Evolutionary Algorithm for Multiple Knapsack Problem

Stefka Fidanova

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

Abstract. In the fields of evolutionary computation some interesting developments similar to ant colony optimization (ACO) algorithms have been proposed. The relation between ACO algorithms and evolutionary algorithms provides a structured way of handling constrained problems which the other approaches are lacking. The idea of ACO algorithms come from nature, the (artificial) ants use (artificial) pheromone to evaluate solutions. In this paper we present a new procedure to update the pheromone using additional reinforcement. Our modified ACO algorithm differs from the original ACO algorithms in several important aspects, whose usefulness we demonstrate by means of an experimental study. For experimental study we use the multiple knapsack problem (MKP), which is a real-life constraint optimization problem.

1 Introduction

There are many NP-hard combinatorial optimization problems (COP) for which it is impractical to find an optimal solution. For such problems the only reasonable way is to look for metaheuristic algorithms that quickly produce good, although not necessarily optimal, solutions. Many researchers have focused on a new class of algorithms called metaheuristics. Metaheuristics are general algorithmic frameworks that can be applied to several different optimization problems with few modifications. Examples of metaheuristics are simulated annealing [1, 10], evolutionary computation [6] and tabu search [9]. Metaheuristics 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.

In the last decade in the field of Evolutionary Computation some interesting developments quite similar to ACO algorithms have been proposed. They have in common the use of a probabilistic mechanism for recombination of individuals. This leads to algorithms where the population statistics are kept in probability vectors. In each iteration of the algorithm these probabilities are used to generate new solutions. The new solutions are then used to adapt the probability vector. The first approach of such a kind was given by the work of Syswerda [14], who replaced the usual two parent recombination operator by an operator called Bit-Simulated Crossover. Another approach called Population-Based Incremental Learning has been proposed by Baluja [15]. The relation between ACO algorithms and Evolutionary algorithms provides a structured way of handling constrained problems which the other approaches are lacking.

ACO algorithms were inspired by the observation of real ant colonies[2, 3, 5]. Ants are social insects, they 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 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. While walking from a food sources to the nest and vice-versa, ants deposit on the ground a substance called pheromone. Ants can smell pheromone, and when choosing their way, 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).

ACO is the recently developed, population-based approach which has been successfully applied to many NP-hard COP [4, 7, 8]. One of its main ideas is the indirect communication among the individuals of ant colony, that 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.

The main purpose of this paper is to use additional reinforcement of the pheromone to the unused movements and thus to effectively avoid premature convergence of the search. We use a particular implementation of ACO algorithm, known as ant colony system (ACS). For experimental study we use the multiple knapsack problem (MKP), as a real-life constraint problem. The empirical results show that the proposed ACO algorithm is currently among the best ACO algorithms for the MKP [12]. The remainder of this paper is structured as follows. Section 2 describes the modified ACO algorithm using additional reinforcement. Section 3 investigates the applicability of the modified ACO algorithm for MKP. Section 4 shows experimental results over some test problems. The paper ends with conclusions and some remarks.

2 The ACO Algorithm

The ACO algorithm make use of simple agents called ants which iteratively construct candidate solutions to a COP. The ants' solution construction is guided by pheromone trail and problem dependent heuristic information. The ACO algorithms can be applied to any COP 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 a candidate solution by starting with a partial solution and then iteratively adding new components to their partial solution 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 is completed, ants give feedback on their solutions by depositing pheromone on the components of their solutions. After that we reinforce the pheromone on the components of the best found solution. Typically, solution components which are part of best solution or are used by many ants will receive a higher amount of pheromone and hence will be more attractive by the ants in future iterations. To avoid the search getting stuck before the pheromone trails get reinforced all, pheromone trails are decreased.

In general, all ACO algorithms adopt specific algorithmic scheme as follows. After the initialization of the pheromone trails and control parameters, a main loop is repeated until a termination condition - which may be a certain number of iterations or a given CPU-time limit - is met. In the main loop, first ants construct feasible solutions, then the pheromone trails are updated. More precisely, partial solutions are seen as states: each ant moves from a state i to another state j to this 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 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 information indicating the a prior 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 a posterior indication of the desirability of that move.

