

# An ant colony optimization approach for the multidimensional knapsack problem

Liangjun Ke · Zuren Feng · Zhigang Ren · Xiaoliang Wei

Received: 12 June 2007 / Revised: 14 April 2008 / Accepted: 3 June 2008 /  
Published online: 20 June 2008  
© Springer Science+Business Media, LLC 2008

**Abstract** Ant colony optimization is a metaheuristic that has been applied to a variety of combinatorial optimization problems. In this paper, an ant colony optimization approach is proposed to deal with the multidimensional knapsack problem. It is an extension of Max Min Ant System which imposes lower and upper trail limits on pheromone values to avoid stagnation. In order to choose the lower trail limit, we provide a new method which takes into account the influence of heuristic information. Furthermore, a local search procedure is proposed to improve the solutions constructed by ants. Computational experiments on benchmark problems are carried out. The results show that the proposed algorithm can compete efficiently with other promising approaches to the problem.

**Keywords** Ant colony optimization · Metaheuristic · Multidimensional knapsack problem

## 1 Introduction

The multidimensional knapsack problem (MKP) consists in finding a subset of an original set of objects such that the total profit of the selected objects is maximized while a set of resource constraints are satisfied, where the terms *profit* and *resource* should be considered in their most general sense. The MKP has been recognized as a model of many real applications such as cutting stock problems (Gilmore and Gomory 1966), project selection and cargo loading (Shih 1979), allocating processors and databases in a distributed computer system (Gavish and Pirkul 1982).

---

L. Ke (✉) · Z. Feng · Z. Ren · X. Wei  
State Key Laboratory for Manufacturing Systems Engineering, Xi'an Jiaotong University, Xi'an, China  
e-mail: [kelj163@163.com](mailto:kelj163@163.com)

It is well-known that the MKP is an NP-hard problem. Exact algorithms, such as branch-and-bound algorithms (e.g. Shih 1979), dynamic programming based algorithms (e.g. Gilmore and Gomory 1966), can solve only instances of very limited size in an acceptable computation time. The intractability of the MKP motivated many researchers to focus on the development of heuristic methods (e.g. Battiti and Tecchiolli 1995; Chu and Beasley 1998; Freville and Plateau 1994; Glover and Kochenberger 1996; Hanafi and Freville 1998).

In the last decade, Ant Colony Optimization (ACO) has been successfully applied to many hard combinatorial optimization problems (Dorigo et al. 1999; Dorigo and Stützle 2004). ACO was inspired by the foraging behavior of real ants (Dorigo et al. 1996). When solving a problem, artificial ants probabilistically construct solutions using heuristic information and pheromone trails. The pheromone trails are updated according to the quality of the solutions constructed by the ants. The success of ACO has motivated many attempts to solve the MKP with it (Alaya et al. 2004; Fidanova 2002; Leguizamon and Michalewicz 1999).

In this paper, we develop an algorithm based on ACO for the MKP. Our algorithm extends Max Min Ant System (MMAS) (Stützle and Hoos 2000) which is one of the most successful ACO algorithms (Dorigo and Blum 2005). In MMAS, upper and lower trail limits are imposed on pheromone trails to avoid stagnation where all ants construct the same solution over and over again, such that no better solution can be found anymore (Stützle and Hoos 2000). Obviously, appropriate limits are important to the performance of MMAS. The upper limit is usually set to an estimate of the asymptotically maximum pheromone trail value, whereas the lower trail limit is relatively difficult to choose. Stützle and Hoos (2000) suggested to set the lower trail limit based on the probability of constructing the best solution found when all the pheromone values have converged to either the upper or lower trail limit (Levine and Ducatelle 2004). This method assumes that the influence of heuristic information can be neglected. However, when ACO is applied to the MKP, heuristic information is important (Alaya et al. 2004; Fidanova 2002; Leguizamon and Michalewicz 1999). In order to overcome this limitation, our algorithm provides a new method to choose the lower trail limit. The basic idea is to change the lower trail limit when the difference between the solutions constructed by ants and the best solution which was reinforced at the previous cycle is small. Since the relative difference between the upper and lower trail limits is dynamic in this method, it is called *dynamic method*. Correspondingly, the proposed algorithm is called dynamic MMAS (DMMAS). In addition, a local search procedure is proposed to improve the solutions generated by ants.

The remainder of this paper is organized as follows. In Sect. 2, the MKP is defined. Then we give a general introduction to ACO and a survey on previous ACO-based algorithms for the MKP. In Sect. 3, the proposed algorithm is presented. Section 4 describes the method of choosing the lower trail limit. Section 5 presents the proposed local search procedure. The computational results on benchmark problems are given in Sect. 6. The conclusion is given in Sect. 7.

## 2 Preliminaries

### 2.1 The multidimensional knapsack problem

The MKP can be formulated as follows (Chu and Beasley 1998):

$$\text{maximise} \quad \sum_{j=1}^n p_j x_j \quad (1)$$

$$\text{subject to} \quad \sum_{j=1}^n r_{ij} x_j \leq b_i, \quad i = 1, \dots, m, \quad (2)$$

$$x_j \in \{0, 1\}, \quad j = 1, \dots, n. \quad (3)$$

There are  $m$  resource constraints in this problem, therefore the MKP is also called the  $m$ -dimensional knapsack problem. Let  $I = \{1, \dots, m\}$  and  $J = \{1, \dots, n\}$  with  $b_i \geq 0$  for all  $i \in I$ , and  $r_{ij} \geq 0$  for all  $i \in I, j \in J$ . A well-stated MKP assumes that  $p_j > 0$  and  $r_{ij} \leq b_i < \sum_{j=1}^n r_{ij}$ . It is clear that each solution  $x = (x_1, \dots, x_n)$  is an  $n$ -dimensional vector. If  $x_j$  is equal to 1, it means that object  $j$  is selected, otherwise, object  $j$  is unselected.

### 2.2 ACO for the MKP

ACO is a class of model-based metaheuristics. It uses a colony of ants, which are guided by pheromone trails and heuristic information, to construct solutions iteratively for a problem. To solve a static combinatorial optimization problem via ACO, the main procedure is described as follows: at each cycle, every ant constructs a solution and then pheromone trails are updated. The algorithm stops iterating when a termination condition is met. Generally, the termination condition may be a maximum number of solution constructions or a given time limit.

Since the progenitor of ACO, called Ant System (AS), was introduced (Dorigo et al. 1996), many improvements have been proposed to make ACO algorithms very effective. One of the most outstanding ACO variants is MMAS (Stützle and Hoos 2000). The prominent characteristic of MMAS is that upper and lower trail limits are imposed on pheromone trails to avoid stagnation. Moreover, the pheromone values are initialized to the upper trail limit. This gives ants higher exploration ability in the early cycles. In addition, MMAS uses only one ant to update pheromone trails. The ant may be the one which constructed the global-best solution or the iteration-best one. In this way, it can make ants exploit those best solutions. MMAS deliberately schedules the global-best solution and the iteration-best solution for pheromone update in order to balance exploitation and exploration. Strong exploitation may be obtained by using the global-best solution, while exploration may be intensified by using the iteration-best solution (Levine and Ducatelle 2004).

In recent years, several ACO-based algorithms have been proposed to deal with the MKP (Alaya et al. 2004; Fidanova 2002; Leguizamón and Michalewicz 1999). Their differences mainly lie in the ways of laying pheromone trails and the definitions of

heuristic information. Let  $S = \{o_1, o_2, \dots, o_{|S|}\}$  be the set of the selected objects, where  $|S|$  is the cardinality of  $S$ . Three ways of laying pheromone trails have been investigated:

- (1) The first way is to lay pheromone trails on each object of  $S$  (Leguizamón and Michalewicz 1999). In this way, each object is associated with an amount of pheromone which represents the preference of the object. It tries to increase the desirability of each object of  $S$  in such a way that, when constructing a new solution, these objects will be more likely to be chosen;
- (2) The second one is to lay pheromone trails on each pair  $(o_u, o_{u+1})$  of successively selected objects of  $S$  (Fidanova 2002). The idea is to increase the desirability of choosing object  $o_{u+1}$  when the last selected object is  $o_u$ . In this way, a pheromone trail represents the preference of selecting a certain object after another one;
- (3) The third one is to lay pheromone trails on all pairs of different objects of  $S$  (Alaya et al. 2004). The idea is to increase the desirability of choosing simultaneously two objects of  $S$  so that, when constructing a new solution, the objects of  $S$  will be more likely to be selected if some objects of  $S$  have been selected.

