An International Journal of Optimization and Control: Theories & Applications
ISSN: 2146-0957 eISSN: 2146-5703
Vol.10, No.1, pp.37-46 (2020)
https://doi.org/10.11121/ijocta.01.2020.00796

RESEARCH ARTICLE

A multi-start iterated tabu search algorithm for the multi-resource agent bottleneck generalized assignment problem

Gulcin Bektur*

Department of Industrial Engineering, Iskenderun Technical University, Turkey
gulcin.bektur@iste.edu.tr

ARTICLE INFO

Article history:
Received: 4 March 2019
Accepted: 25 September 2019
Available Online: 6 October 2019

Keywords:
Multi- start iterated local search
Multi- resource bottleneck generalised assignment problem
Agent qualifications
Integer linear programming model

AMS Classification 2010:
90C10, 90C59

ABSTRACT

In this study, a multi-resource agent bottleneck generalized assignment problem (MRBGAP) is addressed. In the bottleneck generalized assignment problem (BGAP), more than one job can be assigned to an agent, and the objective function is to minimize the maximum load over all agents. In this problem, multiple resources are considered and the capacity of the agents is dependent on these resources and it has minimum two indices. In addition, agent qualifications are taken into account. In other words, not every job can be assignable to every agent. The problem is defined by considering the problem of assigning jobs to employees in a firm. BGAP has been shown to be NP- hard. Consequently, a multi-start iterated tabu search (MITS) algorithm has been proposed for the solution of large-scale problems. The results of the proposed algorithm are compared by the results of the tabu search (TS) algorithm and mixed integer linear programming (MILP) model.

1. Introduction

Assignment problems (AP) are an important topic which is frequently studied in the literature. AP is generally considered in three classes. The simplest form of the AP, in which each agent can be assigned a job at most, is the classic AP. There are m number of agents and n number of jobs in this problem. Each job must be assigned to an agent so that the total cost is minimal. Each agent should also be assigned a job (one-to-one). Another class of the AP is generalized assignment problem (GAP). In GAP, more than one job can be assigned to an agent. Some subclasses of GAP are multi-resource generalized assignment problem (MRGAP), bottleneck generalized assignment problem (BGAP). Another class of AP is multidimensional AP. In multidimensional AP, jobs are assigned to at least two different resources. Detailed information can be reached from the literature review by Pentico [1].

In the GAP, each agent has a certain capacity. Jobs use this capacity and the capacity of the agent cannot be exceeded. In MRGAP, multiple resources are used for the completion of the jobs. Therefore, the capacity of agents depends on these resources. The capacity parameter of the agents has at least two indices due to parameter dependent on the agent and the resource. There are many applications of the MRGAP in real life. For example, in vehicle routing problems, as vehicles are agents, and jobs are considered to be the places where vehicles should be visited, and the capacity of the vehicles depends on both the weight and the volume of the vehicle, the problem can be considered as the MRGAP [2].

In bottleneck assignment problems (BAP), the objective function is the minimization of the maximum assignment cost or maximum load over all agents. Completion time of the jobs also can be taken into account. In other words, completion time of the last job is minimized in the BAP [3].

GAP is an important problem frequently studied in literature. Studies in the literature can be categorized as studies that proposes exact algorithms and heuristic algorithms. In the studies that propose exact algorithms, the branch-bound algorithm ([4] and [5]), the cutting plane algorithm [6], the branch and price algorithm [7, 8], branch- and- cut algorithm for GAP with additional pair constraints [9] and with min- max regret criterion [10] were used. When the exact solution approaches are used, the solution time of the problem is quite prolonged. Since the GAP problem is an NP-hard problem, it is quite common to use heuristic algorithms that gives the near optimal solution in a short time [11]. In the studies using heuristic

*Corresponding author

algorithms, tabu search algorithm [12-14], genetic algorithm [15], bees algorithm [16], a heuristic based on Lagrangian relaxation [17, 18], LP- based heuristic [19], a hybrid heuristic based on scatter search [20], improved differential evolution algorithm [21], a parallel genetic algorithm [22] and simulated annealing algorithm [14] were used. Degroote et al. [23], poposed a methodology for selection the most suitable algorithm for GAP. Chakravarthy et al. [24], proposed a heuristic algorithm for bottleneck generalized assignment problem. For a strategic variant of GAP, approximation algorithm is proposed by Fadaei and Bichler [25]. Detailed information for GAP can be found in the literature review by Öncan [11].

