Applied Mathematics and Computation 218 (2012) 6526–6535

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

The equilibrium generalized assignment problem and genetic algorithm

Linzhong Liu [a,*], Haibo Mu [a], Yubo Song [b], Haiyan Luo [a], Xiaojing Li [a], Fang Wu [a]

[a] School of Traffic & Transportation, Lanzhou Jiaotong University, Lanzhou 730070, China
[b] Mechatronic Technology Institute, Lanzhou Jiaotong University, Lanzhou 730070, China

ARTICLE INFO

Keywords:
Assignment problem
Equilibrium optimization
Generalized assignment problem
Genetic algorithm

ABSTRACT

The well-known generalized assignment problem (GAP) is to minimize the costs of assigning n jobs to m capacity constrained agents (or machines) such that each job is assigned to exactly one agent. This problem is known to be NP-hard and it is hard from a computational point of view as well. In this paper, follows from practical point of view in real systems, the GAP is extended to the equilibrium generalized assignment problem (EGAP) and the equilibrium constrained generalized assignment problem (ECGAP). A heuristic equilibrium strategy based genetic algorithm (GA) is designed for solving the proposed EGAP. Finally, to verify the computational efficiency of the designed GA, some numerical experiments are performed on some known benchmarks. The test results show that the designed GA is very valid for solving EGAP.

1. Introduction

The classical assignment problem (AS) is one of the widely applied and extensively studied combinatorial optimization problems. Its applications include the graph bandwidth problems, telecommunications network design, load balancing, data association problems and multidimensional assignment problems [21]. It has been extended to various versions (*e.g.*, the multi-index AS, the quadratic AS [21] and the nonlinear assignment problem [23]) and lots of the extended versions and applications can be found in [21,23]. The generalized assignment problem (GAP) is another important extended version of the classical AS.

The GAP is a well-known NP-hard combinatorial optimization problem [27]. It considers a situation in which n jobs have to been processed by m agents. The agent which is the generic term for machine, worker and so on in real systems has a capacity expressed in terms of resource which is consumed by jobs processing. The GAP is to find minimum cost assignment of jobs to agents such that each job is assigned to exactly one agent subject to the agents' available capacity. The GAP has been widely applied in real problems, for example, assignment jobs to computers in computer network, storage space allocation, design of communication network with node capacity constrains, manufacturing system, service system, facility location problem, logistics network design, and so on [25].

Motivated by practical applications, various generalizations of GAP have been proposed, for example, the multi-level generalized assignment problem by Laguna et al. [14]; the multi-objective GAP by Subtil et al. [31] and the bi-objective GAP by Zhang and Ong [37]; the dynamic multi-resource generalized assignment problem by Shtub and Kogan [30]; the generalized multi-assignment problem by Park et al. [22]; the multi-resource generalized assignment problem with additional constraints by Privault and Herault [24]; the generalized quadratic assignment problem (GQAP) by Hahn et al. [11], where a special cost, say *interactive cost*, is taken into consideration and the interactive cost is usually associated with two job-agent pairs. In this research, follows from the practical point of view, we proposed the equilibrium generalized assignment problem.

* Corresponding author.
E-mail address: liulinzhong@tsinghua.org.cn (L. Liu).
URL: http://orsc.edu.cn/~lzliu (L. Liu).

0096-3003/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2011.12.025

The research in this paper is mainly motivated by following special jobs assignment, in which the equilibrium is required to evaluate the system, in a real world. In the railway system of China, to avoid the accident caused by the fatigue of the worker, the railway law of China regulates that the working time of the workers or locomotives (the driver) in railway system must be as equilibrium as possible, where the jobs, for instance, of locomotives consisting of train disintegration operations, shunting operations, shunting operation for taking-out and placing-in wagons and transfer wagons to another yard and so on in a shift plan. The related problem of this situation is to find such a jobs assignment subject to the agents' available capacity that the difference between the maximum total cost and the minimum total cost consumed at single agent is minimum. We call this problem as *equilibrium generalized assignment problem* (EGAP). Furthermore, in some situations in real system we need to consider the problem to find the assignment in the optimal equilibrium assignments such that the costs is minimum.

The main contributions of this paper are summarized as follows. Firstly, follows from practical point of view in real systems, we proposed the equilibrium generalized assignment problem (EGAP) and the equilibrium constrained generalized assignment problem (ECGAP), respectively. Finally, a heuristic equilibrium strategy based genetic algorithm (GA) is designed for solving the proposed EGAP.

This paper is organized as follows: firstly, the research literatures with respect to GAP and its variations are surveyed in Section 2. Secondly, the equilibrium generalized assignment problem and the equilibrium constrained generalized assignment problem are proposed in Section 3. To solve the EGAP, a genetic algorithm is designed in Section 4. Finally, the designed GA is tested with some known benchmarks in Section 5.

2. Literature review

The GAP was firstly proposed by Ross and Soland [25] and is inspired by real-life problems such as assigning jobs to computer networks [3], fixed charge plant location where customer requirements must be satisfied by single plant [9] and the single sourcing problem [8]. Other applications which have been studied are the routing problem [8] and the p-median problem [26].

Various approaches can be found to solve this problem and some of the earlier algorithms are surveyed by Cattrysse and Van Wassenhove [5]. Due to its interest, this problem has been studied extensively from an algorithmic point of view. The employed solving approaches include the exact algorithm, the heuristic approach, the Lagrangean related method, the evolution algorithm (EA) and so on. Some of the recent approaches are surveyed as follows.

