Available online at www.sciencedirect.com
ScienceDirect
Advances in Engineering Software 38 (2007) 50–58
ADVANCES IN ENGINEERING SOFTWARE
www.elsevier.com/locate/advengsoft

An efficient solution to biobjective generalized assignment problem

Cai Wen Zhang, Hoon Liong Ong *

Department of Industrial and Systems Engineering, National University of Singapore, 10 Kent Ridge Crescent, Singapore 119260, Singapore

Received 25 October 2005; received in revised form 17 February 2006; accepted 2 June 2006
Available online 4 August 2006

Abstract

The generalized assignment problem (GAP) has found applications in many real world problems. In this paper, we examine the GAP from a multiobjective point of view to accommodate some real world situations where more than one objective is involved. An efficient LP-based heuristic is proposed to solve the biobjective generalized assignment problem (BiGAP). Extensive computational experiments are carried out to evaluate the performance of the proposed method. The results show that the proposed approach is able to generate good approximations to the nondominated frontier of the BiGAP efficiently, especially when the ratio of the number of items to the number of knapsacks is large.

Keywords: ε-Nondominated solution; Generalized assignment problem; LP-based heuristic; Multiobjective integer programming

1. Introduction

Multiobjective optimization techniques have continued to attract a lot of attention in recent years due to their wide applications [7,30,28,1,13,20,26,29,21,14,12,19,33]. In this study, we consider the generalized assignment problem (GAP). GAP and its variants have wide applications in solving real world problems including network design and scheduling, resource scheduling, location and allocation, and so on. It is concerned with optimally assigning n jobs to m agents such that each job is assigned to exactly one agent, while the total resource capacity of each agent is not exceeded. It has been shown that various location problems can be modeled and solved as GAPs [25]. Many researchers have proposed various methods, including exact and approximate ones, for solving the GAP and its variants; see [2,4,6,8,15,17,24,31,32]. An excellent survey on the algorithms for solving the GAP is provided in [5]. In this study, the GAP is considered from a multiobjective point of view, in particular, the biobjective case, and an efficient LP-based heuristic (LPH) is proposed to solve the biobjective generalized assignment problem (BiGAP). It will be shown that the proposed approach can efficiently generate a good approximation to the nondominated frontier of the BiGAP.

The rest of this paper is organized in the following way. Section 2 introduces the BiGAP. The proposed method for solving the BiGAP is described in Section 3. Extensive computational results are presented in Section 4. Finally, Section 5 concludes this study.

2. The biobjective generalized assignment problem (BiGAP)

Given n items, m knapsacks, the profit p_{ij} and the weight w_{ij} corresponding to the assignment of item j to knapsack i, and the total capacity c_i available for knapsack i, the single objective GAP is to determine how to select and assign each of the items to exactly one of the knapsacks in order to maximize the total profit, subject to the knapsack capacity constraints. The GAP is one of the well known NP-hard problems [23].

In this study we are concerned with the BiGAP, where each item has two attributes: attribute-1, denoted p^1_{ij}, and attribute-2, denoted p^2_{ij}. Without loss of generality, we assume both objectives are to be maximized. The

* Corresponding author. Tel.: +65 6874 2562; fax: +65 6777 1434.
E-mail address: iseonghl@nus.edu.sg (H.L. Ong).

0965-9978/$ - see front matter © 2006 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2006.06.003

minimization version of the problem can be transformed into an equivalent maximization form (see [23]). The two objectives are to maximize the total sum of attribute-1 and attribute-2 of the items, respectively. The BiGAP could have many real world applications. For examples, in production planning, $-p_{ij}^1$ and $-p_{ij}^2$ represent the cost and time caused by assigning job j to machine i; in emergency service zone planning, $-p_{ij}^1$ and $-p_{ij}^2$ represent the time and cost caused by assigning zone j to service unit i; in vehicle routing, p_{ij}^1 and $-p_{ij}^2$ represent the profit and traveling cost caused by assigning vehicle i to serve customer j.

Given n items and m knapsacks, the BiGAP can be formulated as follows:

$$\begin{aligned}
\text{P1}: \quad \text{maximize} \quad & z_k = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^k x_{ij} \quad k = 1,2 \\
\text{subject to} \quad & \sum_{j=1}^{n} w_{ij}x_{ij} \leqslant c_i, \quad i \in \{1,\ldots,m\}, \\
& \sum_{i=1}^{m} x_{ij} = 1, \quad j \in \{1,\ldots,n\}, \\
& x_{ij} = 0 \text{ or } 1 \quad \text{for all } i,j,
\end{aligned}$$

where

$$x_{ij} = \begin{cases} 1 & \text{if item } j \text{ is assigned to knapsack } i, \\ 0 & \text{otherwise.} \end{cases}$$

p_{ij}^k attribute-k of item j if it is assigned to knapsack i
w_{ij} weight of item j if it is assigned to knapsack i
c_i capacity of knapsack i

Here, unlike the common practice, we do not assume that w_{ij}, c_i and p_{ij}^k are integral in that such assumptions are not necessary to our method. The number of variables in the above formulation is $(m \times n)$ and the number of constraints is $(m + n)$.

3. An effective approach to solving the BiGAP

In order to introduce our method, we need to use a few concepts including *nondominated solution*, *feasible outcome region*, *extreme nondominated solution* (ENS) and *ε-nondominated solution* (ε-NS solution, in short). The concepts of nondominated solution and feasible outcome region are generally well known. The definitions of ENS and ε-NS solution are given below [33].

