheme whose security realize on the difficulty of solving the MKP. Martello and Toth [9] mention that two-processor scheduling problems may be solved as a MKP. Other applications are industrial management, naval, aerospace, 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 part of the theory of computation dealing with the resources required during the computation to solve a given problem.

The more theoretical application either appear where a general problem is transformed to a MKP or where a MKP appears as a subproblem. We should mentioned that MKP appears as a subproblem when solving the generalized assignment problem, which again is heaving used when solving vehicle routing problems. In addition, MKP can be seen as a general model for any kind of binary problems with positive coefficients [7].

The MKP can be thought as a resource allocation problem, where we have $m$ 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:

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

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.

The partial solution is represented by $S = \{i_1, i_2, \ldots, i_j\}$, where $x_{i_s} = 1 \ \ (s = 1,\ldots,j)$, and the most recent element incorporated to $S$, $i_j$ need not be involved in the process of selecting the next element. One of the basic elements of the ACO metaheuristic is the mapping of the problem onto a graph, thus the solutions of the problem are represented by paths through the graph. We define the graph of the problem as follows: the nodes correspond to the objects, the arcs fully connect nodes. Fully connected graph mens that after the object $i$ we can chose the object $j$, for every $i$ and $j$ if the object $j$ is not chosen yet.When walk through the graph the ants put a pheromone on the arcs.

## 3. ACO for MKP

The ACO algorithms were inspired by the observation of real ant colonies [2, 3]. An interesting behavior is how ants can find the shortest paths between food sources and their nest. While walking from a food source to the nest and vice-versa, ants deposit on the ground a substance called pheromone. Ants can smell pheromone and then they tend to choose, in probability, paths marked by strong pheromone concentrations. The pheromone trail allows the ants to find their way back to the food source (or to the nest).

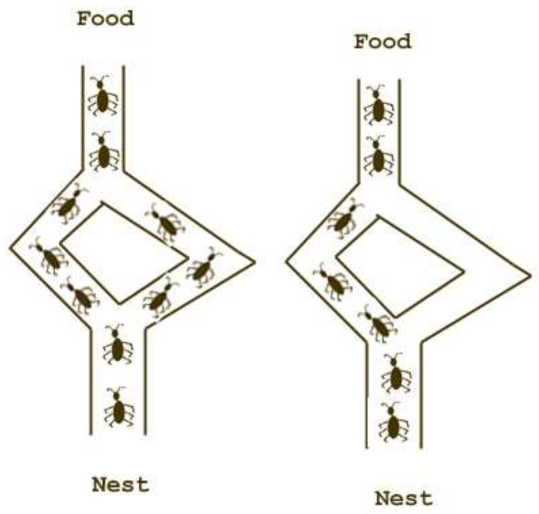


*Figure 1.* Behavior of real ants at the beginning of the search and after some minutes

Figure 1 shows how ants can exploit pheromone to find a shortest path between two points. In this, figure ants arrive at a decision point in which they have to decide to turn on the left or on the right. The ants turning on the left first achieve the food sours. When they return back there is a pheromone only in a left side and they choose it and double the pheromone. Thus, after a short transition period the difference in the amount of pheromone on the two paths is sufficiently large and the new ants will prefer in probability to choose the left path, since at the decision point they receive a greater amount of pheromone on the left path. Very soon all ants will be using the shorter path.

The above behavior of real ants has inspired ACO algorithm. ACO algorithm, which is a population-based approach, has been successfully applied to many NP-hard problems [2, 3]. One of its main ideas is the indirect communication among the individuals of ant colony. This mechanism is based on an analogy with trails of pheromone which real ants use for communication. The pheromone trails are a kind of distributed numerical information which is modified by the ants to reflect their experience accumulated while solving a particular problem.

ACO algorithm make use of simple agents called ants which iteratively construct candidate solutions to a combinatorial optimization problem [2]. The ants' solution construction is guided by (artificial) pheromone trail and problem dependent heuristic information. ACO algorithm can be applied to any combinatorial optimization problem by defining solution components which the ants use to iteratively construct candidate solutions and on which they may deposit a pheromone. Partial problem solution are seen as states: each ant moves from a state $i$ to another state $j$ corresponding to a more complete partial solution. At each step, each ant $k$ computes a set of feasible expansion to its current state and moves to one of these according to a probability distribution.

For ant $k$, the probability $p_{ij}^k$ of moving from a state $i$ to a state $j$ depends on the combination of two values:

- The attractiveness $\eta_{ij}$ of the move as computed by some heuristic;
- The pheromone trail level $\tau_{ij}$ of the move.

The probability $p_{ij}^k$ of selecting $j$ as the next state is given as:

$$p_{ij}^k(t) = \begin{cases} \frac{\tau_{ij}\eta_{ij}(S_k(t))}{\sum_{q \in allowed_k(t)} \tau_{iq}\eta_{iq}(S_k(t))} & \text{if } j \in allowed_k(t) \\ 0 & \text{otherwise} \end{cases} , \tag{2}$$

where

- $\tau_{ij}$ is a pheromone level on arc corresponded to move from $i$ to $j$;
- $\eta_{ij}(S_k(t))$ is the heuristic;
- $allowed_k(t)$ is the set of remaining feasible states.