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

$$p_{ij}^k = \begin{cases} \frac{\tau_{ij}\eta_{ij}}{\sum_{l \in allowed_k} \tau_{il}\eta_{il}} \text{ if } j \in allowed_k \\ 0 \qquad\qquad\qquad otherwise \end{cases}, \tag{1}$$

$allowed_k$ is the set of remaining feasible states. Thus the higher the value of the pheromone and the heuristic information, 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. While building a solution, the ants change the pheromone level of the elements of all solutions by applying local updating rule:

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

$0 < \rho < 1$ models evaporation. After all ants have completed their tours the pheromone level of the elements of the best solution will be update as follows.

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

where $\Delta\tau_{ij}$ depends to the quality of the best solution.

Stagnation situation 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 pheromone level 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. The stagnation situation should be avoided and this can be done by influencing the probabilities for choosing the next solution component, which depend directly on the pheromone trails. Our idea is to use additional reinforcement for unused movements. If some movements are not used in the current tour, additional pheromone reinforcement will be used as follows.

$$\tau_{ij} \leftarrow \tau_{ij} + q\tau_0, \tag{4}$$

where $q \geq 0$ is a parameter. After additional reinforcement, unused movements have great amount of pheromone than used movements that belong to poor solutions and less to the used movements that belong to the best solution. Thus the ants will be forced to choose new direction of search space without repeating the bad experience.

3 The ACO algorithm for MKP

The MKP is a NP-hard COP and belong to the family of the constraint problems and we use it as a test problem in our experiments. MKP has received wide attention from the operation research community, because its industrial applications. These applications include resource allocation in distributed systems, capital budgeting and cutting stock problems and etc. In addition, MKP can be seen as a general model for any kind of binary problems with positive coefficients [11]. We can formulate MKP as:

$$\begin{array}{rl} \max & \sum_{j=1}^{n} p_j x_j \\ \text{subject to} & \sum_{j=1}^{n} r_{ij} x_j \leq c_i \quad i = 1, \ldots, m \\ & x_j \in \{0,1\} \quad j = 1, \ldots, n. \end{array} \tag{5}$$

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.

MKP can be thought as a resource allocation problem, which is a real-life 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. The problem is to maximize the profit within a limited budget.

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 $\tilde{S}_k = \{i_1, i_2, \ldots, i_j\}$ being built by ant k, the probability $p_{i_p}^k$ of selecting i_p as the next item is given as follows:

$$p^k_{i_p} = \begin{cases} \frac{\tau_{i_j i_p} \eta_{i_p}(\tilde{S}_k)}{\sum_{i_b \in allowed_k} \tau_{i_j i_b} \eta_{i_b}(\tilde{S}_k)} & \text{if } i_p \in allowed_k \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

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

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

$$\eta_{ij} = \begin{cases} \frac{p_j^{d_1}}{s_j^{d_2}} & \text{if } \; s_j \neq 0 \\ p_j^{d_1} & \text{if } \; s_j = 0 \end{cases}. \tag{7}$$

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

While building a solution, the ants visit the arcs and change their pheromone level. After all ants have completed their tours, global updating is performed. In the case of MKP $\Delta\tau_{i_j i_p}$ is as follows:

$$\Delta\tau_{i_j i_p} = \begin{cases} f_{gb} \text{ if } (i_j, i_p) \in \text{global best tour} \\ 0 \quad \text{otherwise} \end{cases}, \tag{8}$$

f_{gb} is the best value of objective function from the beginning of the trial.

4 Experimental Results

This section reports the experimental results obtained by applying the modified ACO algorithm using additional reinforcement to the MKP. The modified algorithm have been tested on a set of large MKP from "OR-Library" (*http://mscmga.ms.ic.ac.uk/jeb/orlib*). It was coded in *C* language and run on a *Pentium III* 900MHz. The best results were found when $\rho = 0.9$, $d_1, d_2 \in [1, 9]$ and $q \in [0, 600]$. For other parameters, $\tau_0 = 1$ and n ants. In the beginning of every iteration each node is occupied by an ant. The same result have been achieved, for the same data and different runs, when using the same parameters and the same number of iterations. The maximum number of cycles was set to 500 for all experiments.