Among recent exact algorithms for GAP are the branch-and-price algorithm by Savelsbergh [28], the cutting plane algorithm by Avella et al. [2] and the branch-and-cut algorithm by Nauss [18], where exact optimal solutions to many benchmark instances with up to 200 jobs and 20 agents were obtained by Nauss [18]. Among various heuristic and meta-heuristic algorithms developed for GAP are the combination of the greedy method and local search by Martello and Toth [15], the approximation algorithm [29,7], the heuristic algorithm [10], the hybrid heuristic [1], the ejection chain approach [34], the path relinking approach with ejection chains [35], the neighborhood search algorithm [36,16], the novel and effective integer optimization approach [12]. Furthermore, the Lagrangean relaxation approach is employed for solving GAP, for example, the Lagrangean dual-based branch-and-bound algorithm [22], Lagrangean relaxation guided problem space search heuristics [13], Lagrangean/surrogate relaxation [17] and improved Lagrangean decomposition approach [4].

As a valid tool for solving those in the set of NP-hard problems, the evolutionary algorithm (EA) has been extensively applied to find an approximate solution of the optimization problems. For the application of EA in solving GAP, there have been some efficient attempts. For example, the simulated annealing and tabu search [19], the genetic algorithm [33,6], the differential evolution algorithms [32], the bees algorithm [20] and so on.

To our knowledge, there is not any attempt about the algorithm for EGAP. In this paper, to solve the proposed EGAP, a heuristic equilibrium strategy based genetic algorithm is developed.

3. Some notations and the mathematical models

Firstly, let us recall some preliminary notations. Throughout this paper, all networks are assumed to be the directed and simple complete bipartite network which is usually denoted as $G = (V_1, V_2)$, where the vertices set V of G is defined as $V = V_1 \bigcup V_2$ and the arcs set E is defined as $E = \{(u, v) | u \in V_1, v \in V_2\}$. For convenience, we employ $E(G)$ and $V(G)$ to denote the arcs set and the vertices set of the network G, respectively. For a subset S of E or V, $G[S]$ denotes the induced subnetwork of S.

Assume that $I = \{1, 2, \ldots, n\}$ is the set consisting of n jobs and $J = \{1, 2, \ldots, m\}$ is that consisting of m agents. The related network of the extended versions of GAP in this paper can be denoted as $G = (I, J)$. An arc $(s, t) \in E(G)$, $s \in I$, $t \in J$ implies that the job s can be assigned to the agent t. For an arc $(s, t) \in E(G)$, an index or ordering number $(t - 1)n + s$ is assigned to the arc (s, t). So an arc can also be denoted by its index in some places in the following process.

Provided that weight ξ_{ij} denotes the cost or time of the arc (i, j), $i \in I, j \in J$. The resource consumption associated with job-agent pair (i, j) is denoted as a_{ij} and the available resource at agent $j \in J$ is b_j. Let

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

The vector consisting of x_{ij}, $i \in I, j \in J$ is denoted as $\mathbf{x}$. Obviously, a job assignment is completely characterized by the vector $\mathbf{x}$. Then the mathematical model of the GAP [27] can be formulated as follows

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m}\xi_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{n}x_{ij} \geqslant 1, \quad j \in J, \\
& \sum_{i=1}^{n}a_{ij}x_{ij} \leqslant b_j, \quad j \in J, \\
& \sum_{j=1}^{m}x_{ij} = 1, \quad i \in I, \\
& x_{ij} = 0 \quad \text{or } 1,
\end{aligned} \tag{1}$$

where the first constraint indicates that at least one job is assigned to agent j and the third constraint indicates that each job i is assigned to exactly one agent; the second constraint denotes the resource limitation of agent $j \in J$. The GAP (1) is known to be NP-hard (NPH) [27] and judging the existence of a feasible solution for problem (1) is NP-complete (NPC) [35].

For a given solution $\mathbf{x}$, notation $g_j(\mathbf{x}) = \sum_{i=1}^{n}\xi_{ij}x_{ij}$ is employed to denote the overall costs consumed at agent $j \in J$. For convenience, two notations are defined as follows

$$\begin{aligned}
& f_{\max}(\mathbf{x}) = \max\{g_j(\mathbf{x}) | j \in J\}, \\
& f_{\min}(\mathbf{x}) = \min\{g_j(\mathbf{x}) | j \in J\}.
\end{aligned} \tag{2}$$

As stated previously, the associated costs $g_j(\mathbf{x})$, $j \in J$ are usually required to be equilibrium in some real systems. Here we employ an index $f(\mathbf{x}) = f_{\max}(\mathbf{x}) - f_{\min}(\mathbf{x})$ to measure the equilibrium of a solution $\mathbf{x}$. It is clear that $f(\mathbf{x}) = 0$ if and only if

$$g_j(\mathbf{x}) = \frac{1}{m}\sum_{i=1}^{n}\sum_{k=1}^{m}\xi_{ik}x_{ik}, \quad j \in J.$$