Although there are many studies related to GAP, the number of studies dealing with the MRGAP is less. MRGAP is the generalization of the GAP. GAP has been shown to be NP- hard and MRGAP is also NP-hard [26]. Karsu and Azizoğlu [3], proposed a branch-bound algorithm for the multi-resource bottleneck GAP. Mazzola and Wilcox [2], proposed a three stage heuristic algorithm for the MRGAP. In the first stage, a suitable solution is obtained and at the other stages, this solution is improved. Yagiura et al. [27], proposed a new algorithm for multi-resource generalized quadratic assignment problem. In the algorithm, the path relinking approach was used in the neighborhood generation. Gavish and Pirkul [28], proposed a heuristic algorithm and branch-bound algorithm for the MRGAP. They also proposed some rules for the reduction of the problem dimensions. Yagiura et al. [29], proposed a TS-based heuristic algorithm for the MRGAP. Mitrovic-Minic and Punnen [30], proposed a heuristic algorithm based on a variable neighborhood search for the MRGAP.

The MRGAP problem is the generalized version of GAP and is a more difficult problem to solve. However, many studies in the literature propose an heuristic algorithm for GAP. Wu et al. [10], Souza et al. [20], Sethanan and Pitakaso [21], and Moussavi et al. [31] proposed an heuristic algorithm for the generalized assignment problem. Difference from the literature, in this study an heuristic algorithm is proposed for the multi-resource agent bottleneck generalized assignment problem with agent qualifications. The differences of the study from literature are agent qualifications are taken into account, a different heuristic algorithm is proposed for the larger size test problems than the problem sizes in the literature and the success of the proposed heuristic is shown through test problems by comparing with Tabu Search in the literature. The TS algorithm has been proposed by Karsu and Azizoğlu [3] in the literature for the problem of MRGAP. The proposed iterated local search algorithm is compared with the TS algorithm. Test problems are generated in two different ways as that takes into account agent qualification and not takes into account agent qualification. In addition, larger size test problems are solved and the success of the algorithm has been shown through test problems. In addition, iterative local search algorithm was proposed for the first time for the MRGAP.

The remainder of this paper is organized as follows. The first section of the study is the introduction, in the second section the problem is defined and MILP model is given. In the third section, the proposed solution method is explained. In the fourth section, experimental results are given and conclusions are given in the final section.

2. Problem description

In this study, multi-resource bottleneck generalized assignment problem (MRBGAP) with agent qualifications was addressed. The problem addressed in this study is defined by the problem of assigning employees to jobs in a firm. Employees are considered as agents. Each jobs must be assigned to an employee. More than one job can be assigned to an employee. Employee capacities depend on employee and shift. The shifts are defined as morning, noon, afternoon, evening and night. Employees' capacities (working hours) vary according to shifts. For example, an employee can work more in the morning shift than in the evening shift. For this reason, the employees' capacity parameter has two indices due to the parameter depending on the employee and the shift. The objective function is the minimization of the completion time of the last job, and this objective function is the bottleneck objective function.
The sets, indices, parameters, decision variables, constraints and objective function of the MILP model are given below;

Sets and indices

N:Set of jobs, N= {1, 2, …, n}

M: Set of agents, M= {1, 2,…, m}

R: Set of resource, R= {1, 2,…,r}

j: job indices where $j \in N$.

i: agent indices where $i \in M$.

k: resource indices where $k \in R$.