With respect to heuristic information, Alaya et al. (2004) and Leguizamón and Michalewicz (1999) applied a kind of dynamic heuristic information. Let  $S_k$  be the set of the selected objects at the  $k$ th construction step. For each candidate object  $j$ , the heuristic information  $\eta_{S_k}(j)$  is given as follows:

$$\eta_{S_k}(j) = p_j / \left( \sum_{i=1}^m r_{ij} / d_{S_k}(i) \right) \quad (4)$$

where  $d_{S_k}(i) = b_i - \sum_{g \in S_k} r_{ig}$ . Since  $S_k$  will be changed from step to step, the heuristic information is dynamic. Fidanova (2002) used three kinds of static heuristic information. For each object  $j$ , three kinds of heuristic information  $\eta_j^1$ ,  $\eta_j^2$  and  $\eta_j^3$  are given as follows:

$$\eta_j^1 = p_j^{d_1} \quad (5)$$

where  $d_1$  is a parameter.

$$\eta_j^2 = \begin{cases} p_j^{d_1} / s_j^{d_2} & \text{if } s_j \neq 0 \\ p_j^{d_1} & \text{if } s_j = 0 \end{cases} \quad (6)$$

where  $s_j = \max_i(r_{ij})$ ,  $d_1$  and  $d_2$  are two parameters.

$$\eta_j^3 = \begin{cases} p_j^{d_1} / s_j^{d_2} & \text{if } s_j \neq 0 \\ p_j^{d_1} & \text{if } s_j = 0 \end{cases} \quad (7)$$

where  $s_j = \sum_{i=1}^m r_{ij}$ ,  $d_1$  and  $d_2$  are two parameters.

### 3 Description of the proposed algorithm

DMMAS, the proposed algorithm, follows the standard scheme of ACO, whereas several components are dependent on the characteristics of the MKP. In the following, we will explain how to define pheromone trails and heuristic information, and then describe the details about constructing a solution. Finally, we will present how to update pheromone trails.

#### 3.1 Pheromone trails and heuristic information

It is known that the MKP is a typical subset problem which involves selecting an optimal feasible subset of an original set of objects. Some other examples of ACO algorithms for subset problems are the constraint satisfaction problem (Solnon 2002), the maximum clique problem (Solnon and Fenet 2006) and the edge-weighted  $k$ -cardinality tree problem (Blum and Blesa 2005). In contrast to the ordering problems, such as traveling salesman problem (TSP), subset problems focus on selecting rather than ordering (Solnon and Bridge 2006). For these problems, one may consider two different pheromone structures: a pheromone structure that associates a pheromone trail with every object, or a pheromone structure that associates a pheromone trail with every pair of objects. In DMMAS, the first way of laying pheromone trails is used, i.e., a pheromone trail  $\tau(j)$  is associated with each object  $j$ .

Apart from pheromone trails, heuristic information is another important factor for solution construction. The heuristic information we used is on the basis of the pseudo-utility ratios (Chu and Beasley 1998), which are defined by:

$$\forall j \in J, \quad \eta(j) = \frac{p_j}{\sum_{i=1}^m w_i r_{ij}} \quad (8)$$

where  $w_i$  is the shadow price of the  $i$ th constraint in the linear programming relaxation of the original MKP, which has been adopted by the genetic algorithm in Chu and Beasley (1998). The denominator of (8) represents the tightness of object  $j$ . As indicated in (8), an object with a higher profit and lower tightness will be more desirable for selection.

#### 3.2 Constructing a solution

When constructing a solution, an ant starts with an  $n$ -dimensional vector  $x$ . This vector corresponds to a solution, and each element of this vector is initially set to zero. At the  $k$ th construction step ( $k \geq 1$ ), an ant scratches an object following the transition probability:

$$P(c_k = j | \tau) = \begin{cases} \frac{\tau(j)^\alpha \eta(j)^\beta}{\sum_{u \in U_k} \tau(u)^\alpha \eta(u)^\beta} & \text{if } j \in U_k \\ 0 & \text{otherwise} \end{cases} \quad (9)$$

where  $U_k$  is the set of objects which have not yet scratched and satisfy all constraints, that is,  $U_k = \{o \in \{1, \dots, n\} | (x_o = 0) \wedge (r_{io} \leq b_i - \sum_{j=1}^n r_{ij} x_j, i = 1, \dots, m)\}$ .  $\alpha$  and  $\beta$  ( $\alpha, \beta \geq 0$ ) are two parameters which control the relative importance of pheromone

trails and heuristic information. According to this transition probability, ants prefer selecting those more desirable objects with a higher amount of pheromone. Suppose that object  $i$  is selected, then the  $i$ th element of the vector will be changed into 1. The construction process stops when  $U_k$  is empty.

### 3.3 Pheromone update

Once each ant has constructed a solution, only the best ant is used to update pheromone trails. More formally, pheromone trails are updated as follows:

$$\forall u \in J, \quad \tau(u)^{l+1} = \rho \tau(u)^l + \Delta \tau(u), \quad (10)$$

$$\text{If } \tau(u)^{l+1} < \tau_{\min}, \quad \text{then } \tau(u)^{l+1} = \tau_{\min} \quad (11)$$

$$\text{If } \tau(u)^{l+1} > \tau_{\max}, \quad \text{then } \tau(u)^{l+1} = \tau_{\max} \quad (12)$$

where  $\tau(u)^l$  is the pheromone value of object  $u$  at cycle  $l$ ,  $\rho$  is the pheromone persistence ( $1 - \rho$  is the pheromone evaporation rate). Let  $s^{best}$  be the solution constructed by the chosen ant. It may be the global-best solution  $s^{gb}$  or the iteration-best one  $s^{ib}$ . If element  $u$  of  $s^{best}$  is equal to 1, then  $\Delta \tau(u)$  is equal to  $g(s^{best})$ , where  $g(x) = 1 / \sum_{j=1}^n p_j (1 - x_j)$ . Otherwise,  $\Delta \tau(u)$  is 0.  $\tau_{\max}$  and  $\tau_{\min}$  are the upper and lower trail limits respectively. In this way, those objects selected by the chosen ant will receive more pheromone and therefore will be more likely selected in future cycles. The upper trail limit  $\tau_{\max}$  is initialized to an arbitrarily high value. After a global-best solution  $s^{gb}$  is constructed,  $\tau_{\max}$  is set to  $g(s^{gb}) / (1 - \rho)$ . Note that  $\tau_{\max}$  may be different from cycle to cycle. In the next section, we will discuss how to choose the lower trail limit in detail.

## 4 Selecting the lower trail limit

The lower trail limit is crucial to the performance of the proposed algorithm. When the relative difference between the upper and lower trail limits is very small, the search behavior of ants may be too diversified. Whereas very striking relative difference may lead to stagnation, since it is often caused by a much higher pheromone level on some solution components than on others (Stützle and Hoos 2000). Hence a suitable lower trail limit should be chosen in such a way that a good tradeoff between diversification and intensification can be obtained.