The modified ACO algorithm have been compared with the MAX-MIN algorithm [13], which is the best ACO algorithm. The idea of MAX-MIN algorithm is to use τ_{min} and τ_{max}, a lower and an upper limit of the pheromone respectively. For τ_{max} is used the asymptotic upper bound of the pheromone. For lower limit they use $\tau_{min} = \tau_{max}/2n$, where n is the number of objects. For both algorithms

Table 1. The results of ACS algorithm with additional reinforcement and with upper and lower limit of the pheromone, n is the number of objects, m is the number of constraints

n×m	add. reinforc.	MAX-MIN
100×5	23984	23984
100×5	**24145**	24143
100×5	**23523**	23515
100×5	22874	22874
100×5	**23751**	23263
100×5	**24601**	24523
100×5	**25293**	24769
100×5	23204	23204
100×5	23762	23762
100×5	**24255**	24114
100×5	42705	42705
100×5	42445	42445
100×5	41581	41581
100×5	44911	44911
100×5	42025	42025
100×5	**42671**	42609
100×5	41776	41776
100×5	**44671**	44559
100×5	43122	43122
100×5	**44471**	44364
100×5	59798	59798
100×5	**61825**	61637
100×5	59694	59694
100×5	60479	60479
100×5	**61016**	60954
100×5	**58790**	58695
100×5	**61429**	61406
100×5	61520	61520
100×5	**59290**	59121
100×5	**59896**	59864

n×m	add. reinforc.	MAX-MIN
250×5	58721	58721
250×5	**61161**	60522
250×5	**61671**	61208
250×5	58317	58317
250×5	58199	58199
250×5	**59819**	59762
250×5	**60191**	60125
250×5	60707	60707
250×5	**61576**	61431
250×5	**58323**	58188
250×5	108731	108731
250×5	**109049**	109022
250×5	**108356**	108238
250×5	108766	108766
250×5	**110339**	110159
250×5	109243	109243
250×5	108464	108464
250×5	**107842**	107435
250×5	**109712**	109124
250×5	**106002**	105964
250×5	149246	149246
250×5	155777	155777
250×5	**149104**	148599
250×5	**151896**	151889
250×5	**149931**	149420
250×5	**149789**	149652
250×5	148123	148123
250×5	149589	149589
250×5	154736	154736
250×5	154600	154600

we use same parameters and same number of iterations. In all cases, the achieved result of modified ACO algorithm is equal or better than the result of MAX-MIN algorithm.

Table 1 shows a comparison between our modified ACO algorithm and MAX-MIN algorithm. This table represents number of objects, number of constraints, the best value achieved by both ACO algorithms. Bold numbers indicate the best joined results. From the comparison, it is clear that the modified ACO algorithm performs better than MAX-MIN ACO algorithm. The above results for MKP suggest the effectiveness of the proposed ACO algorithm with additional reinforcement.

5 Conclusion

In this paper we apply the modified ACO algorithm to MKP. This problem received wide attention from the operation research community because its a real-life problem. These applications include economical problems like resource allocation, capital budgeting, cutting stock, some biological problems like proteins and DNA comparison. Recent research has strongly focused on improving the performance of ACO algorithms. In this paper, we have presented ACO algorithm with additional reinforcement of the unused movements. The proposed algorithm aims to exploit a search space, which have not been exploited yet and to avoid premature convergence of the ants' search. Comparing with the MAX-MIN ACO algorithm, the achieved results of our algorithm prove the improved performance over the MAX-MIN ACO algorithm. When usual genetic algorithm is applied for solving constrained problems, sometime it produces unfeasible solutions and it need techniques like local search to reach feasible solution. Comparing with standard genetic algorithm our modified ACO algorithm produces only feasible solutions.

Acknowledgments

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 program 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 discrete 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 Cybernetics - 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. Science **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. Stützle, T., Hoos H.H.: MAX-MIN ant system. In: Dorigo, M., Stützle, T., Di Caro, G. (eds.): Future Generation Computer System Vol. 16 (2000) 889–914
14. Syswerda, G.: Simulated Crossover in Genetic Algorithms. In:Whitley, L.D. (eds.): Proc. of the second workshop on Foundations of Genetic Algorithms, Sam Mateo California, Morgan Kaufmann Publishers (1993) 239-255
15. Baluja, S., Caruna, R.: Removing Genetics from the Standard Genetic Algorithm. In:Priedits, A., Russel, S. (eds.): The international Conference on Machine Learning, Sam Mateo California, Morgan Kaufmann Publishers (1995) 38–46