Advances in Computer Science Research, (ACSR), volume 52
2016 International Conference on Computer Engineering and Information Systems (CEIS-16)

Study on Ant System for the 0-1 Multiple Knapsack Problem with Knapsack Dependent Values

Yue Zhang, Xing-Ye Dong, You-Fang Lin
Beijing Key Lab of Traffic Data Analysis and Mining,
School of Computer and IT, Beijing Jiaotong University,
Beijing 100044, China
E-mail: 14120449@bjtu.edu.cn, xydong@bjtu.edu.cn, yflin@bjtu.edu.cn

***Abstract*-The values of items of the classic knapsack problems are independent of knapsacks. Inspired from some practical problems, a 0-1 multiple knapsack problems with knapsack dependent item values are proposed. A Greedy heuristic is proposed to provide a feasible solution and a basis for further comparison, and then three ant system algorithms are proposed. To relieve the stagnant phenomenon of the AS and MMAS, MMAS2 is developed with the newly proposed method by mapping the objective periodically for the purpose of adjusting the importance of the pheromone trails, in which the max-min pheromone limitation is also applied. And it has a more chance to find better solutions. Experimental results on randomly generated instances show that the proposed MMAS2 surpasses the MMAS significantly in performance.**

Keywords-multiple knapsack problem; knapsack dependent values; max-min ant system; greedy heuristic

I. INTRODUCTION

The Knapsack Problem is a well known combinatorial optimization problem, and has been studied for decades. In the classic knapsack problem, the values or profits and weights of items are given in advance, also a set of knapsacks with deterministic capacities are included. There are n items and m knapsacks in total. The value and weight of item j are denoted by v_j and w_j respectively, and the capacity of knapsack i is denoted by c_i, respectively. The objective is to pack a subset of the items into the knapsacks to maximize the total values without overflow the capacity of each knapsack [1]. The classic KP and its variants have many applications in the real world, such as transportation scheduling and products delivery [2, 3, 4].

The 0-1 knapsack problem has been extensively studied in the literature [1], and many exact algorithms have been proposed [5]. In many application fields, classic knapsack problem needs to be extended, e.g. a multiple knapsack problem combined with the assignment problem [4], and a knapsack problem with setup [3]. In the literature, the values of the items are independent of knapsacks. In this work, a 0-1 multiple knapsack problem with knapsack dependent item values is proposed, i.e. the value of packing item j into knapsack i is denoted by v_{ij}, then the proposed problem can be formulated as follows:

$$max \sum_{j=1}^{n}\sum_{i=1}^{m} x_{ij}v_{ij} \quad (1)$$

s.t.

$$\sum_{j=1}^{n} x_{ij}\, w_j \le c_i,\ 1 \le i \le m, \quad (2)$$

$$\sum_{i=1}^{m} x_{ij} \le 1,\ 1 \le j \le n, \quad (3)$$

$$x_{ij} \in \{0,1\},\ 1 \le i \le m,\ 1 \le j \le n. \quad (4)$$

where x_{ij} is a decision variable, which has value 1 if the item j is packed into the knapsack i and 0 otherwise. So the total value of the packed items is given by formula (1), and formula (2) constraints that the total weight packed in each knapsack does not exceed the corresponding capacity, and the last two formulae limit that each item is either packed into a knapsack or rejected. To the best of our knowledge, this model has not been discussed in the literature.

The proposed model is motivated from a practical problem of selecting the bus stations for different bus fleets in the public transportation system. In the problem, each bus fleet needs to be assigned a bus station, and there exists deadhead drive that means no passengers in bus from the station to their original stop. The objective is to find an assignment for the fleets to minimize the total deadhead drive in the system, i.e. to maximize the profits of the enterprise. This knapsack problem model also can be applied to many other practices, such as drawing up a cargo plan for company, where the cost of packing the cargo to different ship may be different. In this work, an abstract problem is drawn as above, without any practice consideration.