### 4.1 Analysis of Stützle and Hoos' method

In Stützle and Hoos (2000), the lower trail limit  $\tau_{\min}$  is set as

$$\tau_{\min} = \varepsilon \tau_{\max} \quad (13)$$

where  $\varepsilon = (1 - \sqrt[n]{P_{best}}) / ((avg - 1) \sqrt[n]{P_{best}})$ ,  $avg$  is equal to  $n/2$ ,  $n$  is the total number of objects, the parameter  $P_{best}$  is the probability of constructing the best solution found when all the pheromone values have converged to either  $\tau_{\max}$  or  $\tau_{\min}$ . For simplicity, we denote this method as *SH method*.

When the SH method is applied to select the lower trail limit, it is important to choose an appropriate  $P_{best}$ . In order to study the influence of  $P_{best}$  on the capability of ants to explore the search space, two measures are used:

- (1) Similarity ratio: This measure is commonly used to measure diversification, and it has been applied to evolutionary algorithms (Morrison and De Jong 2001). The following similarity ratio is given by Solnon and Fenet (2006):

$$\frac{\sum_{j=1}^n (\sum_{i=1}^{n_a} s_j^i \cdot (\sum_{i=1}^{n_a} s_j^i - 1))}{(n_a - 1) \cdot \sum_{j=1}^n \sum_{i=1}^{n_a} s_j^i} \quad (14)$$

where  $s_j^i$  is the  $j$ th element of solution  $s^i$  which is constructed by the  $i$ th ant,  $n_a$  is the number of ants. If all solutions are the same, the ratio is equal to one. The ratio is equal to zero if all ants select entirely different objects.

- (2) Re-sampling ratio: It is used to measure how effective an algorithm is in sampling the search space (Solnon and Fenet 2006). Let *DiffNum* be the number of unique solutions generated over a whole run and *TotalNum* be the number of all generated solutions, then the re-sampling ratio is defined as  $(TotalNum - DiffNum) / TotalNum$ . Values close to 0 indicate an effective search, that is, few duplicate solutions are generated, while values close to 1 mean that the search is in stagnant condition, i.e., few new solutions are generated.

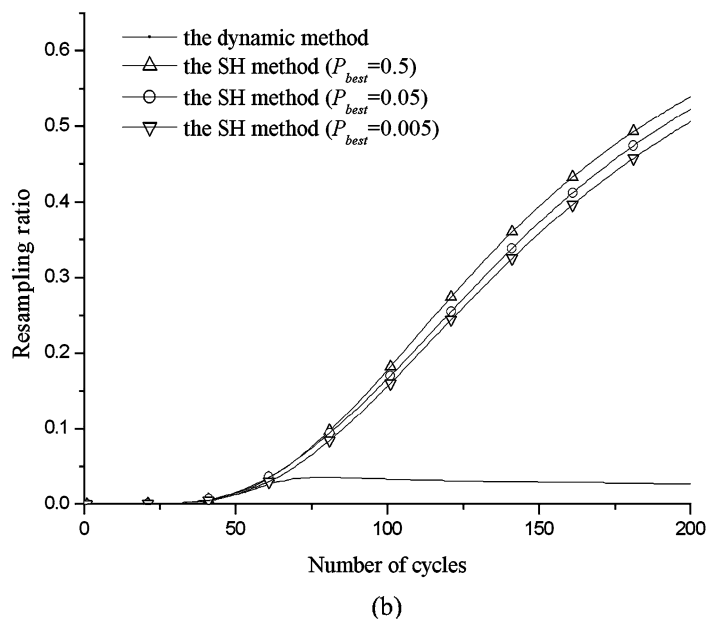
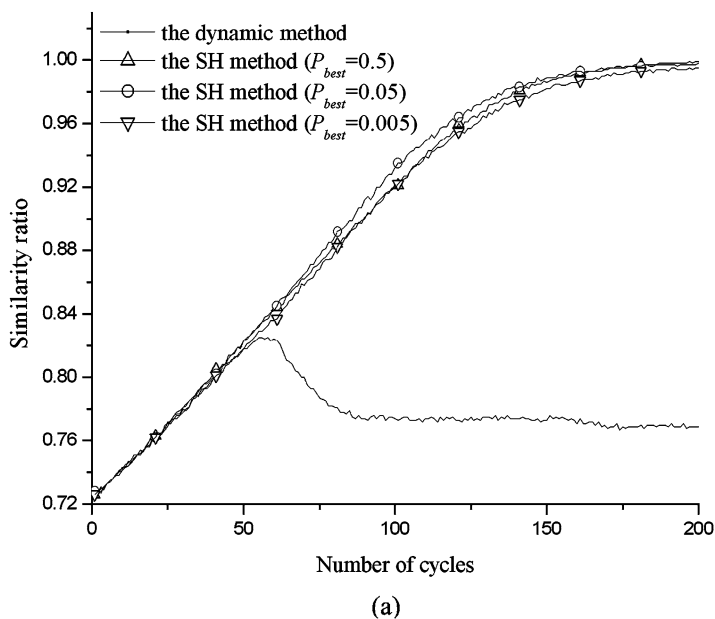
Figure 1 plots the evolution of the similarity ratio and the re-sampling ratio obtained at each cycle where instance 10.100.00 is used as an example. Even when  $P_{best}$  is set to 0.005, the similarity ratio and the re-sampling ratio increase very quickly and reach 0.98 and 0.36 at cycle 150 respectively. That is, ants focus on a very small region of the search space and 36 percent of the solutions generated had already been generated before. In this case, the search has to be diversified. It is clear that heuristic information plays an important role in solution construction, while (13) takes no account of heuristic information. This makes it uneasy to choose  $P_{best}$ . Especially, when this method is applied to a broad range of instances, it will be very inconvenient to choose this important parameter.

## 4.2 The dynamic method

In order to choose a lower trail limit, we first describe the relative differences on pheromone trails. Let  $T_1 = \{\tau(u)^\alpha \eta(u)^\beta | s^{best}(u) = 1, u \in J\}$  and  $T_2 = \{\tau(u)^\alpha \eta(u)^\beta | s^{best}(u) = 0, u \in J\}$ . Suppose the minimal value of  $T_1$  is  $\tau_1$ , the maximal value of  $T_2$  is  $\tau_2$ . Once pheromone trails are updated,  $\tau_1/\tau_2$  increases. Specially, when  $\tau_1/\tau_2 \rightarrow +\infty$ , the probability of constructing  $s^{best}$  at the next cycle will tend to 1, that is, the hamming distance between a random solution  $s$  constructed at the next cycle and  $s^{best}$  will tend to zero. Formally, we have

**Property 1** Let the expected value of the hamming distance between  $s$  and  $s^{best}$  be  $E(d(s, s^{best}))$ , then  $E(d(s, s^{best})) \rightarrow 0$  if and only if  $\tau_1/\tau_2 \rightarrow +\infty$ .

*Proof* Put  $O = \{j \in J | s^{best}(j) = 1\}$ ,  $K = |O|$ . Suppose that when constructing the solution  $s$ , the  $k$ th selected object is  $a_k$  ( $1 \leq k \leq K$ ). Note that  $E(d(s, s^{best})) \rightarrow 0$



**Fig. 1** Evolution of (a) the similarity ratio and (b) the re-sampling ratio for instance 10.100.00 (average over 50 runs). The default value of each parameter was:  $n_a = 50$ ,  $\alpha = 1$ ,  $\beta = 20$ ,  $\rho = 0.95$ ,  $\gamma = 8$ ,  $\lambda = 2$ ,  $P_{best} = 0.9$

if and only if  $P(a_k \in O) \rightarrow 1 (1 \leq k \leq K)$ . Since  $s$  is constructed following the way described in Sect. 3.2, the proof is obvious.  $\square$

According to the definition of  $T_1$  and  $T_2$ , it can be seen that  $E(d(s, s^{best}))$  takes into account heuristic information. According to classical statistical theory (Devore 2000),  $E(d(s, s^{best}))$  can be estimated as follows:

$$E(d(s, s^{best})) \approx avg_d \quad (15)$$

$$avg_d = \frac{\sum_{i=1}^{n_a} d(s^i, s^{best})}{n_a} \quad (16)$$

where  $avg_d$  is called *average distance*,  $n_a$  is the number of ants and  $s^i$  is the solution constructed by ant  $i$ . It is well-known that  $avg_d$  is an unbiased estimator of  $E(d(s, s^{best}))$ . Note that  $s^{best}$  is the solution that has been rewarded at the previous cycle. Since  $avg_d$  can be computed incrementally during the solution construction step, no significant extra computation cost is needed.