Parameters

p_{ijk}*: processing time for job j on agent i and resource k*

b_{ik}*: capacity for agent i on resource k*

$$h_{ij} = \begin{cases} 1; & if\ agent\ i\ capable\ of\ assigning\ job\ j \\ 0; & o.w. \end{cases}$$

M: very large positive number

Decision variables

$$x_{ij} = \begin{cases} 1; & if\ job\ j\ is\ assigned\ to\ agent\ i \\ 0; & o.w. \end{cases}$$

L_{max}: maximum completion time

Model

$$\text{Min } Z = L_{max} \quad (1)$$

s.t.

$$\sum_j p_{ijk} x_{ij} \leq b_{ik} \quad \forall i, k \quad (2)$$

$$\sum_i x_{ij} = 1 \quad \forall j \quad (3)$$

$$\sum_k \sum_j p_{ijk} x_{ij} \leq L_{max} \quad \forall i \quad (4)$$

$$x_{ij} \leq h_{ij} \quad \forall i, j \quad (5)$$

$$x_{ij} \in \{0,1\} \text{ ve } L_{max} \geq 0 \quad (6)$$

Constraint (1) shows the objective function, minimization of the maximum completion time. Constraint (2) ensures agent capacities are not exceeded. With constraint (3), each job is assigned to an agent. The constraint (4) calculates the completion of the last job. The constraint (5) ensures agent qualifications are satisfied. Constraints (6) are the sign constraints.

Table 1: Parameters of p_{ijk}

	pij1				pij2				pij3			
i	1	2	3	4	1	2	3	4	1	2	3	4
1	18	25	6	5	10	23	37	42	36	31	25	14
2	45	41	17	7	9	10	40	15	19	46	15	8
3	44	13	18	10	10	36	9	17	46	9	11	32
4	37	28	45	5	6	28	25	14	10	25	41	15
5	6	31	45	29	24	29	49	35	42	37	9	17
6	37	41	4	11	17	4	30	47	37	38	11	34
7	6	44	30	36	23	40	14	10	45	21	33	12
8	20	25	28	46	42	44	23	11	9	45	26	29
9	21	14	39	14	28	5	27	32	41	44	31	32
10	34	47	48	41	28	8	21	33	48	12	15	32
11	32	44	45	17	4	45	21	45	23	11	16	17
12	32	11	20	44	21	43	25	6	39	26	5	47
13	37	45	8	32	46	17	47	24	20	25	15	11
14	12	27	22	38	5	29	12	40	14	10	5	33
15	34	29	13	24	33	11	16	35	13	36	4	42

The problem is also explained by an example. In the example, there are 15 jobs, 4 employees (agents) and 3 shifts (resources). Table 1 shows p_{ijk} values. h_{ij} and b_{jk} parameters are given in Table 2 and Table 3,respectively.

Example: 15 jobs, 4 employee (agent) and 3 shifts (resource)

Table 2: Parameters of h_{ij}

i	hij														
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	0	1	0	1	1	0	0	1	0	1	1	0	1
2	1	0	1	1	1	0	1	1	0	1	1	1	1	0	1
3	0	1	1	1	0	1	1	0	1	1	0	1	1	1	0
4	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1

Table 3: Parameters of b_{jk}

i	bik		
	1	2	3
1	138	102	147
2	120	100	110
3	130	132	95
4	90	105	100

In the best solution, jobs 2, 7 and 10 are assigned to agent 1; jobs 11, 13 and 15 are assigned to agent 2; jobs 3, 6, 9, 12 and 14 are assigned to agent 3; jobs 1, 4, 5, and 8 are assigned to agent 4. Loads of the agents are 254, 263, 269 and 262, respectively. The objective function value is 269.

3. Multi-start iterated tabu search algorithm

Since the problem is NP-Hard, a multi-start iterated tabu search algorithm has been proposed to solve large problem instances.

The iterated local search algorithm is a heuristic algorithm that has three basic stages. The first stage is the generation of the initial solution. At the second stage the solution is improved by a local search method. The third stage is the perturbation stage. The steps of the iterated local search algorithm are given in Figure 1. Once the initial solution is obtained, the algorithm repeats the local search and perturbation steps until the stopping criterion is achieved. If the solution obtained from the local search is better than the current solution, the solution is considered to be the current solution. The perturbation mechanism is intended to escape from local optimal. In the perturbation phase, the solution is changed slightly.