The related problem is called as *equilibrium generalized assignment problem* (EGAP) in which the equilibrium measure $f(\mathbf{x}) = f_{\max}(\mathbf{x}) - f_{\min}(\mathbf{x})$ is required to be minimum. Clearly, the smaller the $f(\mathbf{x})$, the more equilibrium the indices $g_j(\mathbf{x})$, $j \in J$. In the EGAP, the weight ξ_{ij} of arc (i,j) can be interpreted as the cost or as the working time of assigning job i to agent j and the corresponding model of EGAP can be formulated as follows:

$$\begin{aligned}
\min \quad & f(\mathbf{x}) = f_{\max}(\mathbf{x}) - f_{\min}(\mathbf{x}) \\
\text{s.t.} \quad & \sum_{i=1}^{n}x_{ij} \geqslant 1, \quad j \in J, \\
& \sum_{i=1}^{n}a_{ij}x_{ij} \leqslant b_j, \quad j \in J, \\
& \sum_{j=1}^{m}x_{ij} = 1, \quad i \in I, \\
& x_{ij} = 0 \quad \text{or } 1.
\end{aligned} \tag{3}$$

Because of the same reasons as these in [27], the problem (3) is NPH as well, and judging the existence of a feasible solution for problem (3) is NPC. Clearly, the objective function of EGAP (3) is nonadditive.

A more general and practical situation is that a predetermined equilibrium level S, which expresses an expected upper bound of equilibrium degree of the jobs assignment, is given and two types of weights ξ_{ij} and c_{ij} are associated with each arc (i,j), where the weight ξ_{ij} can be interpreted as the time and c_{ij} as the cost of finishing job i at agent j. By such a way, the related problem can be cast as follows

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\sum_{j=1}^{m}c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{i=1}^{n}x_{ij} \geqslant 1, \quad j \in J, \\
& \sum_{i=1}^{n}a_{ij}x_{ij} \leqslant b_j, \quad j \in J, \\
& \sum_{j=1}^{m}x_{ij} = 1, \quad i \in I, \\
& f_{\max}(\mathbf{x}) - f_{\min}(\mathbf{x}) \leqslant S, \\
& x_{ij} = 0 \quad \text{or } 1,
\end{aligned} \tag{4}$$

where $f_{\max}(\mathbf{x})$ and $f_{\min}(\mathbf{x})$ are defined as that in model (2). From practical point of view, the model (4) is more suitable and flexible than the model (3). Clearly, the model (4) is a directly extended version of model (1) by adding an extra constraint $f_{\max}(\mathbf{x}) - f_{\min}(\mathbf{x}) \leqslant S$. The Eq. (4) is thus called as *equilibrium constrained generalized assignment problem* (ECGAP).

A variation of ECGAP (4) is the following bi-criteria GAP

$$\begin{aligned} \min \quad & \left(\sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}, f_{\max}(\mathbf{x}) - f_{\min}(\mathbf{x}) \right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} \geqslant 1, \quad j \in J, \\ & \sum_{i=1}^{n} a_{ij} x_{ij} \leqslant b_j, \quad j \in J, \\ & \sum_{j=1}^{m} x_{ij} = 1, \quad i \in m, \\ & x_{ij} = 0 \quad \text{or } 1. \end{aligned} \tag{5}$$

Follows from point of view in decision making, the model (5) provides a more comprehensively optional decision making strategies for the decision makers such that they can choose a reasonable solution or a compromise solution from the Pareto-frontier according to their preferences.

4. The heuristic equilibrium strategy based GA

The main purpose of this section is to design a solution algorithm of the EGAP (3). This section is organized as follows: firstly, the solution representation and the solution initial approach are given in Section 4.1. Secondly, the crossover operator and the mutation operator are developed in Section 4.2 and Section 4.3, respectively. Finally, the selection approach and the details of the framework of the designed GA are summarized in Section 4.4.

For convenience, we just take the index k to denote the kth arc e_k in the arcs set E.

4.1. Representation and initialization

The representation which directly impairs the crossover process and mutation process is an important stage for the GA. There are many ways to represent a solution of optimization problem. In this paper, we employ a mapping (*i.e.*, a set consisting of job-agent pairs) from I to J to represent (or encode) a chromosome. For the related problem in this paper, the length of a chromosome can be exactly determined and equals to $n = |I|$, *i.e.*, the number of jobs. A chromosome is encoded as a mapping $\mathbf{x}(i) = j, i \in I, j \in J$, where j is the index of the agent to which the job i is assigned. By such an encoding method, the decoding process is very simple. For convenience, the set of jobs assigned to agent j is denoted as $\mathbf{x}[j]$ and notation $r(\mathbf{x}, j) = b_j - \sum_{i \in \mathbf{x}[j]} a_{ij}$ denotes the remains of resource of agent j.

The initialization serving as the role of generating initial chromosomes is also an important process in genetic algorithm. We take *pop_size* to denote the number of chromosomes. The initialization process of this problem is very simple. It can be described as follows: firstly, assign each agent with a randomly selected job and then randomly select a job from the remain jobs and assign it to a randomly selected agent until all jobs are assigned to agents. The main ideas is summarized as Procedure 1.