Our method of choosing the lower trail limit is given as follows:

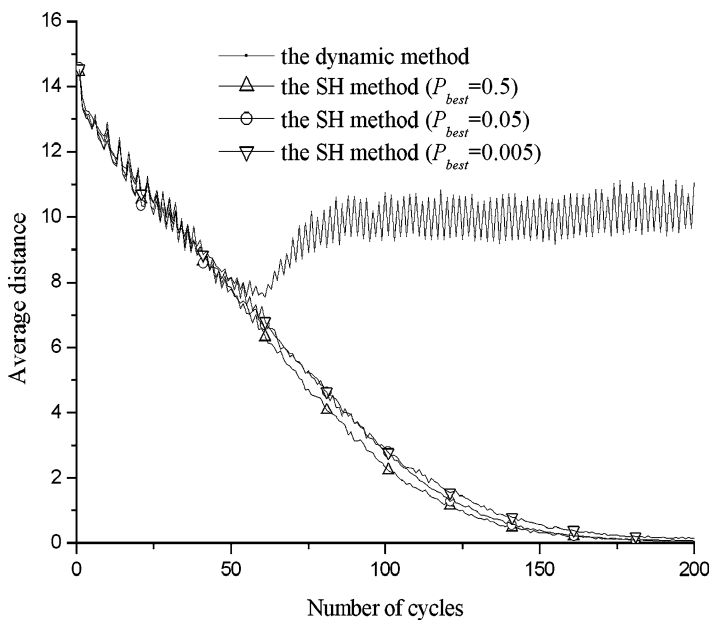
When a new  $s^{gb}$  is found,  $\tau_{\min}$  is initialized to a very small value. This can be realized by setting  $\tau_{\min}$  to  $\varepsilon \tau_{\max}$  where  $P_{best}$  is set to a very large number (e.g.  $P_{best} \geq 0.5$ ). Since then, if the average distance  $avg_d$  is very small, it is possible that the relative difference between  $\tau_1$  and  $\tau_2$  is extremely large. It implies that the relative difference between the lower and upper trail limits is too large. In order to decrease the relative difference, one possible way is to increase  $\tau_{\min}$ . That is,

$$\text{If } avg_d < \gamma, \text{ then } \tau_{\min} := \lambda \tau_{\min} \quad (17)$$

where  $\gamma$  is a positive number,  $\lambda (\lambda > 1)$  is a parameter.

Unlike the SH method, our method provides a new way to balance diversification and intensification by dynamically updating the lower trail limit when the average distance is small. In this method,  $\tau_{\min}$  is initialized to a very small value so that  $\tau_{\max}/\tau_{\min}$  is very large at first. Once the premise of (17) is satisfied,  $\tau_{\max}/\tau_{\min}$  scales down. Moreover,  $\tau_{\max}/\tau_{\min}$  can be large enough in order to favor pheromone guidance. The threshold  $\gamma$  is used to determine whether the relative difference on the trail limits is small or not. With a very large value, the relative difference on the trail limits will be very small after a few cycles. While the relative difference on the trail limits will be large if  $\gamma$  is set to a small value. Specially, when  $\gamma$  is equal to 0,  $\tau_{\min}$  will be changeless at all.

We also analyzed the dynamic method based on similarity ratio and re-sampling ratio. As shown in Fig. 1, the similarity ratios and re-sampling ratios of the dynamic method are smaller than those of the SH method. This means that ants have better ability to explore new solutions. Figure 2 plots the evolution of the average distance of the dynamic method and the SH method. With respect to the dynamic method, the average distances are larger than or close to  $\gamma$  (where  $\gamma = 8$ ) during the run of DMMAS. As to SH method, at the beginning of the search, the average distances are large. After cycle 150, the average distances are very small (less than 1). This confirms that ants only search in a very small region of the search space.



**Fig. 2** Evolution of the average distance for instance 10.100.00 (average over 50 runs). The default value of each parameter was:  $n_a = 50$ ,  $\alpha = 1$ ,  $\beta = 20$ ,  $\rho = 0.95$ ,  $\gamma = 8$ ,  $\lambda = 2$ ,  $P_{best} = 0.9$

In Stützle and Hoos (2000), pheromone reinitializing is applied to avoid stagnation. In practice, one can reinitialize pheromone trails when the average difference is less than  $\gamma$  or when no better solution could be found for  $N_{ni}$  cycles. We measured the significance of the obtained results using the Wilcoxon rank sum test with a significance level of 0.05. The statistical results show that no significant improvement can be obtained by reinitializing pheromone. This may be due to the adoption of short run test. Of course, when long time is allowed, this restarting mechanism may be useful for MMAS as well as DMMAS. In fact, it is profitable to execute several short runs of MMAS than running a single long run in the same whole computation time (Stützle 1998).

## 5 Improving DMMAS with local search

It is known that the coupling of ACO and local search can effectively improve the performance of the pure ACO. In fact, ACO performs a rather coarse-grained search, and then the solutions constructed can be locally optimized by an adequate local search procedure (Dorigo and Stützle 2002).

The local search procedure we proposed is a deterministic procedure. The main idea is to replace each selected object with every two unselected objects, until the replacement with the highest gain is determined. Consequently, at most one selected object is replaced by at most two unselected objects.

The idea of adding an improvement phase to ACO has been widely exploited before (e.g. Blum and Blesa 2005; Levine and Ducatelle 2004; Maniezzo and Colnari

1999; Solnon 2002). The drawback of this approach is that using local search results in a very slow algorithm. We can speed up the local search procedure by sorting the objects in decreasing order of profit. This can be explained as follows: when to determine the second unselected object, if the current gain is less than the highest gain obtained so far, then it is unnecessary to try the remaining objects.

## 6 Experimental study

The algorithm proposed in this paper was coded in C++ and run on a 2.8 GHZ Pentium IV processor. The benchmark instances for evaluating DMMAS and local search procedure are taken from Chu and Beasley (1998).<sup>1</sup> Each instance is identified by notation  $m.n.z$ , where  $m$  and  $n$  are the number of constraints and objects respectively,  $z$  is a label that differentiates between instances with the same number constraints and objects. As suggested in Stützle and Hoos (2000), a dynamic mixed strategy which increases the frequency of using  $s_{gb}$  was adopted for pheromone update: Let  $f^{gb}$  indicate that every  $f^{gb}$  cycles  $s_{gb}$  is used to update pheromone trails. For the first 9 cycles, we set  $f^{gb} = 3$ , for cycles 10 to 24 we set  $f^{gb} = 2$ , and from cycle 25 on  $f^{gb} = 1$ . The initial value of pheromone trails was set to some arbitrarily high value. 50 independent runs were carried out on each instance.