Definition 1 (*Extreme nondominated solution*). An extreme nondominated solution (ENS) to a multiobjective integer programming problem is a nondominated solution with one objective value equal to its maximum possible value.

This concept is proposed to make use of the bounds and domination conditions to reduce the search space (see, e.g. [10]). For example, in Fig. 1 we illustrate the two ENSs A and B to a biobjective integer programming problem. The dots illustrate the boundary of the feasible outcome region. All nondominated solutions will be contained in the region

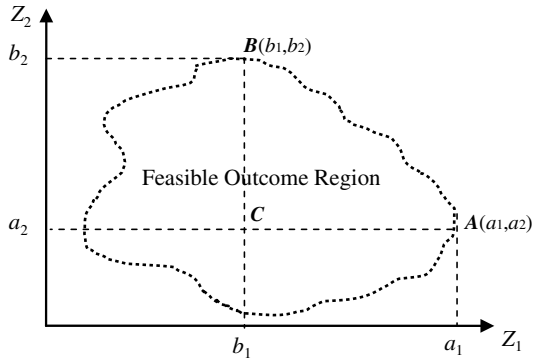

Fig. 1. An illustration of the two ENSs to a biobjective integer programming problem.

A–B–C. On the other hand, it is generally difficult to generate an exact nondominated solution to a multiobjective integer programming problem if the problem size is large. Thus it is desirable to have a measure to evaluate an approximate nondominated solution generated. This has led to the following definition.

Definition 2 (*ε-Nondominated solution*). In the outcome space, let $N(P)$ denote the set of nondominated solutions to an l-objective integer programming problem P and let s be a feasible solution to P. Then solution s is an ε-nondominated solution (ε-NS solution, in short) if there exists $z \in N(P)$ such that

$$\|z - s\|_q^{\lambda} \leqslant \varepsilon \cdot \|z - o\|_q^{\lambda}, \tag{1}$$

where o denotes the origin; ε is a positive value; $\|x-y\|_q^{\lambda} = \left[\sum_{i=1}^{l}(\lambda_i|x_i - y_i|)^q\right]^{\frac{1}{q}}$ denotes the weighted L_q-metric; q is a positive value or equal to ∞; The most commonly used q values include 1, 2 and ∞; $\lambda \in R^l$ is a nonnegative vector of weights; λ_i,x_i and y_i are the i th component of λ, x and y, respectively.

This concept is a natural extension of the ε-optimality in the context of single objective optimization. If a feasible solution s satisfies Condition (1), it means that it is within ε distance from the nondominated frontier. This measure can be used to evaluate the quality of a solution generated to a multiobjective integer programming problem.

In order to solve problem P1 effectively and minimize the unnecessary search, we first find the two ENSs to this problem. Let $A(a_1,a_2)$ and $B(b_1,b_2)$ be the two ENSs (see Fig. 1). By taking the two ENSs into consideration, problem P1 can be reformulated as follows:

$$\begin{aligned}
\text{P2}: \quad \text{maximize} \quad & z_k = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^k x_{ij} \quad k = 1,2 \\
\text{subject to} \quad & \sum_{j=1}^{n} w_{ij}x_{ij} \leqslant c_i, \quad i \in \{1,\ldots,m\}, \\
& \sum_{i=1}^{m} x_{ij} = 1, \quad j \in \{1,\ldots,n\},
\end{aligned}$$

$$z_1 = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^1 x_{ij} \geqslant b_1,$$
$$z_2 = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^2 x_{ij} \geqslant a_2,$$
$$x_{ij} = 0 \text{ or } 1 \quad \text{for all } i, j.$$

The advantage of this new formulation is that it reduces the search region effectively without losing any nondominated solution. This formulation restricts the search within the region A–B–C (see Fig. 1), which contains the nondominated set. For multiobjective integer programming problems, the main challenge is how to obtain very quickly a good approximation of the whole nondominated frontier [10]. In order to do this, we first partition the region A–B–C (see Fig. 1) into smaller subregions and focus on one of them at a time. Particularly, we divide interval $[b_1, a_1]$ into smaller subintervals. At each iteration, we focus on one of the small subintervals and solves the following problem P3 to generate one of the ε-NS solutions. The objective of this procedure is to generate some evenly distributed ε-NS solutions to the BiGAP.

$$\text{P3}: \quad \text{maximize} \quad z_2 = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^2 x_{ij}$$
$$\text{subject to} \quad \sum_{j=1}^{n} w_{ij} x_{ij} \leqslant c_i, \quad i \in \{1, \ldots, m\}, \tag{2}$$
$$\sum_{i=1}^{m} x_{ij} = 1, \quad j \in \{1, \ldots, n\}, \tag{3}$$
$$z_1 = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^1 x_{ij} \geqslant \text{Lower Goal}, \tag{4}$$
$$z_1 = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^1 x_{ij} \leqslant \text{Upper Goal}, \tag{5}$$
$$z_2 = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^2 x_{ij} \geqslant a_2, \tag{6}$$
$$x_{ij} = 0 \text{ or } 1 \quad \text{for all } i, j,$$

where Lower Goal and Upper Goal denote the lower and upper end of the subinterval under study, respectively.

The implementation details of this procedure are:

Step 1. Generate the two ENSs $A(a_1, a_2)$ and $B(b_1, b_2)$ to problem P1. Set the interval under study, [Start, End], to $[b_1, a_1]$.

