ACO Algorithm for MKP Using Various Heuristic Information

Stefka Fidanova

IRIDIA - Université Libre de Bruxelles, Av. Roosevelt 50 - Bruxelles, Belgium
fidanova@ulb.ac.be

Abstract. The ant colony optimization (ACO) algorithms are being applied successfully to diverse heavily constrained problems: traveling salesman problem, quadratic assignment problem. Early applications of ACO algorithms have been mainly concerned with solving ordering problems. In this paper, the principles of the ACO algorithm are applied to the multiple knapsack problem (MKP). In the first part of the paper we explain the basic principles of ACO algorithm. In the second part of the paper we propose different types of heuristic information and we compare the obtained results.

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 only reasonable way is to look for heuristic algorithms that quickly produce good, although not necessarily optimal, solutions. Many researchers have focused their attention on a new class of algorithms called metaheuristic. Metaheuristic are rather general algorithmic frameworks that can be applied to several different optimization problems with few modifications. Examples of metaheuristic are simulated annealing [10], evolutionary computation [6] and tabu search [9]. Metaheuristic are often inspired by natural processes. The above-cited metaheuristic were inspired, respectively, by the physical annealing process, the Darwinian evolutionary process and the clever management of memory structures. One of the most recent nature-inspired metaheuristic is the ant colony optimization [2, 3, 5].

Ant 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. Social insects have captured the attention of many scientist because of the high structuration level their colonies can achieve, especially when compared to the relative simplicity of the colony's individuals. An important and interesting behavior of ant colonies is their foraging behavior, and in particular, how ants can find the shortest paths between food sources and their nest.

ACO is the recently developed, population-based approach which has been successfully applied to several NP-hard combinatorial optimization problems [4, 8, 13]. 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.

Early experiments with the ACO algorithms were connected with ordering problems such as the traveling salesman problem. In this paper we discuss ACO algorithm for multiple knapsack problem. We use a particular implementation of ACO algorithm, known as ant colony system (ACS). We propose different types of heuristic information and we compare obtained results.

The remainder of this paper is structured as follows. In section 2 we describe ACO algorithm. Section 3 of this paper investigates the applicability of the ACO algorithm for solving MKP. In section 4 we propose different types of heuristic information. In section 5 we show experimental results over some test problems. Finally we draw some conclusions.

2 ACO Algorithm

ACO algorithm make use of simple agents called ants which iteratively construct candidate solutions to a combinatorial optimization problem. The ants' solution construction is guided by (artificial) pheromone trail and problem dependent heuristic information. ACO algorithms 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. An individual ant constructs candidate solution by starting with an empty solution and then iteratively adding solution components until a complete candidate solution is generated. We will call each point at which an ant has to decide which solution component to add to its current partial solution a choice point. After the solution construction is completed, the ants give feedback on the solutions they have constructed by depositing pheromone on solution components which they have used in their solution. After that we reinforce the pheromone on the components of the best found solution. Typically, solution components which are part of better solutions or are used by many ants will receive a higher amount of pheromone and hence will more used by the ants in future iterations of the algorithm. To avoid the search getting stuck, typically before the pheromone trails get reinforced all pheromone trails are decreased.

More precisely, partial problem solutions 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 specified as follows.

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

1. The attractiveness η_{ij} of the move, as computed by some heuristic indicating the a priority desirability of that move;

2. The pheromone trail level τ_{ij} of the move, indicating how profitable it has been in the past to make that particular move: it represents therefore an a posterior indication of the desirability of that move.

3 ACO Algorithm for MKP

In the beginning ACO algorithms is made for ordering problems. They have initially been tested on the traveling salesman problem [5, 3]. MKP is quite different from ordering problems. We can formulate MKP as:

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

There are m constraints in this problem, so MKP is also called m-dimensional knapsack problem. Let $I = \{1, \dots, m\}$ and $J = \{1, \dots, n\}$, with $c_i \ge 0$ for all $i \in I$. A well-stated MKP assumes that $p_j > 0$ and $r_{ij} \le c_i \le \sum_{j=1}^{n} r_{ij}$ 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.

MKP can be thought as a resource allocation problem, where we have m resources (the knapsacks) and n objects. Each resource has its own budget (knapsack capacity) and r_{ij} represents the consumption of resource j by object i. We are interested in maximizing the profit, while working with a limited budget.

