Monte Carlo Method for Multiple Knapsack Problem

Stefka Fidanova

CLPP – BAS, Acad. G. Bonchev str. bl.25A, 1113 Sofia, Bulgaria
fidanova@parallel.bas.bg

Abstract. This paper describes Monte Carlo (MC) method for Multiple Knapsack Problem (MKP). The MKP can be defined as economical problem like resource allocation and capital budgeting problems. The Ant Colony Optimization (ACO) is a MC method, created to solve Combinatorial Optimization Problems (COPs). The paper proposes a Local Sear (LC) procedure which can be coupled with the ACO algorithm to improve the efficiency of the solving of the MKP. This will provide optimal or near optimal solutions for large problems with an acceptable amount of computational effort. Computational results have been presented to assess the performance of the proposed technique.

1 Introduction

There are many NP-hard combinatorial optimization problems for which it is impractical to find an optimal solution. Among them is the MKP. For such problems the reasonable way is to look for algorithms that quickly produce near-optimal solutions. ACO [1, 3, 2] is a MC method, created to solve COPs. It is a metaheuristic procedure for quickly and efficiently obtaining high quality solutions to complex optimization problems [7]. ACO algorithm can be interpreted as parallel replicated Monte Carlo systems [10]. MC systems [8] are general stochastic simulation systems, that is, techniques performing repeated sampling experiments on the model of the system under consideration by making use of a stochastic component in the state sampling and/or transition rules. Experimental results are used to update some statistical knowledge about the problem, as well as the estimate of the variables the researcher is interested in.In turn, this knowledge can be also iteratively used to reduce the variance in the estimation of the described variables, directing the simulation process toward the most interesting state space regions. Analogously, in ACO algorithms the ants sample the problem's solution space by repeatedly applying a stochastic decision policy until a feasible solution of the considered problem is built. The sampling is realized concurrently by a collection of differently instantiated replicas of the same ant type. Each ant "experiment" allows to adaptively modify the local statistical knowledge on the problem structure. The recursive retransmission of such knowledge by means of stigmergy determines a reduction in the variance of the whole search process the so far most interesting explored transitions probabilistically bias future search, preventing ants to waste resources in not promising regions of the search.

The ACO algorithms 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 then to that of a single individual component of the colony. An important and interesting of ant colonies is how ants can 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 [4]. 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. The "artificial" 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. When construct a solution, at each step ants compute a set of feasible moves and select the best according to some probabilistic rules. The transition probability is based on the heuristic information and pheromone trail level of the move (usability of the move in the past).

The rest of the paper is organized as follows: Section 2 describes the general framework for MKP as a COP. The search strategy of the local search method is explained in Section 3. Section 4 outlines the implemented ACO applied to MKP. In Section 5 experimental results over test problems are shown. Finally some conclusions are drown.

2 Formulation of the Problem

The MKP has received wide attention from operation research community, because it embraces many practical problems. Applications include resource allocation in distributed systems, capital budgeting and cutting stock problems. In addition, MKP can be seen as a general model for any kind of binary problems with positive coefficients [6, 5]. The MKP can be thought as a resource allocation problem, where we have m resources (the knapsacks) and n objects. Te object j has a profit p_j Each resource has its own budget (knapsack capacity) and consumption r_{ij} of resource j by object i. We are interested in maximizing the profit, 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.

In the MKP we are not interested in solutions giving a particular order. Therefore a partial solution is represented by $S = \{i_1, i_2, \ldots, i_j\}$ and the most recent elements incorporated to S, i_j need not be involved in the process for selecting the next element. Moreover, solutions for ordering problems have a fixed length as we search for a permutation of a known number of elements. Solutions for MKP, however, do not have a fixed length. We define the graph of the problem as follows: the nodes correspond to the items, the arcs fully connect nodes.