Step 2. Ask the decision maker (DM) on how many ε-NS solutions he desires to be generated. Let t denote the number. Then divide the interval [Start, End] evenly into t smaller subintervals with length $= \delta$. That is,
$\delta = (\text{End} - \text{Start})/t$,
Lower Goal $=$ Start $+ i \times \delta$,
Upper Goal $=$ Start $+ (i+1)\delta$, $i = 0,1,\ldots,t-1$.

Step 3. For each subinterval generated, assign the corresponding values to Lower Goal and Upper Goal and solve problem P3 with the LPH algorithm to be described to generate an ε-NS solution.

Step 4. Is DM satisfied with the current set of generated ε-NS solutions? If yes, stop. If no, ask DM to identify the specific interval of interest, $[b^*, a^*]$, and the desired number, t^*, of ε-NS solutions in this interval. Go to Step 2 with Start $= b^*$, End $= a^*$, and $t = t^*$.

3.1. An LP-based heuristic for solving P3

The vast majority of heuristic approaches presented in the literature that aim at generating a good approximation to the nondominated set of multiobjective integer programming problems have been based on adapting to multiobjective context the classical metaheuristics including simulated annealing, tabu search, genetic algorithms, and ant colony; see [22,7,16,28,13,20,26,29,21,9,18]. However, very little work can be found on the performance of LP-based approaches [5]. In the following we propose an LP-based heuristic that is able to solve problem P3 efficiently to generate ε-NS solutions to problem P1.

Lemma 1. *The nondominated frontier of a biobjective linear programming problem, defined as the mapping $Z_2 = g(Z_1)$, is decreasing in the outcome space, as shown in Fig. 2.*

This is actually straightforward [33]. Now consider the LP relaxation problem of P3, we have the following Theorem 1.

Theorem 1. *Let S denote the number of split items in the optimal solution to the LP relaxation problem of P3, F the number of fractional variables, and K_c the number of knapsacks fully occupied. Then we have: $F \leqslant S + K_c + 1$.*

Proof. The LP relaxation problem of P3 can be reformulated in the augmented form as follows:

$$\text{P3a}: \quad \text{maximize} \quad z_2 = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^2 x_{ij}$$
$$\text{subject to} \quad \sum_{j=1}^{n} w_{ij} x_{ij} + s_i = c_i, \quad i \in \{1, \ldots, m\}, \tag{2a}$$
$$\sum_{i=1}^{m} x_{ij} = 1, \quad j \in \{1, \ldots, n\}, \tag{3a}$$
$$\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^1 x_{ij} - s_{m+1} = \text{Lower Goal}, \tag{4a}$$
$$\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^1 x_{ij} + s_{m+2} = \text{Upper Goal}, \tag{5a}$$
$$\sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}^2 x_{ij} - s_{m+3} = a_2, \tag{6a}$$
$$x_{ij} \geqslant 0 \quad \text{for all } i, j$$
$$s_i, s_{m+1}, s_{m+2} \text{ and } s_{m+3} \geqslant 0.$$

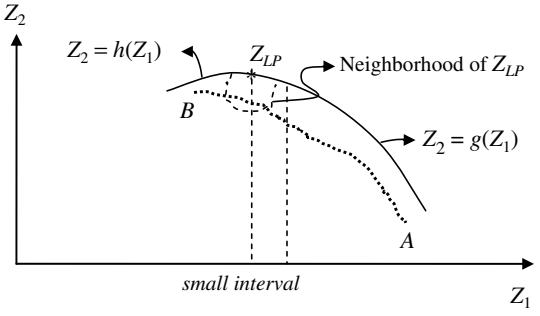

Fig. 2. An illustration of the LP-based heuristic.

Let us consider the number of nonzero variables, including all x_{ij}, s_i, s_{m+1}, s_{m+2} and s_{m+3}, in the optimal solution to problem P3a.

In the first case, if the small subinterval under consideration is completely covered by the nondominated frontier, $Z_2 = g(Z_1)$, then from Lemma 1, we observe that in the optimal solution s_{m+1} is always equal to 0 to make z_2 maximal. This implies at the same time that Constraint (5) is actually redundant and can be removed. As a result, the total number of nonzero variables equals $(n - S + F) + (m - K_c + 2)$. The first term corresponds to the nonzero assignment variables x_{ij}, and the second term corresponds to the surplus or slack variables. On the other hand, the total number of constraints is $(n + m + 3)$. Consequently, we have

$$n - S + F + m - K_c + 2 \leqslant n + m + 3$$

that is, $F \leqslant S + K_c + 1$.

In the second case, the small subinterval under consideration is completely covered by the mapping $Z_2 = h(Z_1)$, which is obviously non-decreasing, as shown in Fig. 2. If $Z_2 = h(Z_1)$ is strictly increasing, then analogously s_{m+2} must be equal to 0 to make z_2 maximal, and Constraint (4) becomes redundant. Eventually, we can also obtain $F \leqslant S + K_c + 1$.

The third case is when both s_{m+1} and s_{m+2} will be nonzero in the optimal solution to problem P3a. This could happen, although rarely, on two occasions. The first one is when the small subinterval under consideration is completely covered by $Z_2 = h(Z_1)$ when it is constant; the second one is when the small subinterval under consideration is partially covered by $Z_2 = g(Z_1)$ and partially by $Z_2 = h(Z_1)$. In this case, the number of total non-zero variables is $(n - S + F) + (m - K_c + 3)$, and the total number of constraints is still $(n + m + 3)$. Consequently, we get $F \leqslant S + K_c < S + K_c + 1$.