### 6.1 Parameters setting

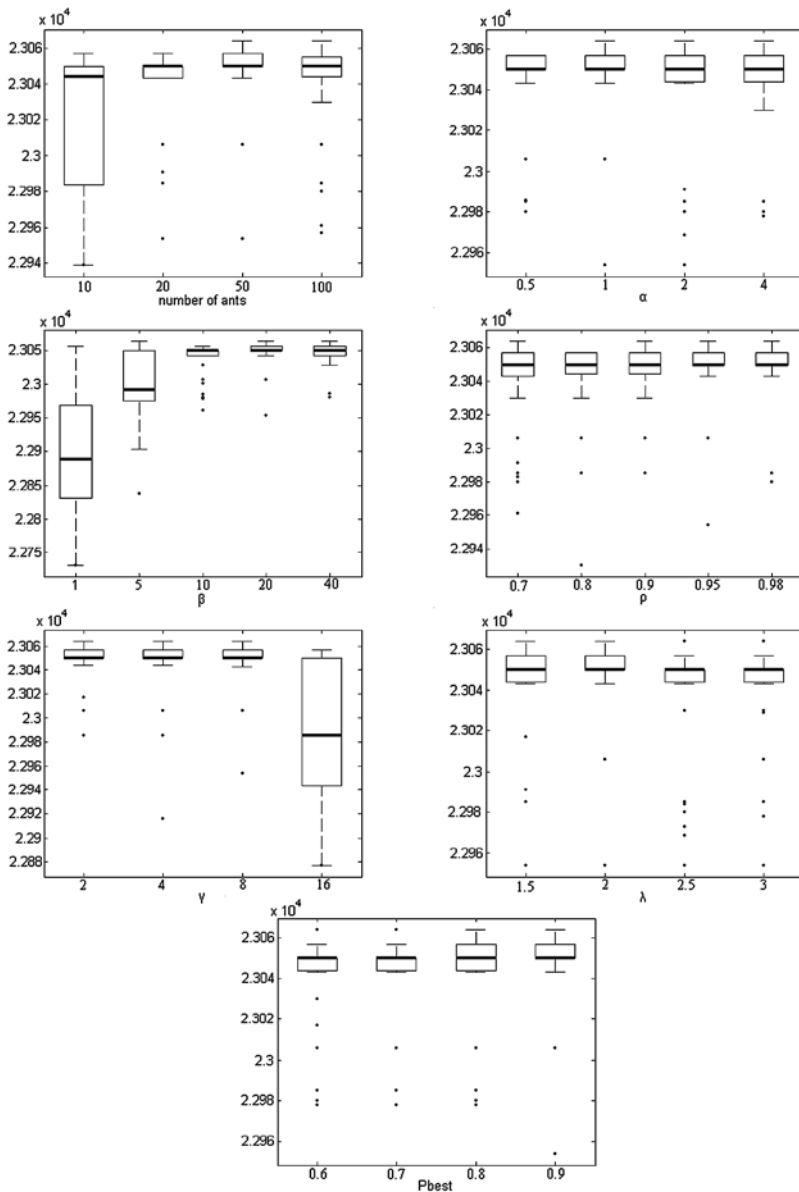
Like Leguizamón and Michalewicz (1999), DMMAS stops iterating when ants have constructed 10000 solutions. We have studied the influence of seven parameters based on experimental results. The default value of each parameter was  $n_a = 50$ ,  $\alpha = 1$ ,  $\beta = 20$ ,  $\rho = 0.95$ ,  $\gamma = 8$ ,  $\lambda = 2$ ,  $P_{best} = 0.9$ . In each experiment only one of the parameter values was changed. The values tested were:  $n_a \in \{10, 20, 50, 100\}$ ,  $\alpha \in \{0.5, 1, 2, 4\}$ ,  $\beta \in \{1, 5, 10, 20, 40\}$ ,  $\rho \in \{0.7, 0.8, 0.9, 0.95, 0.98\}$ ,  $\gamma \in \{2, 4, 8, 16\}$ ,  $\lambda \in \{1.5, 2, 2.5, 3\}$ ,  $P_{best} \in \{0.6, 0.7, 0.8, 0.9\}$ . We used problem 10.100.00 as an example. Figure 3 summarizes the results. For each parameter value, there is a sample of 50 profits. The data is represented in box plot format.

The number of ants was set to 50. With smaller values, solution quality is often decreased. As  $n_a$  increases, more candidate solutions can be constructed at each cycle and the best profit obtained at each run is usually better. However, since the maximum number of solution construction is fixed, solution quality is decreased when  $n_a$  is set to a very large number. This may be due to little cooperation between ants.

It can be seen that solution quality is more sensitive to  $\beta$  than to  $\alpha$ . DMMAS works better when a relatively high value is chosen for  $\beta$ . However, with a very large value (e.g., 40), the ants aggressively select these objects with higher heuristic information, and the efficiency of the algorithm is reduced. When  $\beta = 20$ , a good value of  $\alpha$  is 1. As  $\alpha$  increases or decreases, the performance of DMMAS decreases.

The pheromone persistence  $\rho$  determines the rate of pheromone evaporation. The best results were obtained when  $\rho$  was set to 0.95. With smaller values, the

<sup>1</sup><http://people.brunel.ac.uk/~mastjjb/jeb/info.html>.



**Fig. 3** Box plots for the sensitivity analysis of seven parameters for instance 10.100.00. Performance statistics for each parameter value are based on results from 50 runs. The default value of each parameter was:  $n_a = 50$ ,  $\alpha = 1$ ,  $\beta = 20$ ,  $\rho = 0.95$ ,  $\gamma = 8$ ,  $\lambda = 2$ ,  $P_{best} = 0.9$ . In each experiment only one of the values was changed. The values tested were:  $n_a \in \{10, 20, 50, 100\}$ ,  $\alpha \in \{0.5, 1, 2, 4\}$ ,  $\beta \in \{1, 5, 10, 20, 40\}$ ,  $\rho \in \{0.7, 0.8, 0.9, 0.95, 0.98\}$ ,  $\gamma \in \{2, 4, 8, 16\}$ ,  $\lambda \in \{1.5, 2, 2.5, 3\}$ ,  $P_{best} \in \{0.6, 0.7, 0.8, 0.9\}$

pheromone trails evaporate faster. As a result, the search concentrates earlier around the best solutions. While with larger values of  $\rho$  (e.g., 0.98), the pheromone trails

of the objects which are not reinforced decreases slowly, and hence, more cycles are required to make ants exploit the best solutions.

$\gamma$ ,  $\lambda$ ,  $P_{best}$  have an influence on the lower trail limit  $\tau_{min}$ . As suggested in Sect. 4.2,  $\gamma$  or  $\lambda$  should be set to a low value. When  $\gamma$  or  $\lambda$  is larger,  $\tau_{min}$  will increase faster. In this case, the performance of DMMAS is lower. With different values of  $P_{best}$ , the differences between solution quality are not significant. When  $P_{best} = 0.9$ , the performance is slightly better.

We also used the Wilcoxon rank sum test to determine whether the observed differences are significant at the 0.05 level. The statistical results show the default values are not worse than the other considered values.

## 6.2 The performance of DMMAS

To study the performance of DMMAS, we applied it to twenty-five instances. The first ten instances have 100 objects and 5 constraints, the second ten have 100 objects and 10 constraints, and the last five have 500 objects and 5 constraints.

We first compared the dynamic method with the SH method. When the SH method is used to select the lower trail limit, we observed that a good value of  $P_{best}$  is 0.05. Table 1 shows the results. For each instance, the best and average total profits as well as standard deviations are given. More discussion on performance assessment of stochastic algorithms is available in Birattari and Dorigo (2007). The best result for each instance was in boldface. It can be seen that the dynamic method can provide better results on 21 instances. Nevertheless, when the SH method is used, the results may be improved by fine-tuning the parameters to each individual instances. So far many research efforts have been devoted to parameter configuration, the interested reader is referred to Adenso-Diaz and Laguna (2006), Birattari et al. (2002), Birattari (2004), Hutter et al. (2007). Since the dynamic method is easy to implement and can make ants have very nice search capability, we focus on this method hereafter.

Then we compared DMMAS with other ACO-based algorithms. Since the results of Fidanova (2002) are inferior to those of Alaya et al. (2004), Leguizamón and Michalewicz (1999), we only report the results of the latter two algorithms. The results are summarized in Table 2. ACO<sub>LM</sub> gives the results in Leguizamón and Michalewicz (1999), Ant-knapsack gives the results in Alaya et al. (2004). Since the standard deviations of ACO<sub>LM</sub> are unavailable, only the best and average total profits are given. For other algorithms, the best and average profits as well as standard deviations are given. The best result for each instance was in boldface.