Thus the higher the value of $\tau_{ij}$ and $\eta_{ij}(S_k(t))$ the more profitable it is to include state $j$ in the partial solution. We use a particular implementation of ACO algorithm, known as ant colony system [3].

Pheromone trails are updated in two stages. In the first each ant applies the following local update rule:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\tau_0, \tag{3}$$

where $\rho$, $0 < \rho \leq 1$ is the pheromone decay parameter, it models evaporation in a nature, and $\tau_0$ is the initial pheromone value. The effect of the local updating rule is to decrease the pheromone on arcs which have already been chosen by an ant, so they become less desirable. It is a kind of diversification of the search in the search space.

When all ants have terminated their tour, the pheromone is modified on the arcs belonging to the best tour since the beginning of the trial by the following global updating rule:

$$\tau_{ij} \leftarrow (1-\alpha)\tau_{ij} + \alpha\Delta\tau, \tag{4}$$

where

$$\Delta\tau = Objective.Function(best), \tag{5}$$

- $\alpha$, $0 < \alpha \leq 1$ is the pheromone decay parameter;
- $Objective.Function(best)$ is the value of objective function of the best-so-far tour.

When we calculate the selection probability we can use various types of heuristic information. It can be static and dynamic, and we can use different problems' parameters for heuristic. Static heuristic stays unchanged during the run of the algorithm, while the dynamic heuristic corresponds to the current state of the problem. Obviously, we will include the profit in the heuristics because it is the most important information for objective function. If we include more information for the problem in heuristic, we will achieve better results. Thus we will try to use the constraints in different manners and will compare the results.

### 3.1. Static Heuristics

We propose two types of static heuristics using constraints. We will call them "heuristic A" and "heuristic B" respectively.

- **Heuristic A**. Let $s_j = \sum_{i=1}^{m} r_{ij}$. For heuristic information we use:

$$\eta_{ij} = p_j^{d_1} / s_j^{d_2}, \tag{6}$$

$0 < d_1$ and $0 < d_2$ are parameters. We include the expenses in heuristic. Hence the objects with greater profit and less average expenses will be more desirable. Thus we try to do some balance between expenses and the profit for a given object.

- **Heuristic B**. Let $s_j = \sum_{i=1}^{m} r_{ij}/c_i$. For heuristic information we use:

$$\eta_{ij} = p_j^{d_1} / s_j^{d_2}, \tag{7}$$

$0 < d_1$ and $0 < d_2$ are parameters. Thus the heuristic depends by the profit, the expenses and the budgets. The objects with greater profit which use less parts of the budgets will be more desirable.

### 3.2. Dynamic Heuristics

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

- **Heuristic C** [8]. Let $b_i = c_i - \sum_{j=1}^{n} r_{ij} x_j$ is the remainder of the budget before choosing the next object, and $s_j = \sum_{i=1}^{m} r_{ij}/b_i$ if $b_i \neq 0$ and $s_j = \sum_{i=1}^{m} r_{ij}$ if $b_i = 0$. For heuristic information we use:

$$\eta_{ij} = p_j^{d_1} / s_j^{d_2}, \tag{8}$$

where $d_1 = d_2$. The aim is the heuristic information to have maximal correspondence to the current state of the algorithm and thus to achieve good result. In [8] the authors do not verify if $b_i \neq 0$, but because it can happen and there is division by $b_i$, we add this verification in the algorithm.

- **Heuristic D.** For heuristic information we use:

$$\eta_{ij} = p_j^{d_1} / s_j^{d_2}, \tag{9}$$

$0 < d_1$ and $0 < d_2$ are parameters and they can be different.

The mining of $s_j$ is like in heuristic C.