MKP has received wide attention from the 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 [11].

In MKP we are not interested in solutions giving a particular order. Therefore a partial solution is represented by $\tilde{S} = \{i_1, i_2, \dots, i_j\}$ and the most recent elements incorporated to $\tilde{S}$, 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. The pheromone trail is laid on the visited arcs. For a partial solution $\tilde{S}_k = \{i_1, i_2, \dots, i_j\}$ being built by ant k, the probability $p^k_{i_p}(t)$ of selecting i_p as the next item is given as:

$$p^k_{i_p}(t) = \begin{cases} \frac{\tau_{i_j i_p}\eta_{i_p}(\tilde{S}_k(t))}{\sum_{i_q \in allowed_k(t)} \tau_{i_j i_q}\eta_{i_q}(\tilde{S}_k(t))} & \text{if } i_p \in allowed_k(t) \\ 0 & \text{otherwise} \end{cases} , \qquad (2)$$

where $\tau_{i_j i_p}$ is a pheromone level on the arc (i_j, i_p), $\eta_{i_p}(\tilde{S}_k(t))$ is the heuristic and $allowed_k(t)$ is the set of remaining feasible items. Thus the higher the value of $\tau_{i_j i_p}$ and $\eta_{i_p}(\tilde{S}_k(t))$, the more profitable it is to include item i_p in the partial solution.

While building a solution, the ants visit the arcs and change their pheromone level by applying the local updating rule:

$$\tau_{i_j i_p} \leftarrow (1-\rho)\tau_{i_j i_p} + \rho\tau_0 \quad , \qquad (3)$$

where $0 < \rho < 1$ and τ_0 are constants.

After all ants have completed their tours, global updating is performed. The pheromone level is updated by applying the global update rule:

$$\tau_{i_j i_p} \leftarrow (1-\alpha)\tau_{i_j i_p} + \alpha\Delta\tau_{i_j i_p} \quad , \qquad (4)$$

where $\Delta\tau_{i_j i_p} = \begin{cases} L_{gb} & \text{if } (i_j, i_p) \in \text{global best tour} \\ 0 & \text{otherwise} \end{cases} ,$

$0 < \alpha < 1$, L_{gb} is the best value of objective function from the beginning of the trial.

4 Different Types of Heuristic Information

When we calculate the selection probability we can use different types of heuristic information. We can use different problems' parameters for heuristic.

4.1 Heuristic Using Only the Profit

The first type of heuristic use only the profit of the objects. In this case $\eta_j = p_j^{d_1}$. Thus the objects with greater profit will be more desirable. We will call this heuristic *heuristic1*. We use the constraints only to verify if the solution is feasible.

4.2 Heuristic Using Constraints

The second type of heuristic use not only the profit of the objects. It use also the constraints. We propose two types of heuristics using constraints. We will call them *heuristic2* and *heuristic3* respectively.

– Heuristic2. Let $s_j = \max_i(r_{ij})$. For heuristic we use:

$$\eta_j = \begin{cases} p_j^{d_1}/s_j^{d_2} & \text{if } \max_i(r_{ij}) \neq 0 \\ p_j^{d_1} & \text{if } \max_i(r_{ij}) = 0 \end{cases} . \tag{5}$$

Hence the objects with greater profit and less maximal expenses will be more desirable.

– Heuristic3. Let $s_j = \sum_{i=1}^{m} r_{ij}$. For heuristic information we use:

$$\eta_j = \begin{cases} p_j^{d_1}/s_j^{d_2} & \text{if } s_j \neq 0 \\ p_j^{d_1} & \text{if } s_j = 0 \end{cases} . \tag{6}$$

Hence the objects with greater profit and less average expenses will be more desirable.

5 Experimental Results

We tested our algorithms on the set of 30 large MKP from "OR-Library" available within WWW access at *http://mscmga.ms.ic.ac.uk/jeb/orlib*. When we made the experiments we found the best results when $\rho = \alpha \in [0.01, 0.9]$ and $d_1, d_2 \in [1, 9]$. For other parameters we use $\tau_0 = 1$ and n ants. We stop the algorithms after 200 iterations.

We can conclude that algorithms with heuristic using constraints achieve better result than algorithm with heuristic using only profit of the objects. For all tested problems we achieve better results using heuristic with average expenses. Table 1 makes a comparison between ACO algorithms using different heuristics. It shows number of objects, number of constraints, the best value achieved by the ACO algorithms. With bold are better results.