Iterated local search algorithm is applied to many combinatorial optimization problem successfully. Iterated local search algorithm is proposed for the scheduling problem [32], vehicle routing problem [33-36], quadratic assignment problem [37], quadratic knapsack problem [38], hub location problem [39] and shift scheduling [40].

Firstly, abbreviations used in algorithm are given:

$S_{0(c)(n)}$: Initial (current) (neighbor) solution;

S^p: Perturbated solution;

$E_{0(c)(n)}$: Obj. func. value of the initial (current) (neighbor) solution;

CL: Set of jobs;

TLL: Tabu list length;

v: maximum iteration number of TS;

MTLS: Maximum tabu list size;

E_{best}: Objective function value of the best solution; maxStart: Multi- start number of the algorithm

In this study, multi- start iterated tabu search algorithm is proposed for the multi resource bottleneck generalized assignment problem. Different features have been used to increase the success of the proposed algorithm.

Procedure ILS

Generate initial solution S_0;

Apply local search procedure to S_0 and obtain S^*;

While termination condition not meet

Apply perturbation to S^* and obtain S^p;

Apply local search procedure to S^p and obtain S'';

If $f(S'')<f(S^*)$

$S^* \leftarrow S''$;

End

End

Figure 1. Algorithm of the iterated local search

One of the important features of the proposed algorithm is to start the search process multiple times. This feature provides diversification. Initial solution of the algorithm is generated by randomly or by a greedy algorithm. Throughout our preliminary experiments, it was observed that the algorithm reached better solutions faster by using greedy algorithm as an initial solution finding mechanism. However, only the use of the greedy algorithm caused starting with very similar solutions. Thus, random solutions also taken as an initial solution for the investigation of the unexplored regions in the search space. For this, a random number is derived. If this random number is greater then q (a predetermined parameter) the algorithm uses the greedy algorithm. Otherwise random initial solution is generated. TS algorithm is used as a local search algorithm. The TS algorithm and perturbation mechanisms are applied respectively until the stopping criterion is achieved. Once the stopping criterion has been achieved, an initial solution is generated again and the steps are repeated until the number of multiple starting is reached.

In the next section, initial solution finding mechanisms, TS algorithm used in local search, perturbation mechanism and all steps of algorithm will be described.

3.1. Initial solution finding mechanism

3.1.1. A greedy construction heuristic

When generating the initial solution, a job is assigned to the agent with the smallest completion time as possible. For this, p_{ij} values are calculated using Equation 7. p_{ij} denotes the total completion time of the job according to agent on the basis of resource.

$$p_{ij}=\sum_k p_{ijk} \qquad \forall i,j \tag{7}$$

The p_{ij} values are sorted in ascending order and the sp_{ij} matrix is obtained. The aim is to assign the job to the first agent in the sp_{ij} matrix. However, since each agent has a capacity and agent qualifications are taken into account, the agent cannot be assigned to first agent in the sp_{ij} matrix. If the job is not assigned to the first agent, the job is assigned to second order of the agent in the sp_{ij} matrix. Algorithm is repeated until each job is assigned to an agent and a solution is obtained. The algorithm is given on Figure 2.

3.1.2. Random initial solution

In the random solution finding algorithm, the randomly selected job j* is assigned to the randomly selected agent i*. If job j* is not assignable to i*, another agent is randomly generated. The algorithm is working until a solution is obtained.

3.2. Local search algorithm

The TS algorithm was used as the local search algorithm in the proposed heuristic method.

Two methods are used to generate the neighboring solutions from current solution. The first method is to assign each job in the agent with the largest completion time to the other agents. The other method is the reciprocal displacement of jobs in the agent with the largest completion time with the jobs in other agents. All solutions are derived from the current solution by using neighboring solution generation methods.