For the solving of the proposed problem, firstly a heuristic is designed to provide a feasible solution and a basis for further comparison, then three ant system algorithms are proposed to achieve an optimized solution. The first one is a traditional ant system and named AS. The second one is based on the well known max-min idea [6] and simply named MMAS in this paper. However, in both above ant systems, the stagnant phenomenon is an obvious drawback during the search process. Also, if the objective of

the problem is large, the effect of the pheromone trails as compared with the heuristic information on the selection probability will be quite small. In order to relieve the above phenomena, a scheme is designed to change the limitation of the pheromone periodically to narrow the gap between the maximum pheromone trail and the minimum pheromone trail, and so give the ant more chance to construct a different solution, and then forming the third ant algorithm named MMAS2. Experimental results show that the proposed MMAS2 surpasses the MMAS significantly in performance.

In the rest of the paper, a heuristic is derived from a greedy idea in Section II. In Section III, the basic components of ant system are illustrated firstly, followed by the details of three ant systems. All the presented algorithms are evaluated on a set of randomly generated instances in Section IV, with the analysis of the algorithms features. The paper is concluded in Section V.

II. THE GREEDY HEURISTIC

The idea of the proposed greedy heuristic is that assign the item with maximum unit value to its corresponding knapsack one by one until the knapsacks are full enough and can not pack any more items into them. In details, the unit values v_{ij}/w_j for item j are computed, and if the capacity of all knapsack is smaller than the weight of item j, the unit value is defined as zero, indicating that the item j can not be packed into knapsack i. In the following step, the item j with the maximum v_{ij}/w_j is selected and packed into knapsack i. After that, the capacity of knapsack i is decreased by w_j, and the maximum unit values should be updated for the items which can be packed into knapsack i previously. The solution can be denoted by pairs of (i,j), meaning that item j is packed into knapsack i .The greedy heuristic is denoted by Greedy and described as Algorithm 1.

Algorithm 1: Greedy heuristic

1. Initialize solution S as empty;
2. Compute the unit value v_{ij}/w_j for all items and all knapsacks and the maximum unit value v_{ij}/w_j to its corresponding knapsack i for all items;
3. while(true)
4. Select the maximum unit value v_{ij}/w_j to its corresponding knapsack i from unpacked items;
5. if(the selected v_{ij}/w_j is zero)
6. break;
7. else
8. Put pair (i,j) into solution S, and decrease the remaining capacity of knapsack i by w_j;
9. Update v_{ij}/w_j= 0 and v_{ij}/w_j for all items whose corresponding knapsack is i.;
10. end if
11. end while
12. Output S.

The computation of v_{ij}/w_j needs to compute nm numbers; the loop will run at most n times; and in the loop, the selection of the maximum v_{ij}/w_j can be implemented in O(n); the update procedure for v_{ij}/w_j can be realized in O(nm), so the complexity of the Greedy is O(n^2m).

III. THE PROPOSED ANT SYSTEM

Ant Colony Optimization (ACO) is a widely used metaheuristic, and has been applied to some knapsack problem [7, 8]. In this section, the basic components of ant systems are illustrated firstly, and then a traditional ant system is formulated. After that, an MMAS is designed based on the idea of MMAS proposed by Stützle and Hoos [6]. In order to promote the performance further, a method is developed to adjust the pheromone periodically, alleviating the stagnant phenomenon and augmenting the adaptation of the algorithm to different scales of problems. By applied this method, the MMAS is improved as MMAS2.

A. Solution Construction

Suppose the solution is denoted by S, then the construction process is to choose (i,j) pairs into S continuously until no more items can be packed in any of the knapsacks or all items have been packed. Let the chosen probability for pair (i,j) be p_{ij}, and it is defined as:

$$p_{ij} = \frac{[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{(i,j)\notin S}[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}} \quad (5)$$

where τ_{ij} is the pheromone factor of the pair (i,j), $\eta_{ij}$is the corresponding heuristic factor and defined as v_{ij}/w_j in this work, and α and β are two parameters that determine the relative importance of these two factors. Note that if item j can not be packed into knapsack i, then let v_{ij}/w_j be zero.

The solution construction is described as the following Algorithm 2.

Algorithm 2: Construction Procedure

1. Initialize solution $S \leftarrow \emptyset$;
2. while(true)
3. Compute $total \leftarrow \sum_{(i,j)\notin S}[\tau_{ij}]^{\alpha}[\eta_{ij}]^{\beta}$;
4. if(*total* is zero)
5. break;
6. end if
7. Choose a pair (i,j) according to p_{ij} defined in Eq. (5);
8. Add the pair (i,j) to S;
9. Decrease the remaining capacity of knapsack i by w_j;
10. Update the η_{ij} for the knapsack chosen and for all the items in the chosen knapsack;
11. end while
12. Output S.