It is evident that this proof can be extended to other close variants of the GAP. □

Corollary 1. *From Theorem 1, we can directly derive the following two inequalities*:

1. $S \leqslant m + 1$;
2. $F \leqslant 2m + 2$.

Proof. It is obvious that $F \geqslant 2S$. Thus we have $S \leqslant K_c + 1$. In addition, we know $K_c \leqslant m$, so the corollary follows. □

The GAP and its variant problem P3 have two characteristics. The first one is that relatively few variables having fractional values exist in the optimal solution to their corresponding LP problems. Benders and van Nunen [3] and Trick [27] have proved that in the optimal solution to the LP relaxation of a GAP the number of split items (value proportional to the number of fractional variables) is less than or equal to the number of fully occupied knapsacks. Corollary 1 has shown that the optimal solution to the LP relaxation of problem P3 has at most $(2m + 2)$ fractional variables. The second characteristic that has been observed empirically is that, the optimal solution to the integer programming problem is quite similar to that to its LP relaxation problem. This characteristic is related to the first one to some extent. Furthermore, the gap between the optimal solution to problem P3 and that to the LP problem tends to be small when the number of items is large compared to the number of knapsacks and when the capacity constraints are loose in the sense that for each knapsack the ratio of the total weight of the assigned items to the capacity of the knapsack is relatively smaller than 1.

These characteristics make the LPH algorithm very efficient. The LPH algorithm consists of two basic steps. The first step is to solve the LP relaxation of problem P3 and get the optimal solution Z_{LP}. The second step is to search the neighborhood of Z_{LP} to find one of the ε-NS solutions. This process is illustrated in Fig. 2. Here the neighborhood of Z_{LP} comprises all feasible solutions to problem P1 (not P3) that are very close to Z_{LP}. The LPH algorithm searches the neighborhood of Z_{LP} by solving a subproblem P4 described as follows:

Let

$C3$ the set of constraints (3) in problem P3

C_G the set of constraints in $C3$, in which all the non-zero-coefficient variables have either value 0 or 1 in the optimal solution to the LP relaxation of problem P3

$C_H = C3 - C_G$ is the complementary set of C_G

V the set of all decision variables, x_{ij}

H the set consisting of all nonzero-coefficient variables in constraint set C_H

$G = V - H$ is the complementary set of H

Then the subproblem P4 is formulated as follows:

$$\text{P4}: \quad \text{maximize} \quad z_2' = \sum_{x_{ij} \in H} p_{ij}^2 x_{ij}$$

$$\text{subject to} \quad \sum_{j: x_{ij} \in H} w_{ij} x_{ij} \leqslant c_i - b_i, \quad i \in \{1, \ldots, m\}, \tag{7}$$

$$\sum_{i=1}^{m} x_{ij} = 1, \quad j \in C_H, \tag{8}$$

$$x_{ij} = 0 \text{ or } 1,$$

where

$$b_i = \sum_{x_{ij} \in G} w_{ij} x_{ij}, \quad x_{ij} \text{ take values from } Z_{\mathrm{LP}}. \tag{9}$$

Let Z_{sub} represent the optimal solution to subproblem P4. Then we have the following Theorem 2.

Theorem 2. *A solution s to problem P1, of which the variables belonging to H receive their values from Z_{sub}, if it exists, and the variables belonging to G receive their values from Z_{LP}, is a feasible solution.*

Proof. The proof is straightforward. □

Let n_{v} denote the number of variables in subproblem P4 and n_{c} denote the number of constraints in subproblem P4. Then it is easy to show that

$$n_{\mathrm{v}} = m \cdot S \leqslant m(K_c + 1) \leqslant m(m+1),$$
$$n_{\mathrm{c}} = m + S \leqslant m + K_c + 1 \leqslant 2m + 1.$$

From our experience, when the number of items n is much greater than the number of knapsacks m and the capacity constraints are tight, the number of split items S in the optimal solution to the LP relaxation problem of P3 (i.e. P3a) is usually close to its upper bound $(m+1)$, and the number of fractional variables F is also close to its upper bound $(2m+2)$. We can see that the size of subproblem P4 is much smaller than that of problem P3, especially when the number of items n is much greater than the number of knapsacks m.

It is noted that subproblem P4 is also a GAP, which is described as assigning S items to m knapsacks subject to the resulting capacity constraints. Consequently, a possible way to solve subproblem P4 is to repeatedly solve the LP relaxation of the resulting subproblems. Heuristics based on this technique for solving the GAP can be found in [27]. However, repeatedly employing the LP relaxation could increase the likelihood of infeasibility of the final solution when the capacity constraints are tight. And also, the gap between the optimal solution and the generated solution could be large. Furthermore, the number of split jobs may decrease slowly. On the other hand, we have shown that in subproblem P4 the number of items S is at most $(m+1)$. This makes the subproblem much easier to solve, compared to the original problem which in real world usually has much greater number of items than number of knapsacks. As a result, we directly solve subproblem P4 in the LPH algorithm. Many exact and heuristic algorithms could solve subproblem P4 efficiently. In this study, we employ the ILOG CPLEX 7.5 to solve it. It takes CPLEX little time to solve these subproblems in our experiments. It is possible that the ε-NS solutions generated by the LPH algorithm from the neighborhood of contiguous Z_{LP}'s coincide; however, from our experience this chance is very slim.