The best of these solutions is taken, and if the movement in the generation of the neighbor solution is not in the tabu list, the solution is taken directly as the current solution. If a movement is made in the tabu list and the solution is not a better solution than the best solution, the neighbor solution with the second smallest objective function is chosen and the same test is also applied to this solution. This step is repeated until a solution is accepted.

The length of the tabu list is considered as fixed, and when the tabu list is full, the movement that has been in the list for the longest period is deleted. It is forbidden to carry out the movements in the tabu list. If a better solution is obtained than the best solution, the tabu is eliminated and the relevant solution is taken as the current solution.

If the solution is obtained as a result of the use of the first neighboring method, the job and the relevant agent are added to the tabu list. The movement of this job to the relevant agent during the tabu is prohibited. If the solution is obtained as a result of the use of the second neighboring method, replacement of these jobs is prohibited. The algorithm works until it reaches the predetermined number of iterations. The steps of the tabu search algorithm are given in Figure 3.

Procedure a greedy construction heuristic
Input: p_{ijk}, h_{ij}, b_{ik}, sp_{ij}
Output: Initial solution (S_0), Objective funct. value of S_0 (E_0)
exit1←0, n'←0; CL←{1,…,n}
While exit1==0
  exit2←0; x←1; flag←0;
  While exit2==0
    Select the job j* randomly from the CL and assign the j* to the x th order of the agent in the sp_{ij^*} matrix;
    For k=1 **to** r
      If $b_{i^*k}<p_{i^*j^*k}$ **or** $h_{i^*j^*}==0$
      flag←1;
      End
    End
    If flag==1
     x←x+1;
    Else
    exit2←1; n'←n'+1;
    b_{i^*k}←b_{i^*}k-$pi_{^*j^*k}$;CL←CL\{j*};
    End
    If x==m
      exit2←1; n'←0; Initialize CL and b_{ik};
    End
  End
  If n'==n
  exit1←1;
  End
End
Figure 2. Greedy Construction Heuristic

Procedure TS algorithm
Input: p_{ijk}, b_{ik}, h_{ij}, *m*, *n*, *r*, *v*, *MTLS*, S_0, E_0
Output: Near optimal solution (S^*)
S^*←S_0; E^*←E_0; S_c←S_0; E_c←E_0; *TLL←1;*
While *iter<v*
  Generate neighbor solutions and sort ascending order according to *obj. func.value* ($S_n{}^t$); t←1; check←0;
  While check==0
    If *the movement of* $S_n{}^t$ not tabu **or** $E_n{}^t<E_{best}$
      S_c←$S_n{}^t$; E_c←$E_n{}^t$;
      Insert the movement of S_c at the tabu list;
      TLL←TLL+1; check←1;
    Else
      t←t+1;
    End
  End
  If *TLL==MTLS+1*
    Delete the first element in the tabu list;
    TLL←1;
  End
  If $E_c<E^*$
    S^*←S_c; E^*←E_c;
  End
  iter←iter+1;
End
Figure 3. Tabu search algorithm

3.3. Perturbation mechanism

The iterative local search algorithm uses the perturbation mechanism to escape the local optimal. If the perturbation is too strong, the algorithm can move away from promising regions. If perturbation is too small, the algorithm may loop in previously searched regions. Therefore, it is very important to determine the appropriate perturbation length. With perturbation, a new solution (S'') is derived from one of the methods of the neighboring solution from the current solution (S'). If objective function value of S'' is less than objective function value S', then the perturbated solution will be S''.

If S'' is accepted, the value γ is increased by λ. Otherwise, a new S'' solution is derived. λ is taken as a value between 0 and 1. If the number of consecutive rejected solutions reaches a predetermined value (maxTry), the value of γ is increased by λ. The algorithm works until the number of applied moves equal to pertLength. The perturbation mechanism is given in Figure 4.

Procedure Perturbation
Input: S', pertLength, λ, maxtry
Output: Sp
p←1; γ←1+ λ; Sp←S'; try←0;
While p≤pertLength
  Generate a random solution S'', from S' by applying a randomly selected neighborhood structure;
  If f(S'')≤ γ* f(S')
    Sp←S''; γ← γ + λ; p←p+1; try←0;
  Else
    try←try+1;
    If try>maxtry
      γ← γ +λ; try←0;
    End
  End
End
Figure 4. Perturbation mechanism

3.4. Steps of the algorithm

The steps of the proposed multistart iterated tabu search algorithm are given in Figure 5.