B. Pheromone Trail

The pheromone trails indicate the information laid by ants and reflect the probability for making a decision. The pheromone is laid on all the edges (i,j) in this work. The amount of the pheromone is related to the objective. Suppose S_{best} denotes the current best solution of the problem, and $f(S_{best})$ is a function of the S_{best}.In traditional ant system, the initial pheromone trails $\tau_{ij}^{(0)}$ are determined by $1/f(S_{best})$.

After each iteration, the S_{best} may be updated, and then the pheromone trails need to be updated. Suppose S be the best solution found during the iteration, and the edge (i,j) belongs to S, then the pheromone trail laid on it should be strengthened by:

$$\tau_{ij}^{(k+1)} = \rho\tau_{ij}^{(k)} + \frac{f(S)}{f^2(S_{best})} \quad (6)$$

where $\rho \in (0,1)$ is the persistence factor. This formula indicates that the better the solution S found in an iteration the more pheromone laid on the edges.

For the edges do not belong to S, the pheromone trail laid on it should be weakened by:

$$\tau_{ij}^{(k+1)} = \rho\tau_{ij}^{(k)} \quad (7)$$

C. Traditional Ant System

Based on the above description, the pseudo code of a traditional ant system named AS can be illustrated as Algorithm 3, where max_iter denotes the maximum iterations, ant_num denotes the number of the ants:

Algorithm 3: The Traditional AS

1. Generate an initial solution S by the Greedy heuristic, set $S_{best} = S$;
2. Initialize pheromone trails to $1/f(S_{best})$;
3. Set max_iter, ant_num, and let $iter = 0$;
4. while($iter < max\_iter$)
5. Generate ant_num solutions according to Algorithm 2;
6. Let S be the best one among the solutions found in the previous step;
7. if(S is better than S_{best})
8. Update S_{best} as S;
9. end if
10. Update pheromone trails on the ant cycle solution S;
11. end while
12. Output S_{best}.

D. The Proposed MMAS

In literature, limit the pheromone trails, and so give the worst solution a chance to be constructed and limit the probability of constructing the current best solution, is an effective strategy [6]. According to Stützle and Hoos [6], the pheromone trail is limited to τ_{max} and τ_{min}, which are computed as follows:

$$\tau_{max} = \frac{1}{1-\rho} \cdot \frac{1}{f(S_{best})} \quad (8)$$

$$\tau_{min} = \frac{\tau_{max}(1 - \sqrt[n]{p_{best}})}{(m-1)\sqrt[n]{p_{best}}} \quad (9)$$

where p_{best} denotes the probability that the current best solution is constructed when the algorithm has converged, and it is a preseted parameter.

According to the above definition, τ_{max} and τ_{min} will be changed when a better solution is found. Initially, the current best solution is gotten by the Greedy heuristic, and the initial pheromone for all pairs (i,j) are set to τ_{max}.

The pseudocode of the proposed MMAS is similar to Algorithm 3, other than the limitation of the pheromone, i.e. when the pheromone for pair (i,j) is larger than τ_{max}, then it is just set to τ_{max}; otherwise, if the pheromone is smaller than τ_{min}, it is just set to τ_{min}.

E. The Proposed MMAS2

In traditional ant system, the amount of updated pheromone is related directly to the best solution S_{best} and S found in the iteration. In the experiments, however, we notice that the S is quite large for large instances, such as the instances including 30 knapsacks and 500 items, and then weaks the effect of the pheromone trail on the selection probability as compared with the heuristic information in Eq. (5). So, in order to make the pheromone trails more importance, the maximum trail τ_{max} should be increased, and so the objective should be mapped to a smaller value according to Eq. (8). However, it is hard to determine the mapped value, so we mapped the objective to a range from a relatively smaller value to the objective itself, i.e. the objective is mapped as follows in the kth iteration:

$$f^{(k)}(S) = \frac{1}{3} \cdot S \cdot ratio^{k-K} \quad (10)$$

where $ratio > 1$ is the growth rate of the pheromone trail, and $K = \max\{p \times l | p, l \in N, 0 \le p \times l \le k, ratio^l \ge 3, . ratio^{l-1} < 3\}$. According to this formula, the mapped value will fluctuate from $\frac{S}{3}$ to S.Though this design makes the change of the importance of the pheromone trails, it is quite arbitrarily. And by mapping the objective periodically, it narrows the gap between the maximum pheromone and minimum pheromone every cycle and gives the ant a chance to find better solutions.