One can notice that DMMAS provides the best results and Ant-knapsack is better than ACO<sub>LM</sub>. Moreover, the numbers of solution constructions of these algorithms are 10.000, 60.000, and 10.000 respectively. Thus DMMAS can obtain the best solutions rapidly.

## 6.3 The performance of the hybrid DMMAS

We now studied the performance of the hybrid DMMAS, denoted as DMMAS+ls, which combines DMMAS and the local search procedure proposed in Sect. 5. DMMAS+ls works as follows: at each cycle, once each ant has constructed a solution,

**Table 1** Comparison of the SH method with the dynamic method on 25 instances. The first ten instances have 100 objects and 5 constraints, the second ten have 100 objects and 10 constraints, and the last five have 500 objects and 5 constraints

Instance	Best known <sup>a</sup>	The SH method			The dynamic method		
		Best	Average <sup>b</sup>	Std. dev.	Best	Average	Std. dev.
5.100.00	24381	<b>24381</b>	24354	26.1	<b>24381</b>	<b>24362</b>	23.8
5.100.01	24274	<b>24274</b>	24268	15.9	<b>24274</b>	<b>24273</b>	6.2
5.100.02	23551	<b>23551</b>	23531	8.5	<b>23551</b>	<b>23540</b>	7.2
5.100.03	23534	<b>23534</b>	23477	13.1	<b>23534</b>	<b>23482</b>	14.9
5.100.04	23991	<b>23991</b>	<b>23957</b>	13.3	<b>23991</b>	23954	10.8
5.100.05	24613	<b>24613</b>	24603	5.0	<b>24613</b>	<b>24608</b>	6.3
5.100.06	25591	<b>25591</b>	25559	35.2	<b>25591</b>	<b>25591</b>	0.0
5.100.07	23410	<b>23410</b>	23403	15.1	<b>23410</b>	<b>23404</b>	13.3
5.100.08	24216	<b>24216</b>	<b>24212</b>	8.5	<b>24216</b>	24211	5.9
5.100.09	24411	<b>24411</b>	24392	30.9	<b>24411</b>	<b>24406</b>	13.8
10.100.00	23064	<b>23064</b>	23037	26.4	<b>23064</b>	<b>23045</b>	19.6
10.100.01	22801	<b>22801</b>	22741	34.1	<b>22801</b>	<b>22742</b>	40.8
10.100.02	22131	<b>22131</b>	22078	31.6	<b>22131</b>	<b>22091</b>	29.8
10.100.03	22772	<b>22772</b>	<b>22711</b>	44.4	<b>22772</b>	22710	37.6
10.100.04	22751	<b>22751</b>	22606	33.9	<b>22751</b>	<b>22617</b>	43.9
10.100.05	22777	<b>22777</b>	22647	38.7	<b>22777</b>	<b>22663</b>	40.3
10.100.06	21875	<b>21875</b>	21794	49.7	<b>21875</b>	<b>21826</b>	28.4
10.100.07	22635	<b>22635</b>	22542	27.0	<b>22635</b>	<b>22557</b>	31.0
10.100.08	22511	22438	22396	14.8	22438	<b>22409</b>	17.3
10.100.09	22702	<b>22702</b>	22690	41.4	<b>22702</b>	<b>22696</b>	32.7
5.500.00	120134	120094	120039	23.5	120116	<b>120043</b>	31.2
5.500.01	117864	117840	<b>117779</b>	34.6	117854	117777	33.3
5.500.02	121112	121095	121021	28.9	121102	<b>121023</b>	35.4
5.500.03	120804	120772	120707	33.6	120778	<b>120707</b>	32.6
5.500.04	122319	122311	122241	30.6	<b>122319</b>	<b>122249</b>	32.1

<sup>a</sup>The best known results of the first twenty instances are given in Chu and Beasley (1998), and the others are given in Vasquez and Hao (2001)

<sup>b</sup>All average total profits are rounded off

and before updating pheromone trails, the local search procedure is applied to improve each constructed solution. The largest instances in Chu and Beasley (1998) were used as benchmark problems. They consist of 90 instances with 500 objects and the number of constraints varies from 5 to 30. There are 30 instances in each group which is denoted by  $m.n$ . The termination condition was a given time limit. When DMMA+ls was applied to 5.500, 10.500 and 30.500, the time limits were 100s, 200s and 400s respectively. These values were predetermined based on the criterion that DMMA+ls can converge satisfactorily.

**Table 2** Comparison of DMMAS with two ACO-based algorithms on 25 instances. The first ten instances have 100 objects and 5 constraints, the second ten have 100 objects and 10 constraints, and the last five have 500 objects and 5 constraints<sup>a</sup>

Instance	Best Known <sup>b</sup>	DMMAS			Ant-knapsack			ACOLM	
		Best	Average <sup>c</sup>	Std. dev.	Best	Average	Std. dev.	Best	Average
5.100.00	24381	<b>24381</b>	<b>24362</b>	23.8	<b>24381</b>	24342	29.3	<b>24381</b>	24331
5.100.01	24274	<b>24274</b>	<b>24273</b>	6.2	<b>24274</b>	24247	38.5	<b>24274</b>	24246
5.100.02	23551	<b>23551</b>	<b>23540</b>	7.2	<b>23551</b>	23529	8.0	<b>23551</b>	23528
5.100.03	23534	<b>23534</b>	<b>23482</b>	14.9	<b>23534</b>	23462	32.6	23527	23463
5.100.04	23991	<b>23991</b>	<b>23954</b>	10.8	<b>23991</b>	23946	31.8	<b>23991</b>	23950
5.100.05	24613	<b>24613</b>	<b>24608</b>	6.3	<b>24613</b>	24587	31.3	<b>24613</b>	24563
5.100.06	25591	<b>25591</b>	<b>25591</b>	0.0	<b>25591</b>	25512	43.8	<b>25591</b>	25505
5.100.07	23410	<b>23410</b>	<b>23404</b>	13.3	<b>23410</b>	23371	30.3	<b>23410</b>	23362
5.100.08	24216	<b>24216</b>	<b>24211</b>	5.9	<b>24216</b>	24172	32.9	24204	24173
5.100.09	24411	<b>24411</b>	<b>24406</b>	13.8	<b>24411</b>	24356	44.3	<b>24411</b>	24326
10.100.00	23064	<b>23064</b>	<b>23045</b>	19.6	<b>23064</b>	23016	42.2	23057	22996
10.100.01	22801	<b>22801</b>	<b>22742</b>	40.8	<b>22801</b>	22714	67.2	<b>22801</b>	22672
10.100.02	22131	<b>22131</b>	<b>22091</b>	29.8	<b>22131</b>	22034	66.9	<b>22131</b>	21980
10.100.03	22772	<b>22772</b>	<b>22710</b>	37.6	22717	22634	60.6	<b>22772</b>	22631
10.100.04	22751	<b>22751</b>	<b>22617</b>	43.9	22654	22547	66.3	22654	22578
10.100.05	22777	<b>22777</b>	<b>22663</b>	40.3	22716	22602	63.3	22652	22565
10.100.06	21875	<b>21875</b>	<b>21826</b>	28.4	<b>21875</b>	21777	44.9	<b>21875</b>	21758
10.100.07	22635	<b>22635</b>	<b>22557</b>	31.0	22551	22453	89.2	22551	22519
10.100.08	22511	<b>22438</b>	<b>22409</b>	17.3	<b>22511</b>	22351	69.4	22418	22292
10.100.09	22702	<b>22702</b>	<b>22696</b>	32.7	<b>22702</b>	22591	88.5	<b>22702</b>	22588
5.500.00	120134	120116	<b>120043</b>	31.2	119893	119658	135.8	N.A. <sup>d</sup>	N.A.
5.500.01	117864	117854	<b>117777</b>	33.3	117604	117423	130.4	N.A.	N.A.
5.500.02	121112	121102	<b>121023</b>	35.4	120846	120622	121.4	N.A.	N.A.
5.500.03	120804	120778	<b>120707</b>	32.6	120534	120279	152.3	N.A.	N.A.
5.500.04	122319	<b>122319</b>	<b>122249</b>	32.1	122126	121829	135.2	N.A.	N.A.