Procedure MS- ITS

Input: problem data, maxStart, λ, maxTry, pertLength, *v, MTLS, q, maxiter*

Output: *Near optimal solution (S^*)*

For *s=1:maxStart*

pertLength←1;

Generate a random number rnd;

If rnd≤q

Construct a random initial solution S_0 with the objective function E_0;

(S_0,E_0)←**RandomInitialSolution**(problem data*)*

Else

Construct an initial solution S_0 with the objective function E_0 using greedy algorithm;

(S_0,E_0)←**GreedyInitialSolution**(problem data,sp_{ij}*)*

End

(S^*,E^*)←**TabuSearchAlgorithm**(problem data, v, *MTLS,S_0,E_0)*

$S'=S^*$; $E'=E^*$;

While *iteration<maxiter*

(S^p,E^p)←**Perturbation**(pertLength,λ,maxtry, *S', E')*

(*S'', E''*)←**TabuSearchAlgorithm**(problem data, v, *MTLS,S^p,E^p)*

If *$E''<E^*$*

S^*←S''; E^*←E'';

End

iteration← iteration+1;

End

End

Figure 5. Steps of the algorithm

4. Computational results

The success of heuristic algorithms strongly depends on the selection of the right parameters. The parameters of the heuristic are maxStart, λ, maxTry, pertLength, v, MTLS, maxiter and q. Taguchi experimental design (TED) reduces the number of experiments and it is a successful method for determining the parameters of heuristic algorithms. If the number of parameters is high TED is preferred because it reduces the number of experiments [41]. The parameters of the algorithm were determined with TED due to proposed algorithm has many parameters. Factor levels are given in Table 4. There has been 8 factors and 3 levels. L27 orthogonal array is selected. Since the objective function of the problem is minimization, smaller-the-better type function is selected for the Taguchi design. S/N ratio is given below (Eq- 8). n is the number of observations in each experiment and Y_i is the objective function value.

$$S/N\ ratio = -10 * log\left(\frac{1}{n}\sum_{i=1}^{n} Y_i^{\,2}\right) \quad (8)$$

Table 4: Factor levels

Factors	Levels
maxStart (A)	10; 15 and 20
λ (B)	0.02; 0.03 and 0.04
maxTry (C)	25, 50 and 75
pertLength (D)	10; 20 and 30
v (E)	500; 750 and 1000
MTLS (F)	40; 60 and 80
maxiter (G)	50; 75 and 100
q (H)	0.3; 0.4 and 0.5

In this study, instead of applying TED to each test problem, the following test problems were selected and TED was applied. In Table 5 selected test problems and determined parameters are given. In the selected problems, U [15,25], number of agent is 10 and number of jobs is 200. The largest problem size was preferred. Test problems are generated as described in the study by Karsu and Azizoğlu [3].

Table 5. Selected test problems and determined parameters

c	h_{ij}	A	B	C	D	E	F	G	H
1.4	1	10	0.04	25	10	500	80	50	0.4
	0.7	10	0.02	25	20	500	40	50	0.4
1.6	1	10	0.03	50	20	500	60	50	0.5
	0.7	10	0.02	50	10	500	40	50	0.3

Response table of S/N ratios for the test problem with c=1.4 and h=1 in Table 6.

Table 6. Response table of S/N ratios

Factors	S/N Ratio		
	1	2	3
A	**-63,30**	-63,32	-63,33
B	-63,31	-63,30	**-63,27**
C	**-63,14**	-63,17	-63,16
D	**-63,26**	-63,31	-63,34
E	**-63,08**	-63,21	-63,10
F	-63,32	-63,30	**-63,26**
G	**-63,13**	-63,30	-63,25
H	-63,31	**-63,26**	-63,32

According to the highest S/N values, the levels of the factors are determined and given in Table 5. Taguchi experimental design is analyzed using Minitab 16 for

Windows (Minitab Inc.).