After a solution is constructed, the pheromone trails need to be updated. Suppose the constructed solution in iteration $k+1$ is S, and pair (i, j) is in it, then the pheromone $\tau_{ij}^{(k+1)}$ is updated in the same way as formula (6); otherwise, it is updated as formula (7). Further, if the $\tau_{ij}^{(k+1)}$ is smaller than τ_{min}, then it is set to τ_{min}; if the $\tau_{ij}^{(k+1)}$ is larger than τ_{max}, then it is set to τ_{max}. The pseudocode is similar to the MMAS, and so it is omitted here.

IV. EXPERIMENTAL RESULTS

As the discussed problem is firstly proposed, we generate some instances randomly to test the proposed algorithms. There are totally 150 instances generated independently, arranged into 15 groups. Each group contains 10 problems,

having the same number of items and knapsacks. In detail, the number of items has 5 levels with 20, 50, 100, 200 and 500 for 4 knapsacks, has 4 levels with 50, 100, 200 and 500 for 10 knapsacks, and has 3 levels with 100, 200 and 500 for 20 and 30 knapsacks, respectively. The algorithms are implemented in C++ and tested on a server with sixteen 2.60GHz Intel Xeon processors and 16G RAM running the CentOS operating system. Though there are multiple kernels, we do not use any parallel techniques in the implementation and so only one kernel is used in each experiment.

The parameters for all the ant system algorithms are set as follows: ρ is 0.9, p_{best} is 0.05, max_iter is 500, $\alpha$is 1, $\beta$is 5, $ant_num$is 25 and *ratio* is 1.05. The parameters are chosen according to some preliminary tests on different scales of instances. For small scale instances, the best solutions are collected by CPLEX running on a computer with four 1.7GHz Intel Core processors and 8G RAM running the Windows 7 operating system.

As for the evaluation of the performance for the proposed algorithms, the relative percentage deviation (RPD) is computed as:

$$RPD = \frac{S_{best} - S_{base}}{S_{base}} \times 100 \qquad (11)$$

where S_{best} is the solution found by the proposed algorithms; as for the S_{base}, it is the optimal solution gotten by CPLEX software for small instances, and it is the solution gotten by the proposed Greedy heuristic for large instances. Negative RPD means that the S_{best} is worse than the S_{base}, and positive RPD means that the S_{best} is better than the S_{base}. The larger the RPD reflects the better the performance.

For small instances, i.e. consisting of 4 knapsacks with all levels of item numbers and 10 knapsacks with 50 items, the optimal solutions can be gotten by CPLEX software in reasonable CPU time, and then the RPDs are computed according to (11) with the S_{base} the optimal objective. The comparison results are listed in Table I, where all RPDs for each group are averaged and denoted by ARPD. As the Greedy is a heuristic, i.e. whose result is the same for any run on the same instance, and then the Greedy is only run one time and it can construct a solution in negligible time for all scales of instances, e.g. it can construct a solution for 4 knapsacks and 500 items in 2 milliseconds. As for the proposed ant systems, they are run 10 times independently on each instance and the best solution among the 10 runs is chosen as the S_{best} to compute the RPD. After that, the RPDs are averaged over groups, along with the CPU time and the deviations of the RPDs (denoted by std.). The CPU time of three ant system algorithms is quite similar and so only the CPU time of the MMAS2 is listed.