Procedure 1: Initial()

Input: chromosomes $\mathbf{x}_i$, $i = 1, 2, \ldots,$ *pop_size* and EGAP;
output: chromosomes $\mathbf{x}_i$, $i = 1, 2, \ldots,$ *pop_size*;
1 **for** $i \leftarrow 1$ **to** *pop_size* **do**
2 $I' \leftarrow I$; $m \leftarrow |J|$; $n \leftarrow |I|$;
3 randomly choose m different jobs $i_1, i_2, \ldots, i_m$ from I';
4 **for** $k \leftarrow 1$ **to** m **do** $\mathbf{x}_i(i_k) \leftarrow k$;
5 $I' \leftarrow I' \backslash \{i_k | k = 1, 2, \ldots, m\}$;
6 **while** $|I'| \neq 0$ **do**
7 randomly choose agent k from J;
8 randomly choose job i' from I';
9 $\mathbf{x}_i(i') \leftarrow k$; $I' \leftarrow I' \backslash \{i'\}$;
10 **end**
11 **end**

Because no measure is employed to ensure the feasibility of the obtained chromosomes in Procedure 2, so most of the initial chromosomes may be infeasible.

4.2. Crossover operation

Let $P_c \in (0,1)$ be the crossover probability. In order to determine the parents for crossover operation, we repeat the following process from $i = 1$ to *pop_size*: Randomly generating a real number r from interval $(0,1)$, the chromosome $\mathbf{x}_i$ is selected if $r < P_c$. We denote the selected parents by $\mathbf{x}'_1, \mathbf{x}'_2, \ldots$ and divided them into following pairs:

$$(\mathbf{x}'_1, \mathbf{x}'_2), \quad (\mathbf{x}'_3, \mathbf{x}'_4), \quad (\mathbf{x}'_5, \mathbf{x}'_6), \ldots$$

The base ideas of the crossover is illustrated by Fig. 1. We take the pair $(\mathbf{x}'_1, \mathbf{x}'_2)$ as an example to illustrate the crossover process. Randomly select jobs $i_1, i_2, \ldots, i_k$ from $\{i | \mathbf{x}'_1(i) \neq \mathbf{x}'_2(i), i \in I\}$. Assume that $\mathbf{x}'_1(i_k) = j'_k$ and $\mathbf{x}'_2(i_k) = j''_k$, respectively. For $p = 1$ to k, swap the assignment of job i_p by $\mathbf{x}'_1(i_p) = j''_p$ and $\mathbf{x}'_2(i_p) = j'_p$. For example, the jobs 4 and 5 are selected in Fig. 1. These operations of crossover are summarized as Procedure 2.

Procedure 2: Crossover$(\mathbf{x}'_1, \mathbf{x}'_2)$

Input: $\mathbf{x}'_1, \mathbf{x}'_2$;
result: $\mathbf{x}'_1, \mathbf{x}'_2$;
1 randomly select jobs $i_1, i_2, \ldots, i_k$ from $\{i | \mathbf{x}'_1(i) \neq \mathbf{x}'_2(i), i \in I\}$;
2 **for** $p \leftarrow 1$ **to** k **do**
3 $\quad s \leftarrow \mathbf{x}'_1(i_p); t \leftarrow \mathbf{x}'_2(i_p)$;
4 $\quad$ **if** $\xi_{i_p s} < \xi_{i_p t}$ **then** $\mathbf{x}'_2(i_p) \leftarrow \mathbf{x}'_1(i_p)$;
5 $\quad$ **if** $\xi_{i_p s} > \xi_{i_p t}$ **then** $\mathbf{x}'_1(i_p) \leftarrow \mathbf{x}'_2(i_p)$;
6 **end**

4.3. Mutation operation

Let $P_m \in (0,1)$ be the mutation probability. We employ the following operator to select the chromosome to be mutated: For $i = 1$ to *pop_size*, randomly generate a real number r from interval $(0,1)$; if $r \leqslant P_m$, then the chromosome $\mathbf{x}_i$ is selected to be mutated. Let $\mathbf{x}$ be the selected chromosome to be mutated.

For the EA, if too many infeasible solutions are visited during the iteration, it is doubtless that the efficiency of the algorithm will be greatly decreased. Because some of the chromosomes after crossover operator are infeasible, therefore the mutation serves as the role of local search and that of decreasing the infeasibility of a solution as well.

The essential idea of mutation is to shift the assignment of a selected job i from $\mathbf{x}(i)$ to another agent. To ensure the solution as feasible as possible after mutation, some heuristic measures are adopted to improve the feasibility of the mutated solutions in the mutation process.

The following is the mutation operation for the EGAP (3). Follows from the objective function of this problem, if the corresponding value $f_{\max}(\mathbf{x})$ is decreasing and the value $f_{\min}(\mathbf{x})$ is increasing after the mutation operation, then value of the objective function of the new chromosome is likely decreased. The mentioned heuristic measures can be summarized as follows.

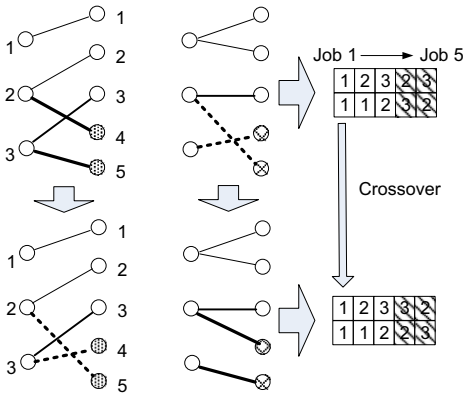

Fig. 1. The crossover operation.