3.2. Evaluation of an ε-NS solution

In the LPH algorithm a feasible solution to problem P1 is evaluated in the following way. We use $q = \infty$ for the unweighted L_q-metric in the definition of ε-NS solution. Let $d(z_1, z_2)$ denote the distance between two solution points z_1 and z_2 in the outcome space. Then we have

$$d(z_1, z_2) = \|z_1 - z_2\|_\infty = \max_i \{|z_{1i} - z_{2i}|\}, \quad i \in \{1, \ldots, l\},$$

where l is the number of objectives.

Let Z_{LP} represent the optimal solution to the LP relaxation problem of P3, and s represent a feasible solution to problem P1 generated by the LPH algorithm from the neighborhood of Z_{LP}. Then according to the definition of ε-NS solution, if there exists a nondominated solution to problem P1, Z^*, which satisfies $d(s, Z^*) \leqslant \varepsilon \cdot d(Z^*, o)$, then s is an ε-NS solution. In other words, the ε value associated with s can be calculated by the following formula:

$$\varepsilon = \frac{d(s, Z^*)}{d(Z^*, o)}. \tag{10}$$

However, $Z^*$is usually unknown. Hence we have to use some estimation. In the actual implementation of the LPH algorithm, we use the following (11) to evaluate a solution s, replacing (10)

$$\varepsilon = \frac{d(s, Z_{\mathrm{LP}})}{d(Z_{\mathrm{LP}}, o)}. \tag{11}$$

3.3. Search strategy

Let us first examine the characteristics of the nondominated frontier of the LP relaxation problem of P1. We have shown in Lemma 1 that the nondominated frontier of a biobjective linear programming problem is decreasing. In Fig. 3, we depict the nondominated frontier of the LP relaxation of a typical BiGAP instance of size (500×30). We can see that as the value of Objective1 increases, the value of Objective2 first decreases slowly and then decreases faster and faster when the value of Objective1 approaches its maximum value. Actually, this is a typical case. We call the portion of the nondominated frontier of the LP relaxation of a BiGAP where the value of Objective2 starts to decrease rapidly as the value of Objective1 increases as the *turning segment*, as illustrated in Fig. 3. The empirical position of this turning segment usually occurs around the 70–90% portion of the whole interval $[b_1, a_1]$.

Let T represent a point on the turning segment, as shown in Fig. 3. We call T the *turning point*. In the search strategy of the proposed method, Objective2 is maximized in subproblem P4 for the interval from B to T, while Objective1 is maximized for interval from T to A in a new subproblem P5 that is defined as

$$\text{P5}: \quad \text{maximize} \quad z_1' = \sum_{x_{ij} \in H} p_{ij}^1 x_{ij}$$

subject to Constraints (7)–(9).

Actually, this is an improved search strategy based on an old one where Objective2 was maximized for the whole interval from B to A. What was discovered from the old

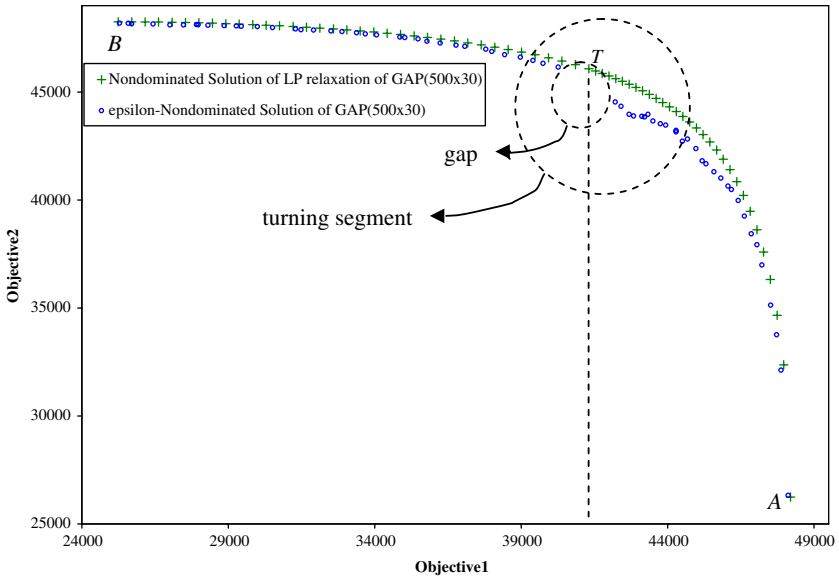

Fig. 3. ε-NS solutions generated by the LPH.

strategy is that the ε-NS solutions generated to the right of T get farther and farther away from the nondominated frontier of the LP problem as the subinterval approaches A. This is because the portion $T-A$ of the nondominated frontier of the LP problem is almost *vertical*. Consequently, a strategy based on maximizing Objective2 in subproblem P4 will not make the searching move toward the nondominated frontier. However, the search strategy of maximizing Objective1 in P5 will do this job. Such a mechanism of the search strategy is illustrated in Fig. 4. Actually, this strategy is quite consistent with what is suggested in the concluding section of [29].

It is noticed that the distance between the ε-NS solutions increases from T to A. This is also due to the "turning" of the nondominated frontier from almost horizontal to almost vertical, since we have chosen to divide the abscissa into equal length intervals. On the other hand, if one replaces Constraints (4) and (5) in problem P3 by two corresponding constraints on Objective2 (i.e. lower and upper ends of the interval over Objective2), replace Constraint (6) by $z_1 \geqslant b_1$, switch the objective function to maximize z_1, and divide the axis of Objective2 into equal length intervals for portion from T to A, then the ε-NS solutions generated for portion $T-A$ would look similar to those generated for portion $B-T$ in the current case. This is actually straightforward.