3 The Local search Strategy

The LS method (move-by-move method) [9] perturbs a given solution to generate different neighborhoods using a move generation mechanism. In general, neighborhoods for large-sized problems can be much complicated to search. Therefore, LS attempts to improve an initial solution by a series of local improving changes. A move-generation is a transition from a solution S to another one $S' \in V(S)$ in one step. These solutions are selected and accepted according to some predefined criteria. The returned solution S' may not be optimal, but it is the best solution in its local neighborhood $V(S)$. A local optimal solution is a solution with the local maximal possible cost value. Knowledge of a solution space is the essential key to more efficient search strategies. These strategies are designed to use this prior knowledge and to overcome the complexity of an exhaustive search by organizing searches across the alternatives offered by a particular representation of the solution. The main purpose of LS implementation is to speed up and improve the solutions constructed by the ACO.

A practical LS strategy that satisfies the MKP requirements has been developed and coupled with ACO algorithm. The MKP solution can be represented by string with 0 for not chosen items and 1 for chosen items. In our LS procedure we will replace one of the 1 with 0 and one of the 0 with 1. The change in cost is completed an the swap is accepted or rejected according to the acceptance strategy.

4 ACO Algorithm for MKP

Real ants foraging for food lay down quantities of pheromone (chemical cues) marking the path that they follow. An isolated ant moves essentially at random but an ant encountering a previously laid pheromone will detect it and decide to follow it with high probability and thereby reinforce it with a further quantity of pheromone. The repetition of the above mechanism represents the auto-catalytic behavior of real ant colony where the more the ants follow a trail, the more attractive that trail becomes.

The ACO algorithm uses a colony of artificial ants that behave as co-operative agents in a mathematic space were they are allowed to search and reinforce pathways (solutions) in order to find the optimal ones. After initialization of the pheromone trails, ants construct feasible solutions, starting from random nodes, then the pheromone trails are updated. At each step ants compute a set of feasible moves and select the best one (according to some probabilistic rules) to carry out the rest of the tour. The transition probability is based on the heuristic information and pheromone trail level of the move. The higher value of the pheromone and the heuristic information, the more profitable is to select this move and resume the search. In the beginning, the ant do feasible moves and select the best one (according to some probabilistic rules) to carry out the rest of the tour. In the beginning, the initial pheromone level is set to a small positive constant value τ_0 and then ants update this value after completing the construction stage. ACO algorithms adopt different criteria to update the pheromone level. In our implementation we use Ant Colony System (ACS) [3] approach.

In ACS the pheromone updating stage consists of:

4.1 Local Update Stage

While ants build their solution, at the same time they locally update the pheromone level of the visited paths by applying the local update rule as follows:

$$\tau_{ij} \leftarrow (1-\rho)\tau_{ij} + \rho\tau_0 \tag{2}$$

Where τ_{ij} is an amount of the pheromone on the arc (i,j) of the graph of the problem, ρ is a persistence of the trail and term $(1-\rho)$ can be interpreted as trail evaporation.

The aim of the local updating rule is to make better use of the pheromone information by dynamically changing the desirability of edges. Using this rule, ants will search in wide neighborhood of the best previous solution. As shown in the formula, the pheromone level on the paths is highly related to the value of evaporation parameter ρ. The pheromone level will be reduced and this will reduce the chance that the other ants will select the same solution and consequently the search will be more diversified.

4.2 Global Updating Stage

When all ants have completed their solution, the pheromone level is updated by applying the global updating rule only on the paths that belong to the best solution since the beginning of the trail as follows:

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

where $\Delta\tau_{ij} = \begin{cases} L_{gb} \text{ if } \ (i,j) \in \text{best solution} \\ \\ 0 \quad \text{otherwise} \end{cases}$,

L_{gb} is the cost (profit) of the best solution from the beginning. This global updating rule is intended to provide a greater amount of pheromone on the paths of the best solution, thus intensifying the search around this solution. The transition probability to select the next item is given as:

$$prob_{ij}^{k}(t) = \begin{cases} \frac{\tau_{ij}\eta_{ij}}{\sum_{j \in allowed_k(t)} \tau_{ij}\eta_{ij}} & \text{if} \quad j \in allowed_k(t) \\ \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where τ_{ij} is a pheromone level on the arc (i, j), η_{ij} is the heuristic and $allowed_k(t)$ is the set of remaining feasible items. Thus the higher the value of τ_{ij} and η_{ij}, the more profitable it is to include item j in the partial solution.