For convenience, we define

$$c_j(\mathbf{x}) = \sum_{i \in \mathbf{x}[j]} \xi_{ij}, \quad j \in J.$$

If $\mathbf{x}$ is feasible, then randomly choose an agent j' from J according to probability distribution

$$P(j) = c_j(\mathbf{x}) / \sum_{k \in J} c_k(\mathbf{x}), \quad j \in J. \tag{6}$$

Then choose a job i' from $\mathbf{x}[j']$ and shift i' to an agent $j'' \neq j'$ randomly chosen from J according to probability distribution

$$P'(j) = c'_j(\mathbf{x}) / \sum_{k \in J} c'_k(\mathbf{x}), \quad j \in J, \tag{7}$$

where $c'_j(\mathbf{x}) = \frac{1}{c_j(\mathbf{x})}, j \in J$.

If $\mathbf{x}$ is infeasible, choose the agent j' with $r(\mathbf{x},j') = \min\{r(\mathbf{x},j) | j \in J\} < 0$ and define $f'[j'] = \{i | a_{ij'} \geqslant |r(\mathbf{x},j')|, i \in \mathbf{x}[j']\}$. If $f'[j'] \neq \emptyset$, then choose job i' from $f'[j']$ uniformly at random and define $J' = \{k | r(\mathbf{x},k) \geqslant a_{i'k}\}$; if $J' \neq \emptyset$, then shift i' to an agent $j'' \neq j'$ randomly chosen from J according to the probability distribution

$$P''(j) = c'_j(\mathbf{x}) / \sum_{k \in J'} c'_k(\mathbf{x}), \quad j \in J', \tag{8}$$

where $c'_j(\mathbf{x}) = \frac{1}{c_j(\mathbf{x})}, j \in J$. If $J' = \emptyset$, choose the agent j'' with

$$r(\mathbf{x},j'') = \max\{r(\mathbf{x},k) | k \in J, k \neq j'\} \tag{9}$$

and shift i' to j''. If $f'[j'] = \emptyset$, then choose job i' from $\mathbf{x}[j']$ uniformly at random and shift i' to an agent $j'' \neq j'$ chosen from J by using the similar ways as that in the case $f'[j'] \neq \emptyset$.

Furthermore, if an agent $j \in J$ is assigned only one job before the mutation, then there may be no job being assigned to this agent after mutation. Therefore, to ensure the feasibility of the chromosome, some adjusting must be made to cope with this situation after the mutation. The base ideas can be described as follows: randomly choose an agent w with $|\mathbf{x}'_1[w]| \geqslant 2$ and then randomly choose a job k from $\mathbf{x}'_1[w]$ and shift the job k to agent j by $\mathbf{x}(k) = j$.

These ideas are illustrated in Fig. 2 and the details of mutation operator of GA for EGAP (3) are summarized as Procedure 3.

Procedure 3: Mutation ($\mathbf{x}$)

Input: chromosome $\mathbf{x}$ to be mutated;
output: chromosome $\mathbf{x}$ after mutation;
repeat
  if $\mathbf{x}$ *is feasible* **then**
    randomly choose $j' \in J$ according to $P(j), j \in J$ in Eq. (6);
    choose a job i' from $\mathbf{x}[j']$;
    randomly choose $j'' \neq j'$ from J according to $P'(j), j \in J$ in Eq. (7);
    $\mathbf{x}(i') \leftarrow j''$;
  else
    choose the agent j' with $r(\mathbf{x},j') = \min\{r(\mathbf{x},j) | j \in J\} < 0$;
    define $f'[j'] = \{i | a_{ij'} \geqslant |r(\mathbf{x},j')|, i \in \mathbf{x}[j']\}$;
    if $f'[j'] \neq \emptyset$ **then**
      choose job i' from $f'[j']$ uniformly at random;
      define $J' = \{k | r(\mathbf{x},k) \geqslant a_{i'k}\}$;
      if $J' \neq \emptyset$ **then**
        randomly choose an agent $j'' \neq j'$ from J according to (8);
      else
        randomly choose the agent j'' according to (9);
      end
      $\mathbf{x}(i') \leftarrow j''$;
    else
      choose job i' from $\mathbf{x}[j']$ uniformly at random; randomly choose an agent $j'' \neq j'$ from J with the similar method as that in the case $f'[j'] \neq \emptyset$;$\mathbf{x}(i') \leftarrow j''$;
    end
  end
until $\mathbf{x}$ *is feasible*;

Remark 4.1. By the mutation operation, an infeasible solution $\mathbf{x}$ can be quickly permutated into a feasible solution. Generally speaking, this permutation can improve the objective value of EGAP (3) as well.

4.4. Selection process

The evaluation function eval($\mathbf{x}$) of EGAP (3) is defined as the reciprocal of the value of the objective function, namely

$$\text{eval}(\mathbf{x}) = \frac{1}{f_{\max}(\mathbf{x}) - f_{\min}(\mathbf{x})}$$

Suppose that fit ($\mathbf{x}_i$) denotes the fitness of the ith chromosome $\mathbf{x}_i$, $i = 1,2,\ldots,pop\_size$. Let

$$\text{fit}(\mathbf{x}_i) = \frac{\text{eval}(\mathbf{x}_i)}{\sum_{k=1}^{pop\_size}\text{eval}(\mathbf{x}_k)}, \quad p_i = \sum_{k=1}^{i}\text{fit}(\mathbf{x}_i). \tag{10}$$

Then we employ the spanning roulette wheel approach to select the chromosomes: randomly generate a number $p \in (0,1)$; if $p \in [p_{i-1}, p_i)$, then the chromosome $\mathbf{x}_i$ is selected. The select process is summarized as Procedure 4.