The MILP model was coded in GAMS 24.1.3 program and the Cplex solver was used. Heuristic algorithms were coded in the MATLAB R2012b and implemented on an Intel (R) Core ™ i7- 5500 U CPU at 2.40 GHz with 12 GB of RAM memory and the Windows 10 operating system. The proposed algorithm is compared with the Tabu Search algorithm. The tabu search algorithm and the proposed algorithm were run in equal iterations. For this reason, the tabu search algorithm has been run with a (maxStart * v * maxiter) number of iterations. The neighbor generation of the TS algorithm is same with proposed heuristic approach. The results of the proposed algorithm, the tabu search algorithm, and the MILP (mixed integer linear programming) model are given in Table A1-A2. In Table A1 agent number is 5 and job numbers are 75 and 100. In Table A2 agent number is 10 and job numbers are 150 and 200. The problem specifications including the number of jobs, distribution of the processing times, the value of the c, probability of the parameter h_{ij} are given in the first, second, third and fourth columns, respectively. The MILP solution and CPU time, iterated local search solution and CPU time of the algorithm, TS solution and CPU time are given in Table A1-A2. The results of the ILS and TS algorithms are compared by the MILP solution. The heuristic error was calculated as follows:

$$\%Error = \frac{Heuristic\ solution - MILP\ solution}{MILP\ solution} \times 100$$

In the study, 48 test problems were solved by MILP model, ILP algorithm and TS algorithm. The success of the proposed algorithm is shown by comparing the results of the MILP model and TS algorithm. The proposed heuristic algorithm gave better results than TS algorithm in all test problems except 2 test problems. ILS algorithm gave the same result with TS for 5 test problems. The average error rates in heuristic algorithms according to the number of jobs are given in the Table 7.

Table 7. Average error according to job number

Job number	Average Error	
	ILS	TS
75	0	0,31
100	0,005	0,28
150	0,005	0,87
200	0,13	1,23
Average	0,035	0,6725

The proposed heuristic algorithm found the optimal results for test problems with 75 jobs. When the number of jobs is taken as 100, ILS found the optimal results in all test problems except 1 test problem. MILP model cannot find the best results for test problems with 150 jobs at 3600 sec. The proposed heuristic algorithm found a better solution in a shorter time than the MILP model for 5 test problems out of 12 test problems. When the number of jobs was 200, the ILS algorithm gave better solutions for 6 test problems out of 12 test problems in a shorter time than the MILP model. If the tables are interpreted considering the number of agents, the MILP model found optimal solution for all test problems with 5 agents. The proposed heuristic algorithm found optimal solution for test problems with 5 agents except 1 test problem. MILP model could not find the best solution in 3600 seconds for test problems with 10 agents. The proposed heuristic algorithm provided better solution in 11 test problems out of 24 test problems in a shorter time than the MILP model. When the number of agents was 10, the TS algorithm provided a better solution for only 2 test problems than the ILS algorithm. The proposed heuristic algorithms were run in equal iteration number and ILS algorithm provided solutions in a shorter time than TS algorithm for all test problems. As a result, ILS algorithm displayed better performance than TS algorithm for MRBGAP.

5. Conclusion

In this article, multi-resource agent bottleneck generalized assignment problem (MRBGAP) is considered. The MRBGAP problem is the generalized version of GAP and is a more difficult problem to solve. However, many studies in the literature propose heuristic algorithms for GAP. Agent qualifications are examined firstly with this study for the MRBGAP. Due to the NP hardness of the problem, a multi- start iterated tabu search algorithm is proposed for the solution of the problem. In addition, proposed heuristic algorithm is compared with TS algorithm. According to experimental comparisons, ILS algorithm yielded better results than TS algorithm. ILS algorithm found better results than the MILP model in a shorter time for 10 agents. With this study, large problem instances are generated for MRBGAP. This study is the first to use ILS algorithm for the MRBGAP. Future studies will focus on matheuristic algorithm, other meta heuristic algorithms and stochastic version of the problem.