TABLE I. TABLE COMPARISON RESULTS ON SMALL INSTANCES

Prob.		CPLEX	Greedy	AS		MMAS		MMAS2		
m	n	CPU/s	ARPD	ARPD	std.	ARPD	std.	ARPD	std.	CPU/s
4	20	<1	-5.874	-0.039	0.12	0	0	0	0	<1
	50	1	-3.112	-0.100	0.12	-0.027	0.05	-0.027	0.05	<1
	100	12	-2.364	-0.163	0.13	-0.119	0.14	-0.107	0.14	3
	200	382	-1.463	-0.178	0.10	-0.149	0.12	-0.120	0.07	14
	500	26993	-1.338	-0.247	0.12	-0.148	0.07	-0.124	0.08	89
20	50	12	-8.215	-0.872	0.68	-0.549	0.61	-0.497	0.60	2

From Table I, it can be seen that all three ant system algorithms are much better than the Greedy heuristic, showing the optimization ability of the ant system. The ARPDs of the MMAS are all larger than the counterpart of the AS, and it confirms that the max-min strategy is quite robust and it is also effective on this problem. However, the MMAS2 performs the best. The ARPDs gotten by it are all significantly larger than that of the MMAS on all group of instances other than the two smallest groups, on which the performance of both is the same. The deviations of the RPDs indicate that the performance is quite stable, especially for the MMAS and MMAS2. The ARPDs of all three ant systems are less than 1%, indicating that they can find quite good solutions. As for the group of 4 knapsacks with 20 items, the MMAS2 and MMAS can find the optimal solution in each run.

For large problems, i.e. for the remaining tested instances, the CPLEX software can't obtain their optimal solutions in reasonable computational time. So, the proposed ant system algorithms are only compared with the proposed Greedy heuristic. The comparison results are listed in Table II, where the meanings of the columns are the same as that in Table I.

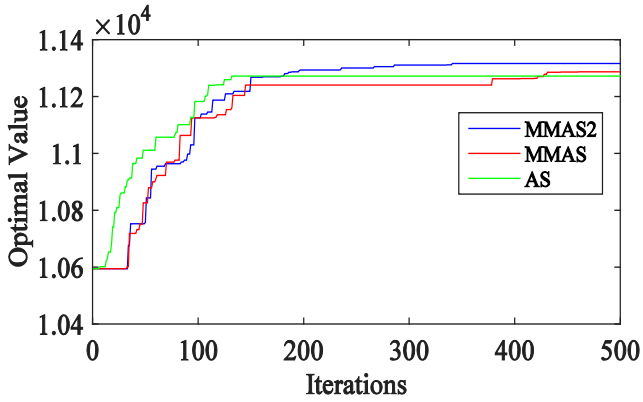
TABLE II. TABLE COMPARISON RESULTS ON LARGE INSTANCES

Prob.		Greedy	AS		MMAS		MMAS2		
m	n	CPU/s	ARPD	std.	ARPD	std.	ARPD	std.	CPU/s
10	100	<1ms	4.239	1.48	4.373	1.45	4.572	1.42	7
	200	<1ms	2.175	0.72	2.243	0.67	2.379	0.72	30
	500	2ms	1.425	0.30	1.520	0.28	1.581	0.29	196
20	100	<1ms	8.714	1.54	9.595	1.75	9.642	1.46	16
	200	<1ms	4.448	1.15	4.814	1.13	4.927	1.24	61
	500	2ms	1.877	0.30	2.037	0.24	2.138	0.28	370
30	100	<1ms	11.879	3.12	13.002	3.32	13.058	3.12	26
	200	<1ms	6.095	1.30	6.507	1.49	6.615	1.53	125
	500	2ms	2.508	0.57	2.621	0.53	2.722	0.49	535

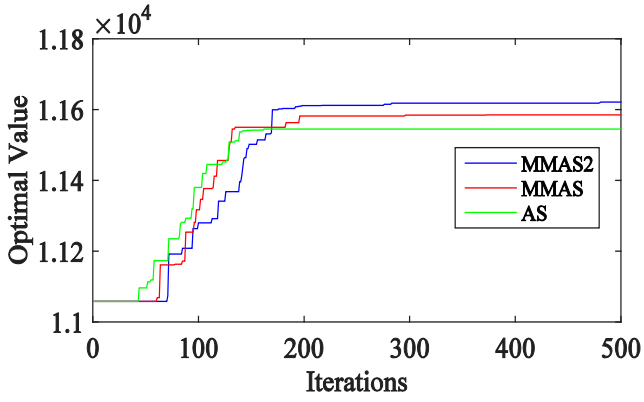
The results show that all the ant system algorithms can improve the performance of the Greedy heuristic much, and the smaller the instances are, the larger the improvement achieves. As for the performance of the three ant systems, the MMAS2 ranks the first, followed by the MMAS. This further confirms the effectiveness of the max-min strategy, and the effectiveness of the fluctuation design of the objective.

