IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 8 Issue 1, January 2021
**ISSN (Online) 2348 – 7968 | Impact Factor (2020) – 6.72**
www.ijiset.com

# Solving Generalized Assignment Problem with the Differential Evolutionary Algorithm

**Ziqiao Yu[1], Xiaoxia Zhang[2]**
[1] College of Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China, patejuliacaqdj70@gmail.com

[2] Professor of College of Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China, zhangxiaoxia@ustl.edu.cn

**Abstract**
The Generalized Assignment problem (GAP) is a kind of combination optimization problem with the aim of finding the minimum cost. There are many intelligent optimization algorithms for GAP. Differential Evolutionary algorithm (DE) is one of the most efficient algorithms with strongly global search capacity. However, DE algorithm is often used to solve the continuous problem instead of the combination optimization problem such the GAP. Therefore, we propose a specific algorithm illustration of original DE algorithm and lead the repairing process for the infeasible solutions in solving the GAP. Finally making the comparison between the results of DE algorithm and the classical optimization results can prove the efficient of the DE algorithm in GAP.
***Keywords:*** *Differential Evolutionary Algorithm, Generalized Assignment Problem, repairing process.*

## 1. Introduction

With the gradual development of the information technology, the aim is how to design the best assignment scheme within the various constraints whatever in the current hot 5 G network technology or in a large number of engineering manufacturing problems to achieve the goal of reducing economic costs. Therefore, the generalized assignment problem (GAP) has become the focus of research.
The GAP is NP-hard problem which is the aim of finding the best solution with the minimum cost under the constraints. However, there are many algorithms including the exactly algorithms and intelligent optimization algorithms to solve the GAP. In recent years, with the data scale expansion and difficulty upgrading involved in the GAP, it is difficult to support the solution of GAP by using exactly algorithms, such as Hungarian algorithm and branch bound algorithm. Therefore, the intelligent optimization algorithms have been used to solve the GAP. The intelligent optimization algorithm can search the optimal value closer to the classical optimal value when solving the GAP. The common algorithms include the genetic algorithm [1], tuba search algorithm [2], dynamic tuba search algorithm (DTS) [3], simulate annealing algorithm (SA) [4], bee colony algorithm (BA) [5], variable depth search algorithm (VDS)[6], and differential evolutionary algorithm (DE) [7] and so on.
DE algorithm is one of the most efficient intelligent optimization algorithms. It is proposed by Storn and Price is a population-based random search algorithm, which is easy to show its strong global search ability in continuous search [8]. Therefore, it is widely used to solve the problem of continuous function. With the gradually development of DE algorithm, it has been applied to solve combination optimization problems. Tasgetiren et al [9] proposed a discrete differential evolution algorithm with different population initialization schemes to solve the single machine total weighted delay problem with sequence correlation setting time. Deng et al [10] proposed a new network representation-based acceleration method to evaluate the improvement of the whole insertion neighborhood of job replacement on the basis of the DE algorithm to solve the pipeline workshop scheduling problem. Therefore, DE algorithm has been widely used to various applications in combination problems and achieved the good efficiency.
However, there are few studies for DE algorithm to solve the GAP. Therefore, we propose an specific illustration of DE algorithm for the GAP including the encode and decode process, repairing process, mutation process, crossover process and selection process. Especially in the repairing process, we led into an parameter of pseudo-utility ratios to decide the order of reassigned task in order to change the infeasible solution into the feasible solution.

## 2. Problem Description

The GAP was first proposed by Ross and Soland in 1975 [11]. The GAP aims at finding an assignment solution which has the lowest cost. Define the set of machines as $I = \{1, 2, \ldots, m\}$ and the set of tasks as $T = \{1, 2, \ldots, n\}$. The standard integer programming formulation of the generalized assignment problem is described below:

*Objection function:*

$$\text{minimize} \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

*Subject to:*

$$\sum_{j=1}^{n} r_{ij} x_{i,j} \le b_i \qquad \forall i, 1 \le i \le m \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad \forall j, 1 \le j \le n \tag{3}$$

$$x_{ij} \in \{0,1\} \qquad \forall i, 1 \le i \le m \quad \forall j, 1 \le j \le n \tag{4}$$

In Equation (1), $c_{ij}$ means cost money when task $j$ works in machine $i$. Therefore, Equation (1) is an objective function which expresses to find the minimum total cost of the assignment of task $j$ to machine $i$. Equation (2) aims to make the total resource of each assignment project in the limitation of all assigned machines resource. In this expression, $r_{ij}$ is source cost when task $j$ works in machine $i$ and $b_i$ is the resource owns to machine $i$. Equation (3) uses decision variable $x_{ij}$ to show the assignment condition. When a task is assigned to a certain machine, $x_{ij}$ will be expressed as 1, otherwise 0. Equation (4) ensures the decision variable including 1 or 0.