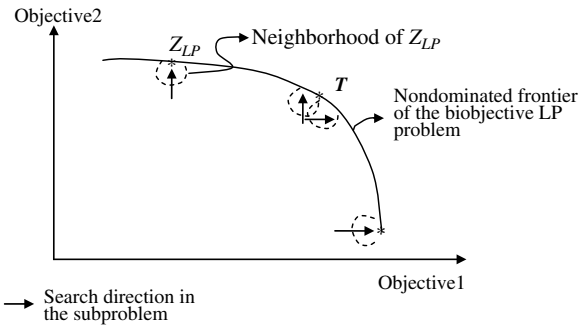

Fig. 4. An illustration of the search strategy.

For example, the ε-NS solutions generated by the LPH algorithm to a typical problem instance of size (500×30) are plotted in Fig. 3. We observed that there is always a small 'gap' on the approximate frontier of the BiGAP occurring around the turning point T. In other words, it seems difficult to generate some ε-NS solutions around the turning point using this strategy. This is due to the sudden shift of the search direction in the subproblem from Objective2 to Objective1, as illustrated in Fig. 4.

In order to improve the results and fill up the 'gap', we modify the search strategy by artificially creating an 'overlap' at around the turning point. As illustrated in Fig. 5, subproblem P4 is employed for interval from B to T_2, while subproblem P5 is employed for interval from T_1 to A. In this case, the quality of the generated approximate frontier improves. Fig. 5 depicts the generated ε-NS solutions to a typical problem instance of size (500×50). We see that the 'gap' has disappeared.

In addition, from our experiments we found that quite similar results can be achieved if we replace subproblem P5 by the following subproblem P6.

$$\text{P6}: \quad \text{maximize} \quad z'_{12} = \sum_{x_{ij} \in H} p^1_{ij} x_{ij} + \sum_{x_{ij} \in H} p^2_{ij} x_{ij}$$

$$\text{subject to Constraints (7)–(9).}$$

In this case, the searching in the subproblem will be in the 45° direction. Obviously, we can further generalize the formulation of the objective function in the subproblem, such as by using the weighted sum of the two objectives given that the weights are properly chosen according to the curvature of the nondominated frontier of the LP problem.

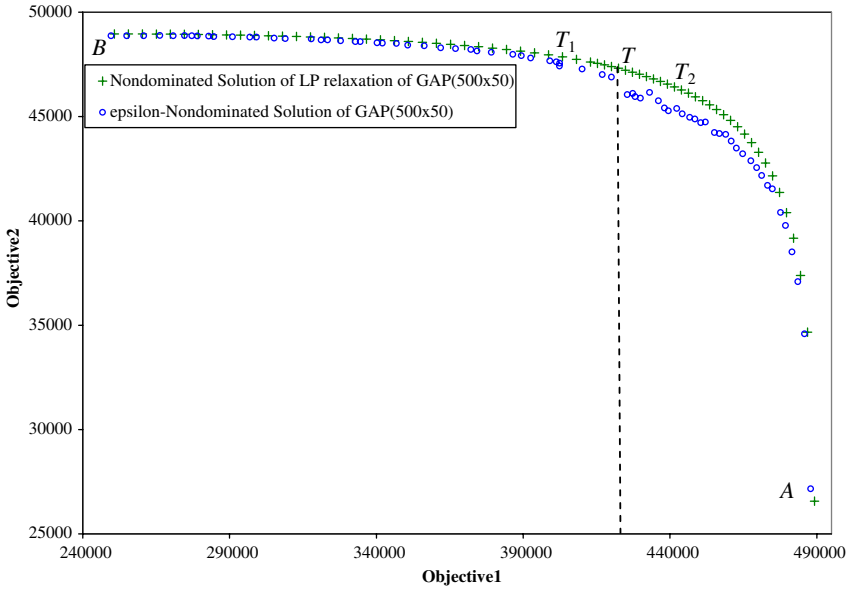

Fig. 5. ε-NS solutions generated by the LPH using the modified search strategy.

4. Computational experiments and results

Unlike the single objective GAP, for which there are many existing benchmark test problems, no benchmark test problems are available for the BiGAP as this is the first time to our knowledge that the GAP is studied from a multiobjective perspective. Consequently, we use problem instances randomly generated by the following method in this study. For each problem instance, the attribute-1's of the items p^1_{ij} are randomly generated from a uniform distribution over interval $[1,1000]$, the attribute-2's of the items p^2_{ij} are randomly generated from interval $[1,100]$, and the weights of the items w_{ij} are also randomly generated from interval $[1,100]$. We have conducted extensive experiments, which show that the proposed method is robust to the length of interval from which uniform data are generated. For example, if both p^1_{ij} and p^2_{ij} are generated from $[1,1000]$, the results are quite similar. Following the method of Martello and Toth [23], the capacities of the knapsacks are calculated according to the following formula:

$$c_i = 0.8\sum_{j=1}^{n}\frac{w_{ij}}{m} \quad \text{for } i = 1,\ldots,m.$$

We have selected the number of items $\{200,400,600,800,1000\}$ and the number of knapsacks $\{10,20,30,40,50\}$ as components of the problem sizes. For each combination of them we have randomly generated 20 problem instances. For each problem instance, the number of generated ε-NS solutions is 71. All experiments were conducted on a Pentium III 868 MHz PC. The computational results are summarized in Table 1. Each entry of the ε value is the average over the 71 solutions, and in turn over the 20 problem instances. It is observed that the ε value of solutions from the turning segment is greater than those from the two tails of the frontier. The ε value is a measure estimating the average distance from the generated approximate frontier to the true nondominated frontier. For instance, if the ε value is 0.0120, it means that the distance from the generated approximate frontier to the true nondominated frontier is around 1.2% of the distance from the true nondominated frontier to the origin on the average. Each entry of the time taken to generate these 71 ε-NS solutions is the average over the 20 problem instances.