Procedure 4: Select ()

Input: chromosomes $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{pop\_size}$;

Output: the selected chromosomes $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{pop\_size}$;

compute the fitness with Eq. (10);
for $p \leftarrow 1$ **to** *pop_size* **do**
    randomly generate a number $p \in (0,1)$;
    if $p \in [p_{i-1}, p_i)$ **then** chromosome $\mathbf{x}_i$ is selected;
end
replace the chromosomes with the selected chromosomes;

Algorithm 5: heuristic equilibrium strategy based GA for EGAP (3)

Input:P_c, P_m, EGAP, *pop_size* and the number T of iteration;
output:the obtained best chromosome;
`Initial ()` /* see Procedure 1*/
for $t \leftarrow 1$ **to** T **do**
    $p \leftarrow 0$; $q \leftarrow 0$;
    for $k \leftarrow 1$ **to** *pop_size* **do**
        generate $r \sim U(0,1)$; **if** $r \geqslant P_c$ **then continue**;
        if $p = 0$ **then** $p \leftarrow k$ **else** $q \leftarrow k$;
        if $q = 0$ **then continue**;
        if $\mathbf{x}_p = \mathbf{x}_q$ **then** {$p \leftarrow 0$; $q \leftarrow 0$; **continue**;}
        `Crossover`($\mathbf{x}_p, \mathbf{x}_q$) /* see Procedure 2*/
        $p \leftarrow 0$;$q \leftarrow 0$;
    end
    for $k \leftarrow 1$ **to** *pop_size* **do**
        generate $r \sim U(0,1)$; **if** $r \geqslant P_m$ **then continue**;
        `Mutation` ($\mathbf{x}_k$) /* see Procedure 3*/
    end
    compute p_i according to Eq. (10);
    `Select()` /* see Procedure 4*/
end
report the obtained best chromosome as the final solution;

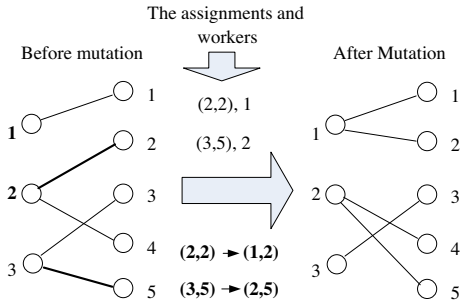

Fig. 2. The mutation operation.

Table 1
The test results for small scale benchmarks (5 runs per instance).

Test examples				$f_{max}(\mathbf{x}) - f_{min}(\mathbf{x})$		Errors	
Problem ID	No.	m	n	Min	Max	Mean	Standard deviation
Gap8	1	8	48	0/5	0/5	0.0	0.00
	2	8	48	0/5	0/5	0.0	0.00
	3	8	48	0/5	0/5	0.0	0.00
	4	8	48	0/5	0/5	0.0	0.00
	5	8	48	0/5	0/5	0.0	0.00
Gap10	1	10	40	0/2	1/3	0.4	0.28
	2	10	40	0/2	1/3	0.4	0.28
	3	10	40	0/3	1/2	0.4	0.24
	4	10	40	0/2	1/3	0.4	0.28
	5	10	40	0/3	1/2	0.4	0.24
Gap11	1	10	50	0/2	1/3	0.4	0.28
	2	10	50	0/3	1/2	0.4	0.24
	3	10	50	1/5	1/5	1.0	0.00
	4	10	50	0/4	1/1	0.2	0.16
	5	10	50	0/2	1/3	0.4	0.28
Gap12	1	10	60	0/3	1/2	0.4	0.24
	2	10	60	0/4	1/1	0.2	0.16
	3	10	60	0/5	0/5	0.0	0.00
	4	10	60	0/3	1/2	0.4	0.24
	5	10	60	0/5	0/5	0.0	0.00

5. The numerical experiments

In this section, some numerical experiments are given to verify the efficiency of the designed GA for solving EGAP (3). The numerical experiments in this section are divided in two parts, *i.e.*, the experiments on small scale benchmarks in Section 5.1 and that on relatively large scale benchmarks in Section 5.2.

By extensive sensitivity analysis, the adopted parameters in following experiments are: the crossover probability $P_c = 0.5$, the mutation probability $P_m = 0.3$, the population size $pop\_size = 2n$; the adopted evolution generations are $Gen = 2000$ for $m \times n \leqslant 4000$ and $Gen = 3000$ for $m \times n \geqslant 4000$. The algorithm is tested on a computer IBM ThinkPad R40/Pentium 2(2 GHz) 256 M memory.

5.1. Tests for small scale examples

The examples in this section are those taken from the OR-Library.[1] There are overall 12 groups of benchmarks identified from gap1 to gap12 in OR-Library and each group of them includes 5 test problems. The experiment results of gap8, gap10, gap11 and gag12 are listed in Table 1. The other examples are so easy to obtain the optimal solution, therefore, they are omitted here.