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

$$\eta_{ij} = \begin{cases} p_j^{d_1}/s_j^{d_2} & \text{if} \quad s_j \neq 0 \\ \\ p_j^{d_1} & \text{if} \quad s_j = 0 \end{cases} \tag{5}$$

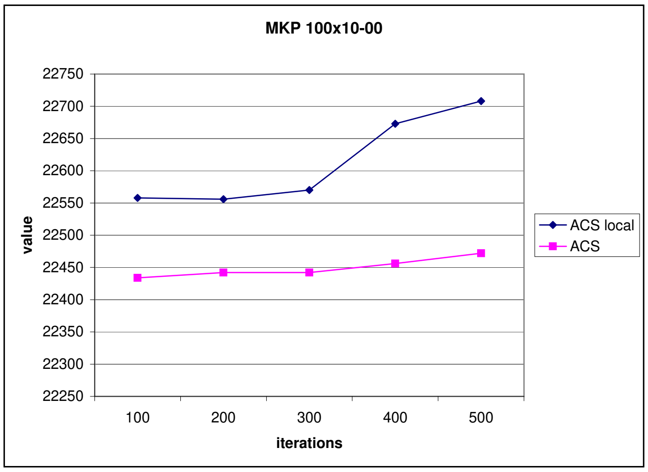
where $d_1 > 0$ and $d_2 > 0$ are parameters. Hence the objects with greater profit and less average expenses will be more desirable.

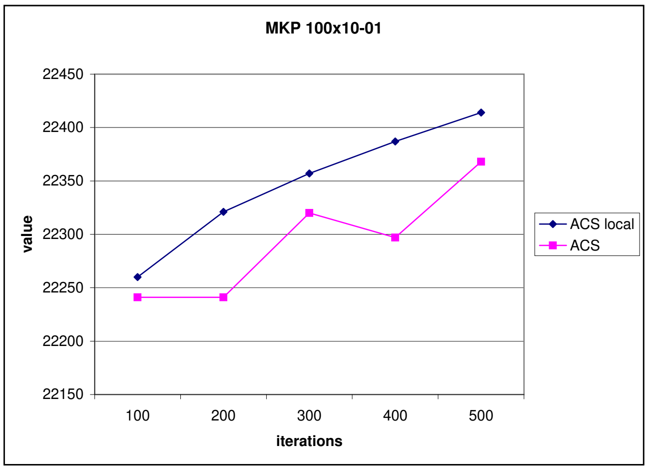
5 Experimental results

This section reports on the computational experience of the ACS coupled with local search procedures using large MKP from "OR-Library" available within WWW access at *http://mscmga.ms.ic.ac.uk/jeb/orlib*. To provide a fair comparison for the above implemented ACS algorithm, a predefined number of iterations, K=500, is fixed for all the runs. The developed technique has been coded on $C++$ language and implemented on a Pentium 3 (450 MHz). Because of the random start of the ants we can use less ants then the number of the nodes (objects). After the tests we found that 10 ants are enough to achieve good results. Thus we decrease the running time of the program. The results on the Figure 1 shows the advantage to use local search procedures. The reported results are average over 5 runs of the program. The chosen LS strategy appears to be very effective for MKP. For all tested problems our algorithm achieved, in some of the runs, the best found results in the literature. In all tested problems achieved results using LS procedure are better than without local search.

6 Conclusion

In this paper a local search procedure have been developed. The comparison of the performance of the ACS coupled with this procedure applied to different MKP problems are reported. The results obtained are encouraging and the ability of the developed technique to generate rapidly high-quality solutions for


MKP 100x10-00
22750
22700
22650
22600
22550
22500
22450
22400
22350
22300
22250
value
100
200
300
400
500
iterations
ACS local
ACS


MKP 100x10-01
22450
22400
22350
22300
22250
22200
22150
value
100
200
300
400
500
iterations
ACS local
ACS