Table 1
Summary of the computational results (based on 71 ε-NS solutions)

Number of knapsacks M	Average results	Number of items n				
		200	400	600	800	1000
10	ε value	0.0120	0.0062	0.0042	0.0031	0.0025
	Time (s)	1.36	3.34	6.05	9.57	14.01
20	ε value	0.0185	0.0095	0.0064	0.0049	0.0039
	Time (s)	3.02	8.55	16.8	28.31	45.52
30	ε value	0.0271	0.0143	0.0097	0.0074	0.0059
	Time (s)	5.38	16.01	32.35	59.67	96.31
40	ε value	0.0359	0.0191	0.0130	0.0100	0.0079
	Time (s)	8.06	24.61	56.97	110.74	177.08
50	ε value	0.0426	0.0241	0.0162	0.0124	0.0098
	Time (s)	11.25	41.38	89.86	167.87	276.85

From Table 1, we see that given the value of n the ε value increases as the number of knapsacks m increases. This implies that the quality of the generated approximate frontier decreases as the number of knapsacks increases. On the contrary, given the value of m the ε value decreases as the number of items n increases. This implies that the quality of the generated approximate frontier increases as the number of items increases. Furthermore, it is noted that the ε value is closely related to the ratio of the number of

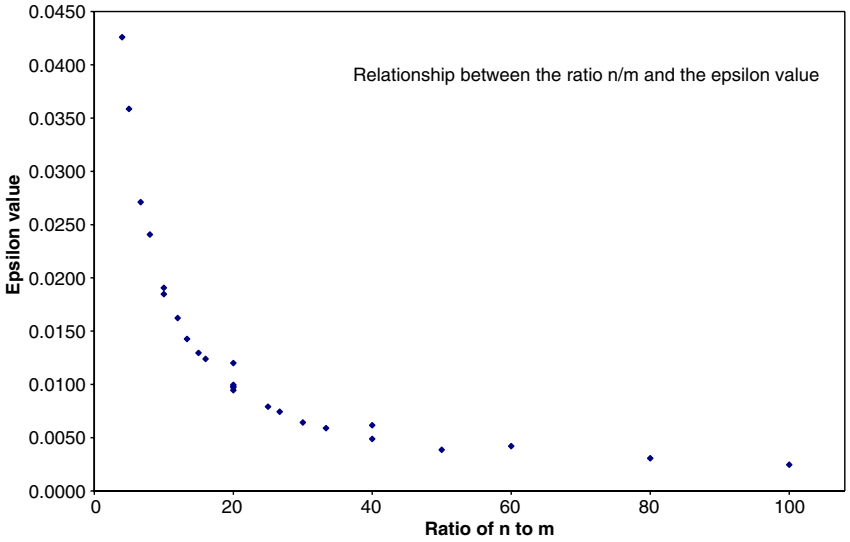

Fig. 6. Relationship between the ratio of n to m and the ε value.

items to the number of knapsacks, i.e. n/m. In Fig. 6, we plot the 25 data points from Table 1. It is evident that the ε value decreases very fast as the ratio of n/m increases. In real world problems, the ratio of n/m is usually large, and thus the ε value would be small, which means the quality of the generated approximate frontier would be good. A theoretical justification for this relationship between the ε value and n/m may not be easy, but it can be intuitively explained in the following way. Suppose a scenario where we fix m and increase n, which means the ratio n/m increases. From Corollary 1, the size of the subproblem (P4 or P5) almost remains the same, as it mainly depends on m. Put it loosely, the effect of the optimal solution to the LP problem, Z_{LP}, on the final integral solution to problem P3 will increase, while the effect of the optimal solution to the subproblem, Z_{sub}, on that almost remains the same. Consequently, the larger is n, the relatively closer is Z_{LP} to the final integral solution to problem P3, which implies the smaller is the ε value.

On the other hand, from Table 1 we see that the time taken to generate those ε-NS solutions increases as the problem size increases, both in the dimension of n and in the dimension of m. It is noted that the times taken are quite short, even if the problem size is very large. These computational experiments showed that the proposed method is able to generate a good approximation to the nondominated frontier of a BiGAP efficiently, especially when the ratio of n to m is large.

5. Conclusions

In this paper, the GAP has been examined from a multiobjective point of view to accommodate some real world situations where more than one objective is involved. In particular, we studied the biobjective GAP (BiGAP). An efficient approach, the core of which is an LP-based heuristic (LPH), has been proposed. It is able to generate a good approximation to the nondominated frontier of a BiGAP efficiently, even if the problem size is very large. This approach has a few advantages over one that employs a posteriori elicitation of preference information (see [11]). Firstly, one only needs to deal with single objective optimization problems, for which many effective algorithms and commercial solution packages like ILOG CPLEX are available. Secondly, the ε-NS solutions are generated according to the DM's actual necessity, and thus the DM will not be overwhelmed by a huge number of candidate solutions, especially when these solutions are presented to him graphically. Thirdly, this approach requires little input from the DM. He is only required to give the number of solutions to be generated and specify the interval of the nondominated frontier that he is most interested in. This is much easier to the DM than specifying some reference directions or some reference points or making many pairwise comparisons. Finally, the proposed approach is readily extended to general multiobjective cases. However, it is more important to convey the idea and approach clearly and readably in the first place in order to extend it further to more general cases. The biobjective case best serves this purpose, as it allows us to illustrate the idea and approach graphically.