## 3. Differential evolutionary algorithm for GAP

### 3.1 Encoding and decoding schemes in DE

In DE, each individual is a solution of GAP, which is expressed as a vector of $n$ tasks to be performed. The elements of each individual take random values from the machine set $I = \{1, 2, \ldots, m\}$. The population of every generation consists of $N$ individuals. Therefore, the DE population can be defined as $X = [x_1(g), x_2(g), \ldots, x_{np}(g)]$ and individual $x_i$ can be defined as $x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]$. The following procedure is used for generating initial solutions:

Step 1: The tasks to be performed are assigned to machines from the machine set $I = \{1, 2, \ldots, m\}$.

Step 2: Ensure that all the tasks have been finished.

Step 3: Check whether the generated solutions meet the resource limit constraints.

Step 4: If any solution becomes infeasible, reassign the tasks, which violate the resource limit constraints, to other machines in descending order of a pseudo-utility ratio.

### 3.2 Repairing process

We describe the repair process for infeasible solutions in detail. The pseudo-utility ratio $q_{ij}$ will be introduced to modify the infeasible solution and design the assigning order for the tasks which violate the resource limit constraint. Equation (5) shows the pseudo-utility ratio.

$$q_{ij} = c_{ij}/r_{ij} \tag{5}$$

The specific steps are described as follows:

Step 1: Constraint examination. According to the order of tasks, delete the tasks which violate the resource constraints in task set *T* and return the occupied space $b_i$ of the machine resource.
Step 2: Reassigning process. In the introduction of an order matrix *F*, each column in this order matrix represents the unit value of the task j assigned to each machine $q_{ij}$ in descending order, where $f_{ij}$ represents the unit value ranking $q_{ij}$. And according to the unit value ranking $f_{ij}$, we reassign the tasks in descending and ascending order.

a. Descending process. According to the number in *F*, the removed tasks found the machine within the resource limitation in descend order of the $f_{ij}$.
b. Ascending process. If the removed tasks can not find the proper machines in descending process, the task will find the proper machine within the resource limitation in ascending order of the $f_{ij}$.

## 3.3 mutation process

In the mutation process, each individual from the current population is transformed into mutation individuals. Each mutation individual $v_i$(g) is generated by using Equation (6).

$$V_i(g) = X_{Fg}(g) + F \times (X_{F1}(g) - X_{F2}(g)) \tag{6}$$

Where $x_{Fg}$(g) is the global optimization solution which has the best fitness value, $x_{r1}$(g) and $x_{r2}$(g) are two randomly selected individuals from the current population *X*(g), and *F* is the scaling factor which in between 0 and 2. The shrinkage level of the disturbing individual ($x_{r1}$(g) - $x_{r2}$(g)) depends on *F* and influences the searching direction for the global optimal solution in the DE algorithm.

For illustrating the mutation process, consider $x_{Fg}$(g) = [1,1,2,3,2], $x_{r1}$(g) = [2,2,3,3,1], $x_{r2}$(g) = [2,2,1,3,2] and *F* = 0.6. All the three individuals are feasible. The cost of $x_{Fg}$(g) is 101, which is lower than those the other two individuals according to Tables 2 and 3. Firstly, the shrinkage level of the disturbing individual in Eq. (6) is [0,0,2,0,-1]. Then, the scaling factor *F* is set as 0.6. It can be obtained a temporary individual which is [0,0,1.2,0,-0.6]. The mutation individual $v_i$(g) is obtained by adding the temporary individual to $x_{Fg}$(g). Therefore, the floating-point individual $a_i$(g) is [1.0,1.0,3.2,3.0,1.4]. Finally, the floating-point mutation individual $a_i$(g) is transformed into an integer individual for obtaining the mutation individual $v_i(g)$ = [1,1,3,3,1]. The specific transforming form is shown in Table 1.

**Table 1.** An example of a solution

| *j* | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $a_{ij}(g)$ | 1.0 | 1.0 | 3.2 | 3.0 | 1.4 |
| $v_{ij}(g)$ | 1 | 1 | 3 | 3 | 1 |

## 3.5 crossover process

When the mutation process is finished, the crossover process starts. In the crossover process, the operating objects include the initial and mutation individual vectors of $v_i$(g)= [$v_{i1}$(g), $v_{i2}$(g), ..., $v_{im}$(g)] and $x_i$ (g)= [$x_{i1}$(g), $x_{i2}$(g), ..., $x_{in}$(g)]. A binomial crossover scheme is adopted to generate a trial individual by using Equation (7).

$$u_{ij}(g) = \begin{cases} v_{ij}(g) & if\ rand \le p_c \quad ;j = 1,\ldots,m \\ x_{ij}(g) & otherwise \end{cases} \tag{7}$$

Where $p_c$ is a crossover factor which lies between 0 and 1. *rand* (0,1) is a random number between 0 and 1. If *rand* (0,1) is less than $p_c$, then the variable from the mutation individual is selected to form the trial individual. It can avoid the stagnation of search due to the population homogenization.