The test results are reported in Table 1 where each of the benchmarks is executed 5 runs. Follows from Table 1, there are 8 examples of which the probability to find the optimal solution in 5 runs are 100% and all the solutions found in each of 5 runs are the optimal solutions or the secondary optimal solutions. The average run time of the program with these example is 97 s. So the designed algorithm is efficient to solve the small scale benchmarks.

[1] http://people.brunel.ac.uk/~mastjjb/jeb/orlib/gapinfo.html.

Table 2
The test results for large scale benchmarks (20 runs per instance).

Test examples				Test results	
Problem ID	No	m	n	Mean	Standard deviation
Gap_d	1*	5	100	0.00	0.00
	2*	5	200	1.00	0.00
	3	10	100	0.55	0.21
	4	10	200	2.20	0.44
	5	10	400	1.25	0.16
	6	15	900	0.35	0.48
	7	20	100	1.60	0.43
	8	20	200	1.35	0.24
	9	20	400	0.45	0.59
	10*	20	1600	1.00	0.00
	11	30	900	1.35	0.24
	12	40	400	0.30	0.13
	13	40	1600	1.70	0.45
	14	60	900	1.40	0.27
	15	80	1600	1.80	0.65
Gap_e	1*	5	100	0.00	0.00
	2*	5	200	0.00	0.00
	3	10	100	1.35	0.26
	4	10	200	1.25	0.22
	5	10	400	1.60	0.37
	6	15	900	0.20	0.10
	7	20	100	1.30	0.24
	8	20	200	0.55	0.17
	9	20	400	0.75	0.26
	10	20	1600	1.45	0.27
	11	30	900	1.35	0.25
	12	40	400	1.55	0.34
	13	40	1600	0.35	0.13
	14	60	900	1.80	0.42
	15	80	1600	1.75	0.39

5.2. Tests for large scale examples

In this section, the designed GA for solving EGAP Eq. (3) is tested on the benchmarks taking from the GAP benchmarks web-site.[2] There are 5 groups of benchmarks, which are identified as gap_a, gap_b, gap_c, gap_d and gap_e, respectively, and each of them has 15 examples, in this web-site. In this section, we just give out the test results for gap_d and gap_e. The test results are reported in Table 2, where each of the benchmarks is executed 20 runs.

Note that value 0 is an obvious lower bound of the objective function of EGAP (3). So, follows from Table 2, the error between the obtained average value of the objective function and this lower bound is very small and, to a great extent, the obtained solution can be used to approximate the true optimal solution. Furthermore, the standard deviations in Table 2 are relatively small and, correspondingly, the designed GA is stable for solving the EGAP (3).

From the point of view in computing efficiency, for the same benchmarks, the designed GA for EGAP is more efficient than some heuristic algorithms for GAP in sense of compute time and convergence speed. From the computing processes of these test problems, we find that the larger the $\frac{n}{m}$, the higher the computing efficiency, *i.e.*, the designed algorithm can quickly converge to the solutions with very small objective value.

6. Conclusions

In this paper, the equilibrium generalized assignment problem and the equilibrium constrained generalized assignment problem are proposed. Follows from the test results, the designed GA for solving EGAP (3) is very efficient to find a valid approximate solution of the EGAP (3).

Additional research is needed regarding the further application of the EGAP and the idea in the designed GA in real systems. For example, we know that the GAP has been applied in the vehicle route problem (VRP) [8] in logistics distribution. Actually, the equilibrium of the working time of the vehicles (*i.e.*, the drivers) is usually required in the VRP of logistics distribution in real system. Furthermore, as stated previously, the Eq. (4) is a more suitable form of EGAP in practice. Therefore, how to design an efficient GA to solve Eq. (4) is another research topic in future.

[2] http://www.al.cm.is.nagoya-u.ac.jp/~yagiura/gap/.

Acknowledgement

This research is supported by National Natural Science Foundation of China (No. 60174049).