References

[1] Alves MJ, Climaco J. An interactive reference point approach for multiobjective mixed-integer programming using branch-and-bound. Eur J Oper Res 2000;124:478–94.
[2] Amini MM, Racer M. A hybrid heuristic for the generalized assignment problem. Eur J Oper Res 1995;87:343–8.
[3] Benders JF, van Nunen JAEE. A property of assignment type mixed integer linear programming problems. Oper Res Lett 1983;2(2):47–52.
[4] Cattrysse D, Degraeve A, Tistaert J. Solving the generalized assignment problem using polyhedral results. Eur J Oper Res 1998;108:618–28.

[5] Cattrysse DG, Wassenhove LNV. A survey of algorithms for the generalized assignment problem. Eur J Oper Res 1992;60:260–72.
[6] Chu PC, Beasley JE. A genetic algorithm for the generalized assignment problem. Comput Oper Res 1997;24(1):17–23.
[7] Czyzak P, Jaszkiewicz A. Pareto simulated annealing – a metaheuristic technique for multiple-objective combinatorial optimization. J Multi-Criteria Decis Anal 1998;7:34–47.
[8] Diaz JA, Fernandez E. A tabu search heuristic for the generalized assignment problem. Eur J Oper Res 2001;132:22–38.
[9] Doerner K, Gutjahr WJ, Hartl RF, Strauss C, Stummer C. Pareto ant colony optimizaiton: a metaheuristic approach to multiobjective portfolio selection. Ann Oper Res 2004;131:79–99.
[10] Ehrgott M, Gandibleux X. A survey and annotated bibliography of multiobjective combinatorial optimization. OR Spektrum 2000;22: 425–60.
[11] Evans GW. An overview of techniques for solving multiobjective mathematical programs. Manage Sci 1984;30(1):1268–82.
[12] Fu Y, Diwekar UM. An efficient sampling approach to multiobjective optimization. Ann Oper Res 2004;132:109–34.
[13] Gandibleux X, Freville A. Tabu search based procedure for solving the 0–1 multiobjective knapsack problem: the two objective case. J Heuristics 2000;6:361–83.
[14] George JW, Revelle CS. Bi-objective median subtree location problems. Ann Oper Res 2003;122:219–32.
[15] Haddadi S, Ouzia H. Effective algorithm and heuristic for the generalized assignment problem. Eur J Oper Res 2004;153:184–90.
[16] Hapke M, Jaszkiewicz A, Slowinski R. Interactive analysis of multiple-criteria project scheduling problems. Eur J Oper Res 1998; 107:315–24.
[17] Higgins AJ. A dynamic tabu search for large-scale generalized assignment problems. Comput Oper Res 2001;28:1039–48.
[18] Jaszkiewicz A. A comparative study of multiple-objective metaheuristics on the bi-objective set covering problem and the Pareto memetic algorithm. Ann Oper Res 2004;131:135–58.
[19] Junker U. Preference-based search and multi-criteria optimization. Ann Oper Res 2004;130:75–115.
[20] Karasakal EK, Koksalan M. A simulated annealing approach to bicriteria scheduling problems on a single machine. J Heuristics 2000;6:311–27.
[21] Lourenco HR, Paixao JP, Portugal R. Multiobjective metaheuristics for the bus driver scheduling problem. Transport Sci 2001;35(3): 331–43.
[22] Marett R, Wright M. A comparison of neighborhood search techniques for multi-objective combinatorial problems. Comput Oper Res 1996;23(5):465–83.
[23] Martello S, Toth P. Knapsack problems: algorithm and computer implementations. New-York: John Wiley & Sons; 1990.
[24] Osorio MA, Laguna M. Logic cuts for multilevel generalized assignment problems. Eur J Oper Res 2003;151:238–46.
[25] Ross GT, Soland RM. Modeling facility location problems as generalized assignment problems. Manage Sci 1977;24:345–57.
[26] Teghem J, Tuyttens D, Ulungu EL. An interactive heuristic method for multiobjective combinatorial optimization. Comput Oper Res 2000;27:621–34.
[27] Trick MA. A linear relaxation heuristic for the generalized assignment problem. Nav Res Logist 1992;39:137–51.
[28] Ulungu EL, Teghem J, Fortemps Ph, Tuyttens D. MOSA method: a tool for solving multiobjective combinatorial optimization problems. J Multi-Criteria Decis Anal 1999;8:221–36.
[29] Viana A, Sousa JP. Using metaheuristics in multiobjective resource constrained project scheduling. Eur J Oper Res 2000;120:359–74.
[30] Visee M, Teghem J, Pirlot M, Ulungu EL. Two-phase method and branch and bound procedures to solve the bi-objective knapsack problem. J Global Optim 1998;12:139–55.
[31] Wilson JM. An algorithm for the generalized assignment problem with special ordered sets. J Heuristics 2005;11:337–50.
[32] Yagiura M, Ibaraki T, Glover F. A path relinking approach with ejection chains for the generalized assignment problem. Eur J Oper Res 2005;169:548–69.
[33] Zhang CW, Ong HL. Solving the biobjective zero-one knapsack problem by an efficient LP-based heuristic. Eur J Oper Res 2004;159:545–57.