## 3.6 selection process

The DE algorithm uses one to one competition between parents and offspring to greedily select. Equation (8) describes the selection strategy which chooses the individuals of population for the next generation from the corresponding trial individuals in the current generation.

$$x_i(g+1) = \begin{cases} u_i(g) & if\ f(u_i) < f(X_i) \\ x_i(g) & otherwise \end{cases} \tag{8}$$

The searching is stopped when the iteration *g* exceeds the maximum iterations $G_m$. Otherwise, the DE algorithm continues the searching for optimizing the population until meeting with all the conditions.

## 4. Computational Results

In this paper, the algorithm is tested by MATLAB R2017b. And we use 12 instance tests. The scales of them are 5*15 and 5*20. In table2 and table 3, there are five columns which show the instance name, scale, *opt*, *best* and *Dev* of two algorithms. The deviations can be calculated in Equation (9).

$$Dev = \frac{best - opt}{opt} \times 100\% \tag{9}$$

Where *best* is the optimal value which is obtained by DE algorithm. *Opt* is the classical optimal value (lower bound) of instance in GAP.

Table 2: DE simulation outcomes in 5*15 instances

| ***instance*** | ***scale*** | ***opt*** | ***best*** | ***Dev (%)*** |
|---|---|---|---|---|
| GAP1-1 | 5*15 | 336 | 331 | 1.49 |
| GAP1-2 | 5*15 | 327 | 327 | 0.00 |
| GAP1-3 | 5*15 | 339 | 337 | 0.59 |
| GAP1-4 | 5*15 | 341 | 334 | 0.88 |
| GAP1-5 | 5*15 | 326 | 324 | 0.61 |

Table 3: DE simulation outcomes in 5*20 instances

| ***instance*** | ***scale*** | ***opt*** | ***best*** | ***Dev*** |
|---|---|---|---|---|
| GAP2-1 | 5*20 | 434 | 428 | 1.38 |
| GAP2-2 | 5*20 | 436 | 421 | 3.44 |
| GAP2-3 | 5*20 | 420 | 418 | 0.48 |
| GAP2-4 | 5*20 | 419 | 415 | 0.95 |
| GAP2-5 | 5*20 | 428 | 420 | 1.87 |

From Table 2 and Table 3, it can be seen that the optimal values obtained by DE algorithm are closer to the classical optimal values. The deviations of DE algorithm are less than 2%. Therefore, DE algorithm has good performance in solving the GAP.

## 5. Conclusions

In this paper, we proposed a specific algorithm illustration of original DE algorithm and lead the repairing process for the infeasible solutions in solving the GAP. And using 12 instances exanimate the DE algorithm, it has been proved that the DE algorithm has good performance in solving the GAP. It also be proved that DE algorithm has advantage of solving the GAP and is further worth to studying.

**Acknowledgments**

This work it was supported by Project of Liaoning Xincheng Co., Ltd (Grant No. L20170989) and National College Students' innovation and entrepreneurship training program project (No. 202010146483)

## References

[1] Beasley, J. E.; Chu, P. C. A genetic algorithm for the set covering problem[J]. EUR. J. OPER. RES. 1996, 94(2), 392-404.
[2] Degroote, H.; González-Velarde, J. L.; De, Causmaecker. P. Applying Algorithm Selection ‐ a Case Study for the Generalized Assignment Problem[J]. endm. 2018, 69, 205-212.
[3] Higgins A J. A dynamic tabu search for large-scale generalized assignment problems[J]. COMPUT. OPER. RES. 2001, 28(10), 1039-1048.
[4] Osman, I. H. Heuristics for the generalized assignment problem: simulated annealing and tabu search approaches[J]. OR. Spectrum. 1995, 17(4), 211-225.
[5] Choong, S. S.; Wong, L. P.; Lim, C. P. An artificial bee colony algorithm with a modified choice function for the traveling salesman problem[J]. SWARM. EVOL. COMP. 2019, 44, 622-635.
[6] Çetin, K.; Tuzkaya, G.; Vayvay, O. A mathematical model for personnel task assignment problem and an application for banking sector[J]. IJOCTA. 2020, 10(2), 147-158.
[7] Sethanan, K.; Pitakaso, R. Improved differential evolution algorithms for solving generalized assignment problem[J]. EXPERT. SYST. APP. 2016, 45, 450-459.
[8] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," J. Global Optim, vol. 11, no. 4, pp. 341-359, 1997.
[9] Tasgetiren, M. F.; Suganthan, P. N.; Pan, Q. K. An ensemble of discrete differential evolution algorithms for solving the generalized traveling salesman problem[J]. APPL. MATH. COMPUT. 2010, 215(9), 3356-3368.
[10] Deng, G.; Gu, X. A hybrid discrete differential evolution algorithm for the no-idle permutation flow shop scheduling problem with makespan criterion[J]. COMPUT. OPER. RES. 2012, 39(9), 2152-2160.
[11] Ross, G.T.; Soland, R. M. A branch and bound algorithm for the generalized assignment problem[J]. MATH. PROGRAM. 1975, 8(1), 91-103.

**Ziqiao Yu** is currently reading in College of Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China. She is a postgraduate in the third year.

**Xiaoxia Zhang** is currently working as a Professor at the College of Software Engineering, University of Science and Technology LiaoNing, Anshan 114051, China.