References

[1] M. Amini, M.M. Racer, A hybrid heuristic for the generalized assignment problem, European Journal of Operational Research 87 (1995) 343–348.
[2] P. Avella, M. Boccia, I. Vasilyev, A computational study of exact knapsack separation for the generalized assignment problem, Computational Optimization and Applications 45 (2010) 543–555.
[3] V. Balachandran, An integer generalized transportation model for optimal job assignment in computer networks, Operational Research 24 (1976) 742–759.
[4] A. Barcia, K. Jornsten, Improved Lagrangean decomposition: an application to the generalized assignment problem, European Journal of Operational Research 114 (1999) 165–177.
[5] D.G. Cattrysse, L.N. Van Wassenhove, A survey of algorithms for the generalized assignment problem, European Journal of Operational Research 60 (1992) 260–272.
[6] P.C.H. Chu, J.E. Beasley, A genetic algorithm for the generalised assignment problem, Operations Research 24 (1997) 17–23.
[7] R. Cohen, L. Katzir, D. Raz, An efficient approximation algorithm for the generalized assignment problem, Information Processing Letters 100 (2006) 162–166.
[8] A. De Maio, C. Roveda, An all zero-one algorithm for a certain class of transportation problems, Operations Research 19 (1971) 1406–1418.
[9] A. Geoffrion, G.W. Graves, Multicommodity distribution system design by Benders decomposition, Management Science 20 (1974) 822–844.
[10] S. Haddadi, H. Ouzia, Effective algorithm and heuristic for the generalized assignment problem, European Journal of Operational Research 153 (2004) 190–194.
[11] P.M. Hahn, B. Kim, M. Guignard, J.M. Smith, Y. Zhu, An algorithm for the generalized quadratic assignment problem, Computational Optimization and Applications 40 (2008) 351–372.
[12] S.L. Janak, M.S. Taylor, C.A. Floudas, M. Burka, T.J. Mountziaris, Novel and effective integer optimization approach for the NSF panel-assignment problem: a multi-resource and preference constrained generalized assignment problem, Industrial and Engineering Chemistry Research 45 (2006) 258–265.
[13] V. Jeet, E. Kutanoglu, Lagrangian relaxation guided problem space search heuristics for generalized assignment problems, European Journal of Operational Research 182 (2007) 1039–1056.
[14] M. Laguna, J.P. Kelly, J.L. Gonzalez-Velarde, F. Glover, Tabu search for the multilevel generalized assignment problem, European Journal of Operational Research 82 (1995) 176–189.
[15] S. Martello, P. Toth, An algorithm for the generalized assignment problem, in: Proceedings of the Ninth IFORS International Conference on Operational Research, pp. 589–603.
[16] S. Mitrovic-Minic, A.P. Punnen, Local search intensified: very large-scale variable neighborhood search for the multiresource generalized assignment problem, Discrete Optimization 6 (2009) 370–377.
[17] M.G. Narciso, L.A.N. Lorena, Lagrangean/surrogate relaxation for generalized assignment problems, European Journal of Operational Research 182 (2007) 1039–1056.
[18] R.M. Nauss, Solving the generalized assignment problem: an optimizing and heuristic approach, INFORMS Journal on Computing 15 (2003) 249–266.
[19] I.H. Osman, Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches, OR Spectrum 17 (1995) 211–225.
[20] L. Ozbakir, A. Baykasoglu, P. Tapkan, Bees algorithm for generalized assignment problem, Applied Mathematics and Computation 215 (2010) 3782–3795.
[21] P.M. Pardalos, F. Rendl, H. Wolkowicz, The Quadratic Assignment: A Survey and Recent Developments, in: P.M. Pardalos, H. Wolkowicz (Eds.), Quadratic Assignment and Related Problems, volume 16 of DIMACS: Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, USA, 1994, pp. 1–42.
[22] J.S. Park, B.H. Lim, Y. Lee, A Lagrangean dual-based branch-and-bound algorithm for the generalized multi-assignment problem, Management Science 44 (1998) 271–282.
[23] A.B. Poore, Multi-dimensional Assignment Problems Arrising in Multi-target and Multi-sensor Tracking, in: P.M. Pardalos, L.S. Pitsoulis (Eds.), Nonlinear Assignment Problems: Algorithms and Applications, Combinatorial Optimization, vol. 7, Kluwer Academic Publishers, Netherlands, 2000, pp. 13–34.
[24] C. Privault, L. Herault, Solving a real world assignment problem with a metaheuristic, Journal of Heuristics 4 (1998) 383–398.
[25] G.T. Ross, R.M. Soland, A branch and bound algorithm for the generalized assignment problem, Mathematical Programming 8 (1975) 91–103.
[26] G.T. Ross, R.M. Soland, Modeling facility location problems as generalized assignment problems, Management Science 24 (1977) 345–357.
[27] S. Sahni, T. Gonzalez, *p*-Complete approximation problems, Journal of the ACM 23 (1976) 555–565.
[28] M. Savelsbergh, A branch-and-price algorithm for the generalized assignment problem, Operations Research 45 (1997) 831–841.
[29] D.B. Shmoys, E. Tardos, An approximation algorithm for the generalized assignment problem, Mathematical Programming 62 (1993) 461–474.
[30] A. Shtub, K. Kogan, Capacity planning by the dynamic multi-resource generalized assignment problem (DMRGAP), European Journal of Operational Research 105 (1998) 91–99.
[31] R.F. Subtil, E.G. Carrano, M.J.F. Souza, R.H.C. Takahashi, Using an enhanced integer NSGA-II for solving the multiobjective generalized assignment problem, in: 2010 IEEE World Congress on Computational Intelligence, pp. 1–7.
[32] M.F. Tasgetiren, P.N. Suganthan, T.J. Chua, A. Al-Hajri, Differential evolution algorithms for the generalized assignment problem, in: 2009 IEEE Congress on Evolutionary Computation, pp. 2606–2613.
[33] J.M. Wilson, A genetic algorithm for the generalised assignment problem, Journal of the Operational Research Society 48 (1997) 804–809.
[34] M. Yagiura, T. Ibaraki, F. Glover, An ejection chain approach for the generalized assignment problem, INFORMS Journal on Computing 16 (2004) 133–151.
[35] M. Yagiura, T. Ibaraki, F. Glover, A path relinking approach with ejection chains for the generalized assignment problem, European Journal of Operational Research 169 (2006) 548–569.
[36] M. Yagiura, S. Iwasaki, T. Ibaraki, F. Glover, A very large-scale neighborhood search algorithm for the multi-resource generalized assignment problem, Discrete Optimization 1 (2004) 87–98.
[37] C.W. Zhang, H.L. Ong, An efficient solution to biobjective generalized assignment problem, Advances in Engineering Software 38 (2007) 50–58.