5.1 Conclusion

In this paper we presented a version of ACO algorithms extended to handle knapsack problems. We have presented different types of heuristic informations. When we use profit of the objects and constraints we obtain better results than using only the profit of the objects. The results indicate the potential of the ACO

n×m	heuristic1	heuristic2	heuristic3
100×5	21640	22224	**23984**
100×5	21175	22542	**24143**
100×5	20477	22177	**23515**
100×5	21443	22390	**22874**
100×5	22169	23246	**23263**
100×5	20675	22427	**24523**
100×5	22068	24286	**25177**
100×5	21432	21771	**23204**
100×5	20501	22832	**23762**
100×5	22006	22893	**24208**
100×5	39510	41700	**42705**
100×5	40700	41804	**42445**
100×5	40329	40922	**41435**
100×5	41390	43856	**44911**
100×5	39253	40979	**42025**
100×5	40268	42130	**42671**
100×5	38138	40307	**41776**
100×5	41406	43304	**44559**
100×5	40029	42015	**43122**
100×5	41968	43135	**44364**
100×5	41968	43135	**44364**
100×5	58178	58660	**59798**
100×5	60031	60913	**61821**
100×5	58109	59117	**59694**
100×5	58592	59633	**60479**
100×5	59144	60050	**60954**
100×5	57218	58127	**58695**
100×5	59464	60193	**61406**
100×5	60180	60360	**61520**
100×5	57836	58659	**59121**
100×5	58186	59060	**59864**
Average	40117.166	41390.4	**42400.666**

Table 1. Experimental results using different heuristics, n is the number of objects and m is the number of constraints

approach for solving multiple knapsack problem. We can use similar technique for another constraint problems too. Our empirical results show that our ACO algorithm is currently among the best performing algorithms for this problem [1, 12].

Acknowledgements

Stefka Fidanova was supported by a Marie Curie Fellowship of the European Community program "Improving Human Research Potential and the Socio-Economic Knowledge Base" under contract number No HPMFCT-2000-00496. This work was supported by the "Metaheuristics Network", a Research Training Network funded by the Improving Human Potential programme of the CEC, grant HPRN-CT-1999-00106. The information provided in this paper is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

References

1. Chu, P.C., Beasley, J.E.: A genetic algorithm for the multiple knapsack problem. Journal of Heuristics **4** (1998) 63–86
2. 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
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. Dorigo, M., Di Caro, G.,Gambardella, L.M.: Ant algorithms for distributed discret optimization. Artificial Life **5** (1999) 137–172
5. Dorigo, M., Maniezzo, V., Colorni, A.: The ant system: Optimization by a colony of cooperating agents. IEEE Transaction on Systems, Man. and Cibernetics - Part B, **26** (1996) 29–41
6. Fogel, B.: Evolutionary computation: Toward a new philosophy of machine intelligence. IEEE Press, New York, (1994)
7. Gambardella, M.L., Taillard, E.D., Agazzi, G.: A multiple ant colony system for vehicle routing problems with time windows. In: Corne, D., Dorigo, M., Glover, F. (eds.): New Ideas in Optimization, McGraw-Hill, (1999) 63–76
8. Gambardella, L.M., Taillard, E.D., Dorigo, m: Ant colonies for the QAP. J. of Oper. Res. Soc. **50** (1999) 167–176
9. Glover, F.: Tabu search. ORSA J. of Comput. **1** (1989)
10. Kirkpatrick, S., Gelatt, C.D., Vechi, M.P.: Optimization by simulated annealing. Scienece **220** (1983) 671–680
11. Kochenberger, G., McCarl, G., Wymann, F.: A heuristic for general integer programming. Decision Sciences **5** (1974) 36–44
12. Leguizamon, G., Michalevich, Z.: A new version of the ant system for subset problems. In: Proceedings of Int. Conf. on Evolutionary Computations, Washington (1999)
13. Michel, R., Middendorf, M.: An island based ant system with lookahead for the shortest common super-sequence problem. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H-P. (eds.): Proceedings of the Fifth International Conference on Parallel Problem Solving from Nature, Lecture Notes in Computer Science Vol. 1498 (1998) 692–708