<sup>a</sup>The standard deviation of ACO<sub>LM</sub> is not available<sup>b</sup>The best known results of the first twenty instances are given in Chu and Beasley (1998), and the others are given in Vasquez and Hao (2001)<sup>c</sup>All average total profits are rounded off<sup>d</sup>N.A.: not available

At first, we evaluated the influence of the local search. Table 3 summaries the results of DMMAS and DMMAS+ls. According to the results, it is clear that the local search can effectively improve the performance of the pure ACO. In addition, pheromone trails have an important influence on the hybrid DMMAS. Guided by pheromone trails (where  $\alpha = 1$ ), ants can construct better solutions for the local search procedure.

**Table 3** Comparison of DMMAS+ls with DMMAS on thirty instances. The instances have 500 objects and 5 constraints. For each instance, the best and average total profits as well as standard deviation (std. dev.) are reported<sup>a</sup> (over 50 runs)

Instance	DMMAS+ls ( $\alpha = 1$ )			DMMAS+ls ( $\alpha = 0$ )			DMMAS		
	Best	Average	Std. dev.	Best	Average	Std. dev.	Best	Average	Std. dev.
5.500.00	<b>120148</b>	<b>120111</b>	17.3	119860	119648	67.1	120116	120056	25.5
5.500.01	<b>117879</b>	<b>117841</b>	13.7	117494	117360	59.1	117857	117786	27.9
5.500.02	<b>121131</b>	<b>121097</b>	17.7	120708	120526	54.1	121109	121043	27.2
5.500.03	<b>120804</b>	<b>120776</b>	11.3	120473	120293	48.6	120785	120715	24.1
5.500.04	<b>122319</b>	<b>122303</b>	16.3	121988	121812	53.8	<b>122319</b>	122254	29.8
5.500.05	<b>122024</b>	<b>121991</b>	14.8	121695	121562	47.2	121992	121936	23.6
5.500.06	<b>119127</b>	<b>119093</b>	12.6	118735	118614	46.7	119096	119043	26.4
5.500.07	<b>120568</b>	<b>120525</b>	20.0	120209	120121	48.0	120536	120472	27.6
5.500.08	<b>121575</b>	<b>121537</b>	14.4	121095	120961	51.3	121551	121479	31.9
5.500.09	<b>120717</b>	<b>120678</b>	17.9	120334	120192	53.3	120692	120627	25.7
5.500.10	<b>218428</b>	<b>218397</b>	12.9	218111	217943	54.3	218400	218344	28.1
5.500.11	<b>221202</b>	<b>221168</b>	15.5	220808	220668	47.7	221191	221117	30.9
5.500.12	<b>217534</b>	<b>217513</b>	13.4	217150	217039	55.9	217528	217459	30.9
5.500.13	<b>223560</b>	<b>223547</b>	11.0	223236	223136	39.6	<b>223560</b>	223499	24.6
5.500.14	<b>218966</b>	<b>218956</b>	11.1	218675	218528	52.4	218962	218905	25.9
5.500.15	<b>220530</b>	<b>220497</b>	14.1	220228	220132	45.9	220496	220455	17.2
5.500.16	<b>219989</b>	<b>219974</b>	15.8	219632	219519	46.6	219987	219924	31.5
5.500.17	<b>218194</b>	<b>218171</b>	10.9	217848	217758	47.7	218180	218124	25.8
5.500.18	<b>216963</b>	<b>216948</b>	11.2	216634	216551	44.1	216958	216904	28.6
5.500.19	<b>219719</b>	<b>219694</b>	8.0	219367	219188	49.4	219704	219657	20.6
5.500.20	<b>295828</b>	<b>295809</b>	13.9	295628	295485	42.3	<b>295828</b>	295764	20.9
5.500.21	<b>308086</b>	<b>308069</b>	9.7	307893	307805	32.7	308077	308023	25.6
5.500.22	<b>299796</b>	<b>299781</b>	13.0	299620	299527	36.4	<b>299796</b>	299738	16.5
5.500.23	<b>306480</b>	<b>306467</b>	9.0	306338	306238	31.1	<b>306480</b>	306427	27.5
5.500.24	<b>300342</b>	<b>300334</b>	11.2	300175	300076	29.7	300334	300280	22.2
5.500.25	<b>302571</b>	<b>302556</b>	7.6	302421	302327	37.9	302560	302525	19.7
5.500.26	<b>301329</b>	<b>301317</b>	7.9	301157	301082	36.0	301325	301278	26.7
5.500.27	<b>306454</b>	<b>306426</b>	8.5	306269	306200	26.2	306422	306388	20.4
5.500.28	<b>302828</b>	<b>302810</b>	13.5	302671	302566	39.3	302809	302765	22.1
5.500.29	<b>299906</b>	<b>299894</b>	9.1	299756	299656	38.0	299902	299845	23.8

<sup>a</sup>All average total profits are rounded off

Finally, we compared DMMAS+ls with two heuristic approaches which are among the best performing algorithms for MKP. The one is GA in Chu and Beasley (1998), which incorporates the standard genetic algorithm with a heuristic operator.

**Table 4** The average total profits of DMMAS+ls, GA and  $z^*$  on 90 largest instances<sup>a</sup>

Group	Tightness ratio	DMMAS+ls	GA	$z^*$
5.500	0.25	<b>120629</b>	120616	120623
5.500	0.5	<b>219509</b>	219503	219507
5.500	0.75	<b>302362</b>	302355	302360
10.500	0.25	<b>118603</b>	118566	118600
10.500	0.5	<b>217309</b>	217275	217298
10.500	0.75	<b>302588</b>	302556	302575
30.500	0.25	115541	115470	<b>115547</b>
30.500	0.5	<b>216223</b>	216187	216211
30.500	0.75	<b>302406</b>	302353	302404

<sup>a</sup>All averages are rounded off

The other is  $z^*$  in Vasquez and Hao (2001), which combines the linear programming with tabu search.

Table 4 shows the results of these algorithms. The second column in Table 4 indicates the tightness ratios (Chu and Beasley 1998). Since only the best total profits of GA and  $z^*$  are available, we report the best total profits of DMMAS+ls (see details in Appendix). It can be seen that DMMAS+ls outperforms GA. Compared with  $z^*$ , DMMAS+ls obtains better results in 8 out of 9 groups. With regard to CPU time, DMMAS+ls can deal with each instance within 400s. Therefore DMMAS+ls can obtain promising solutions within a reasonable amount of computation time.

## 7 Conclusions

In this paper, we have proposed an algorithm based on ACO, called DMMAS and applied to the MKP. DMMAS differs from the standard ACO in many components due to the characteristics of the MKP. A problem-dependent pheromone trail and heuristic information were defined. We also proposed a method to choose the lower trail limit. Additionally, we presented a hybrid algorithm which combines DMMAS with a local search procedure.

We compared DMMAS with three ACO based algorithms. The comparison shows that DMMAS is superior to those algorithms. We also applied our hybrid algorithm to the benchmark problems and compared with two promising hybrid algorithms. The results demonstrate that our hybrid algorithm is competitive.

Although the method of choosing the lower trail limit is motivated by the MKP, it can be applied to other subset problems. In addition, since machine learning provides an alternative and promising approach in tuning ACO (Birattari et al. 2002; Birattari 2004), we will further study this interesting direction.