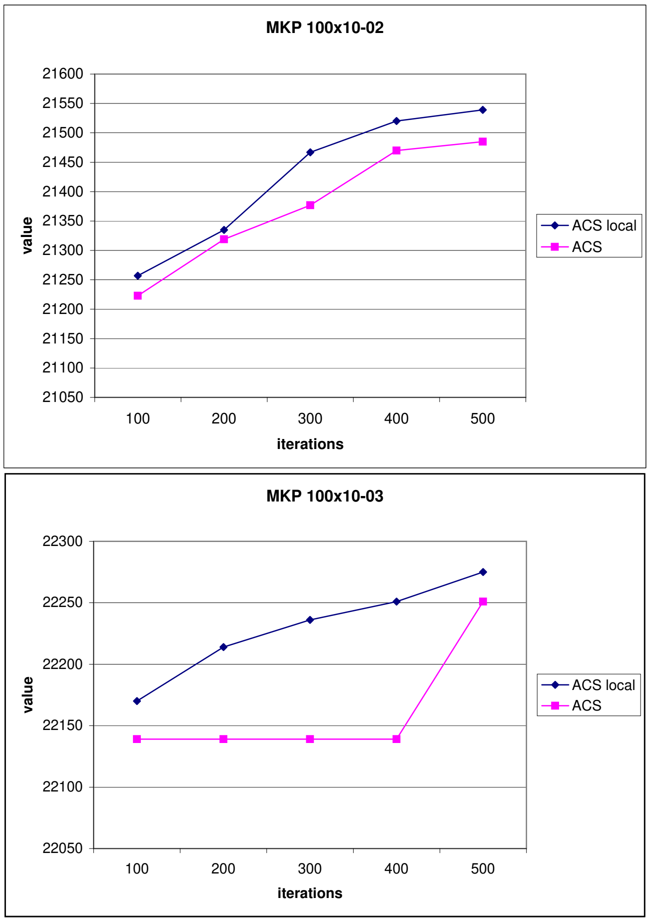


Fig. 1. The figures shows the average solution quality per iteration over 5 runs

MKP can be seen. For future work another important direction for current research is to try different strategies to explore the search space more effectively and provide good results.

Acknowledgments

Stefka Fidanova was supported by the European Community program "Center of Excellence" under contract number No ICA1-2000-70016.

References

1. Dorigo, M., Di Caro, G.:*The Ant Colony Optimization metaheuristic*.In: Corne, D., Dorigo, M., Glover, F.(eds.): New Idea in Optimization, McGrow-Hill (1999) 11–32.
2. Dorigo, M., Di Caro, G., Gambardella, L. M.:*Ant Algorithms for Distributed Discrete Optimization*. J of Artificial Life 5 (1999) 137–172.
3. Dorigo, M., Gambardella, L.M.:*Ant Colony System: A cooperative Learning Approach to the Traveling Salesman Problem*. IEEE Transaction on Evolutionary Computation 1 (1999) 53–66.
4. Gambardella, M. L., Taillard, E. D., Dorigo, M.: *Ant Colonies for the QAP*. J of Oper. Res. Soc. 50 (1999) 167–176.
5. Kochenberger, G., McCarl, G., Wymann, F.:*A Heuristic for General Integer Programming*. J of Decision Sciences 5 (1974) 34–44.
6. Leguizamon, G., Michalevich, Z.: *A New Version of Ant System for Subset Problems* In: Proceedings of Int. Conf. on Evolutionary Computations, Washington (1999).
7. Osman, I. H., Kelley, J. P.:*Metaheuristic: An Overview*. In: Osman, I. H., Kelley, J.P. (eds.), Metaheuristic: Theory and Applications, Kluwer Academic Publishers (1996).
8. Rubinstein, R. Y.:*Simulation and the Monte Carlo Method*. John Wiley &Sons(1981).
9. Schaffer, A. A., Yannakakis, M.: *Simple Local Search Problems that are Hard to Solve*. Society for Industrial Applied Mathematics Journal on Computing, Vol 20 (1991) 56–87.
10. Strelsov, S., Vakili, P.:*Variance Reduction Algorithms for Parallel Replicated Simulation of Uniformized Markov chains*. J. of Discrete Event Dynamic Systems: Theory and Applications, 6(1996) 159–180.