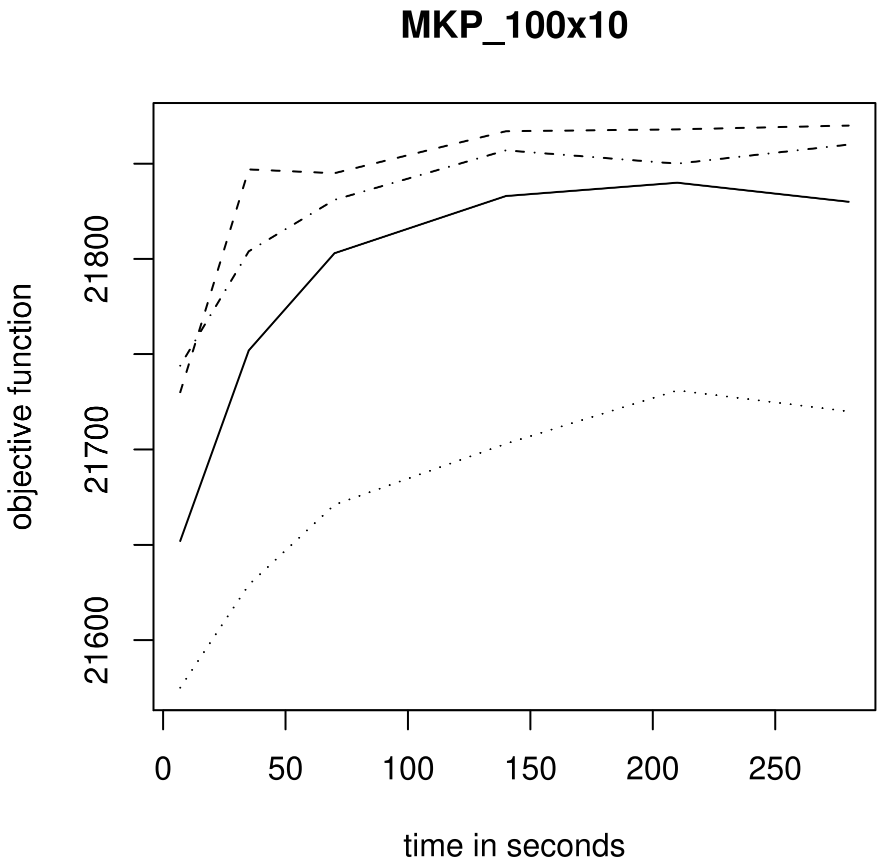


*Figure 2.* The graphics shows the average solution quality (value of the total cost of the objects in the knapsack) over 20 runs. Dash dot line represents heuristic A,dash line - heuristic B, dots line - heuristic C and thick line - heuristic D.

The parameters $d_1$ and $d_2$ show the importance of the profit and the constraints in heuristic information. If $d_1 > d_2$, than the profit is more important and in opposite case the constraints are more important. If we use greater values for $d_1$ and $d_2$, than the heuristic information is more important than the pheromone in the transition probability.

## 4. Experimental results

We were running experiments of our ACO algorithm for all four heuristics on 5 different instances from "OR-Library" available within WWW access at **http://mscmga. ms.ic.ac.uk/jeb/orlib**, with 100 objects and 10 constraints. For all 5 instances we were running experiments for a range of evaporation rates and the parameters $d_1$ and $d_2$ in order to find the best rate for every instance. The results of this experiments are shown in Figure 2. We fixed the initial pheromone value to be $\tau_0 = 0.5$. The developed technique has been coded in $C++$ language and implemented on a Pentium III (450MHz). Because of the random start of the ants in every iteration we can use less ants than the number of objects. After the tests we found that 10 ants are enough to achieve good result for a reasonable time. After choosing for every problem instance the best rate for parameters we could compare the different heuristic representations. In Figure 2 we show average results over all 5 problem instances and every instance is run 20 times with the same parameter settings. To can observe the influence of the heuristics better we use ACO algorithm without local search or other methods to improve the results and we run the different test instances on same random sequences. First our observation is that the heuristic B shows advantage over the other three heuristics representations. It means that it is important the chosen objects to have small expenses, but it is more important the expenses to 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 achieve weaker results by dynamic 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. Nevertheless we do not use any method to improve the results we achieve results close to the best found in the literature.

## 5. Conclusion

In this work we compare fore types of heuristics, statics and dynamics, for ACO algorithms to solve MKP, a problem with large application in real life and industry including transportation and logistics. We observed that using static heuristics result in improved performance. This means that it doesn't seem beneficial to use dynamic heuristics. They are more resource consuming without achieving better result. It is more beneficial to learn the influence of static heuristics for the result to be strongly increasing from the beginning. For future work important direction for current research is to try different strategies for management of the search process more effectively and to provide good results.

## References


1. Diffe W., Hellman M. E. : New direction in cryptography. IEEE Trans, Inf. Theory, IT - 36,(1976), 644–654.
2. Dorigo M., Di Caro G. : The ant colony optimization metaheuristic. In: Corne D., Dorigo M., Glover (eds.): New Ideas in Optimization, McCrow-Hill (1999), 11–32.
3. Dorigo M., Gambardella L. M.: Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1 (1999), 53–66.
4. Fogel B.: Evolutionary computation: Toward a new philosophy of machine intelligence. IEEE Press, New York, (1994).
5. Glover F.: Tabu search, ORSA J. of Computing 1 (1989).
6. Kirkpatrick S., Gelatt C. D., Vechi M. P.: Optimization by simulated annealing. Science 220, (1983), 671–680.
7. Kochemberger G., McCarl G., Wymann F.:Heuristic for general inter programming, J. of Decision Sciences 5, (1974), 34–44.
8. Leguizamon G., Michalewicz Z.: A new version of ant system for subset problems, in proceedings of the Congress on Evolutionary Computing, July 6-9, (1999), 1459–1464.
9. Martello S., Toth P.: A mixtures of dynamic programming and branch-and-bound for the subset-sum problem, Management Science, 30, (1984), 765–771.
10. Sinha A., Zoltner A. A. : The multiple-choice knapsack problem, J. of Operational Research, 27, (1979), 503–515.