**Acknowledgements** The authors would like to thank the anonymous reviewers for their helpful comments and suggestions. This work is supported by National Basic Research Program (973 Program) under grant No. 2007CB311006.

## Appendix

Tables 5 and 6:

**Table 5** Best total profits of the instances with 500 objects and 10 constraints

Instance	Best	Instance	Best	Instance	Best
10.500.00	117784	10.500.10	217353	10.500.20	304353
10.500.01	119198	10.500.11	219041	10.500.21	302371
10.500.02	119196	10.500.12	217797	10.500.22	302416
10.500.03	118813	10.500.13	216868	10.500.23	300757
10.500.04	116487	10.500.14	213816	10.500.24	304367
10.500.05	119454	10.500.15	215086	10.500.25	301796
10.500.06	119813	10.500.16	217931	10.500.26	304949
10.500.07	118312	10.500.17	219984	10.500.27	296450
10.500.08	117779	10.500.18	214346	10.500.28	301331
10.500.09	119197	10.500.19	220865	10.500.29	307089

**Table 6** Best total profits of the instances with 500 objects and 30 constraints

Instance	Best	Instance	Best	Instance	Best
30.500.00	115942	30.500.10	218034	30.500.20	301643
30.500.01	114732	30.500.11	214626	30.500.21	300014
30.500.02	116613	30.500.12	215903	30.500.22	305062
30.500.03	115263	30.500.13	217862	30.500.23	302001
30.500.04	116487	30.500.14	215622	30.500.24	304416
30.500.05	115734	30.500.15	215829	30.500.25	296962
30.500.06	114107	30.500.16	215883	30.500.26	303328
30.500.07	114252	30.500.17	216448	30.500.27	306944
30.500.08	115271	30.500.18	217333	30.500.28	303158
30.500.09	117011	30.500.19	214690	30.500.29	300531

## References

- Adenso-Diaz, B., Laguna, M.: Fine-tuning of algorithms using fractional experimental designs and local search. *Oper. Res.* **54**, 99–114 (2006)
- Alaya, I., Solnon, C., Ghdira, K.: Ant algorithm for the multidimensional knapsack problem. In: *International Conference on Bioinspired Optimization Methods and their Applications*, pp. 63–72 (2004)
- Battiti, R., Tecchiolli, G.: Local search with memory: Benchmarking RTS. *OR Spektrum* **17**, 67–86 (1995)
- Birattari, M.: The problem of tuning metaheuristics as seen from a machine learning perspective. PhD thesis, Université Libre de Bruxelles, Brussels, Belgium (2004)

- Birattari, M., Dorigo, M.: How to assess and report the performance of a stochastic algorithm on a benchmark problem: Mean or best result on a number of runs. *Optim. Lett.* **1**(3), 309–311 (2007)
- Birattari, M., Stützle, T., Paquete, L., Varrenttrapp, K.: A racing algorithm for configuring metaheuristics. In: Langdon, W.B. (ed.) *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 11–18. Morgan Kaufmann, San Francisco (2002)
- Blum, C., Blesa, M.J.: New metaheuristic approaches for the edge-weighted k-cardinality tree problem. *Comput. Oper. Res.* **32**, 1355–1377 (2005)
- Chu, P.C., Beasley, J.E.: A genetic algorithm for the multidimensional knapsack problem. *J. Heur.* **4**, 63–86 (1998)
- Devore, J.L.: *Probability and Statistics: for Engineering and the Sciences*. Duxbury, N. Scituate (2000)
- Dorigo, M., Blum, C.: Ant colony optimization theory: A survey. *Theor. Comput. Sci.* **344**, 243–278 (2005)
- Dorigo, M., Stützle, T.: The ant colony optimization metaheuristic: algorithms, applications, and advances. In: Glover, F., Kochenberger, G. (eds.) *Handbook of Metaheuristics*, pp. 251–285. Kluwer Academic, Norwell (2002)
- Dorigo, M., Stützle, T.: *Ant Colony Optimization*. MIT Press, Cambridge (2004)
- Dorigo, M., Maniezzo, V., Coloni, A.: Ant system: Optimization by a colony of cooperating agents. *IEEE Trans. Syst. Man Cybern. Part B* **26**, 29–41 (1996)
- Dorigo, M., Di Caro, G., Gambardella, L.M.: Ant algorithms for distributed discrete optimization. *Artif. Life* **5**, 137–172 (1999)
- Fidanova, S.: ACO algorithm for MKP using various heuristic information. In: *The 5th International Conference on NMA. Lecture Notes in Computer Science*, vol. 2542, pp. 438–444. Springer, Berlin (2002)
- Freville, A., Plateau, G.: An efficient preprocessing procedure for the multidimensional 0-1 knapsack problem. *Discrete Appl. Math.* **49**, 189–212 (1994)
- Gavish, B., Pirkul, H.: Allocation of databases and processors in a distributed computing system. In: Akoka, J. (ed.) *Management of Distributed Data Processing*, pp. 215–231. North-Holland, Amsterdam (1982)
- Gilmore, P.C., Gomory, R.E.: The theory and computation of knapsack functions. *Oper. Res.* **14**, 1045–1075 (1966)
- Glover, F., Kochenberger, G.A.: Critical event tabu search for multidimensional knapsack problems. In: Osman, I.H., Kelly, J.P. (eds.) *Metaheuristics: Theory and Applications*, pp. 407–427. Kluwer Academic, Norwell (1996)
- Hanafi, S., Freville, A.: An efficient tabu search approach for the 0-1 multidimensional knapsack problem. *Eur. J. Oper. Res.* **106**, 659–675 (1998)
- Hutter, F., Hoos, H., Stützle, T.: Automatic algorithm configuration based on local search. In: AAAI-07, pp. 1152–1157. AAAI Press, Menlo Park (2007)
- Leguizamón, G., Michalewicz, Z.: A new version of ant system for subset problem. In: *Proceedings Congress on Evolutionary Computation*, pp. 1459–1464 (1999)
- Levine, J., Ducatelle, F.: Ant colony optimization and local search for bin packing and cutting stock problems. *J. Oper. Res. Soc.* **55**, 705–716 (2004)
- Maniezzo, V., Coloni, A.: The ant system applied to the quadratic assignment problem. *IEEE Trans. Knowl. Data Eng.* **11**, 769–778 (1999)
- Morrison, R.W., De Jong, K.A.: Measurement of population diversity. In: Collet, P., Fonlupt, C., Hao, J.K., Lutton, E., Schoenauer, M. (eds.) *The 5th International Conference on EA. Lecture Notes in Computer Science*, vol. 2310, pp. 31–41. Springer, Berlin (2001)
- Shih, W.: A branch and bound method for the multiconstraint zero-one knapsack problem. *J. Oper. Res. Soc.* **30**, 369–378 (1979)
- Solnon, C.: Ants can solve constraint satisfaction problems. *IEEE Trans. Evol. Comput.* **6**, 347–357 (2002)
- Solnon, C., Bridge, D.: An ant colony optimization meta-heuristic for Subset selection problems. In: Nedjah, N., Mourelle, L. (eds.) *System Engineering Using Particle Swarm Optimization*, pp. 7–29. Nova Science, Hauppauge (2006)
- Solnon, C., Fenet, S.: A study of ACO capabilities for solving the maximum clique problem. *J. Heur.* **12**, 155–180 (2006)
- Stützle, T.: Parallelization strategies for ant colony optimization. In: *Proceedings of the 5th International Conference on Parallel Problem Solving from Nature*, pp. 722–731 (1998)
- Stützle, T., Hoos, H.H.: Max-min ant system. *Future Gener. Comput. Syst.* **16**, 889–914 (2000)
- Vasquez, M., Hao, J.K.: A hybrid approach for the 0-1 multidimensional knapsack problem. In: *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, vol. 1, pp. 328–333 (2001)