In order to observe whether the performance between the ant systems is statistically different, paired sample *t*-Test is conducted between the MMAS2 and MMAS, and the *p*-value is 0.003<0.05, indicating that the MMAS2 performs significantly better than MMAS. The *p*-value between MMAS and AS is 0.007<0.05, showing that the MMAS performs significantly better than AS.

To have a closer view about the search process of the ant systems, an instance each from two groups is selected, and evolution process for three ant systems is plotted in Fig. 1. From Fig. 1, it can be seen that at the beginning of the search, the AS can find a better solution firstly. However, the AS is trapped into a local optima quickly, with little chance to escape from it. As for the MMAS, though it costs more time to find a better solution, it still has a chance to find better solutions after much iterations. The MMAS2 performs similar to MMAS at the beginning of the search, and it has more chance to find better solutions and performs better when much iterations have carried out. For both MMAS and MMAS2, the stagnant phenomenon is an obvious alleviated, and the proposed mapping the objective periodically is quite effective. The performance of these ant systems is quite similar on other scale of instances, and so more details are omitted here.



(a) 20 knapsacks and 200 items



(b) 30 knapsacks and 200 items

Figure 1. The evaluation process of the three ant systems on different instances.

V. CONCLUSIONS

In this paper, a 0-1 multiple knapsack problem with knapsack dependent item values is proposed. Different from the classic knapsack problems, in which the values of the items are deterministic and independent of knapsacks, the values of the discussed problem are variable and dependent with the knapsacks. To our knowledge, this model is firstly discussed in this work.

A heuristic Greedy is proposed, which can construct a solution in negligible time even for the instances including 30 knapsacks and 500 items, and the performance is quite good. Then three ant system algorithms are proposed and called AS, MMAS and MMAS2, respectively. Both AS and MMAS are derived from the ideas in literature, with the problem-specific adaptation. In order to promote the performance further, MMAS2 is applied, with the newly proposed method by mapping the objective periodically for the purpose of adjusting the importance of the pheromone trails. Experimental results show that the proposed MMAS2 relieves the stagnant phenomenon. Also, MMAS2 surpassed the MMAS significantly in performance.

In our future work, the upper bound of this problem is going to be estimated. And in the practice of the public transportation system in China, the bus stations are often too small to park all the buses, and so only some part of each

fleet can park into the bus stations, then how to make the assignment of the bus stations for all the fleets under some constraints is an interesting problem, i.e. the discussed problem needs to be extended to satisfy the practice.

ACKNOWLEDGMENT

This work is supported by The Fundamental Research Funds for the Central Universities of China (Project Ref. 2014JBM034, Beijing Jiaotong University).

REFERENCES

[1] Kellerer, H., Pferschy, U., Pisinger, D., 2004. Knapsack problems. Berlin, Heidelberg: Springer.

[2] Chen, K., Ross, S.M., 2014. An adaptive stochastic knapsack problem. European Journal of Operational Research, 239:625-635.

[3] Chebil, K., Khemakhem, M., 2015. A dynamic programming algorithm for the Knapsack Problem with Setup. Computers & Operations Research, 64(C):40-50.

[4] Kataoka, S., Yamada, T., 2014. Upper and lower bounding procedures for the multiple knapsack assignment problem. European Journal of Operational Research, 237(2):440-447.

[5] Martello, S., Pisinger, D., Toth, P., 2000. New trends in exact algorithms for the 0-1 knapsack problem. European Journal of Operational Research, 123(2), 325-332.

[6] Stützle, T., Hoos, H., 2000. MAX–MIN ant system. Future Generation Computer Systems, 16:889-914.

[7] Lee, S.Y., Bau, Y.T., 2012. An ant colony optimization approach for solving the Multidimensional Knapsack Problem. Computer & Information Science (ICCIS), International Conference on, IEEE, pp. 441-446.

[8] Nakbi, W., Alaya, I., Zouari, W., 2015. A hybrid lagrangian search ant colony optimization algorithm for the multidimensional knapsack problem. Procedia Computer Science, 60(1):1109-1119.