References

[1] Pentico, D.W. (2007). Assignment problems: A golden anniversary survey. *European Journal of Operational Research*, 176, 774- 793.

[2] Mazzola, J. B., & Wilcox, S.P. (2001). Heuristics for the multi- resource generalized assignment problem. *Naval Research Logistics*, 48, 468- 483.

[3] Karsu, Ö., & Azizoğlu, M. (2012). The multi-resource agent bottleneck generalised assignment problem. *International Journal of Production Research*, 5(2), 309- 324.

[4] Ross, G. T. (1975). A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming*, 8, 91- 103.

[5] Posta, M., Ferland, J.A., & Michelon, P. (2012). An exact method with variable fixing for solving the generalized assignment problem. *Computational Optimization and Applications*, 52, 629- 644.

[6] Avella, P., Boccia, M., & Vasilyev, I. (2010). A computational study of exact knapsack separation

for the generalized assignment problem. *Computational Optimization and Applications*, 45, 543- 555.

[7] Savelsbergh, M. (1997). A branch- and- price- algorithm for the generalized assignment problem. *Operations Research*, 831- 841.

[8] Pigatti, A., Poggi, M., & Uchoa, E. (2005). Stabilized branch- and- cut- and- price for the generalized assignment problem. *Electronic Notes in Discrete Mathematics*, 19, 389- 395.

[9] Öncan, T., Şuvak, Z., Akyüz, M. H., & Altınel, İ. K. (2019). Assignment problem with conflicts. *Computers and Operations Research* (In press).

[10] Wu, W., Iori, M., Martello, S., & Yagiura, M. (2018). Exact and heuristic algorithms for the interval min- max regret generalized assignment problem. *Computers and Industrial Engineering*, 125, 98- 110.

[11] Öncan, T. (2007). A survey of the generalized assignment problem and its applications. *Information Systems and Operational Research*, 45 (3), 123- 141.

[12] Higgins, A. J. (2001). A dynamic tabu search for large- scale generalised assignment problems. *Computers and Operations Research*, 28, 1039- 1048.

[13] Diaz, J. A., & Fernandez, E. (2001). A tabu search heuristic for the generalized assignment problem. *European Journal of Operational Research*, 132, 22- 38.

[14] Osman, İ. (1995). Heuristics for the generalized assignment problem: simulated annealing and tabu search approaches. *Operations Research- Spektrum*, 17, 211- 225.

[15] Chu, P. L., & Beasley, J. E. (1997). A genetic algorithm for the generalized assignment problem. *Computers and Operations Research*, 24 (1), 17- 23.

[16] Ozbakır, L., Baykasoğlu, A., & Tapkan, P. (2010). Bees algorithm for generalized assignment problem. *Applied Mathematics and Computation*, 11, 3782- 3795.

[17] Jeet, V., & Kutanoglu, E. (2007). Lagrangian relaxation guided problem space search heuristics for generalized assignment problem. *European Journal of Operational Research*, 182 (3), 1039- 1056.

[18] Litvinchev, I., Mata, M., Rangel, S., & Saucedo, J. (2010). Lagrangian heuristic for a class of the generalized assignment problems. *Computers and Mathematics with Applications*, 60 (4), 1115- 1123.

[19] French, A. P., & Wilson, J. M. (2007). An LP- based heuristic procedure for the generalized assignment problem with special ordered sets. *Computers and Operations Research*, 34 (8), 2359- 2369.

[20] Souza, D. S., Santos H. G., & Coelho, I. M. (2017). A hybrid heuristic in GPU- CPU based on scatter search for the generalized assignment problem. *Procedia Computer Science*, 108, 1404- 1413.

[21] Sethanan, K., & Pitakaso, R. (2016). Improved differential evolution algorithms for solving generalized assignment problem. *Expert Systems with Applications*, 45, 450- 459.

[22] Liu, Y. Y., & Wang, S. (2015).A scalable parallel genetic algorithm for the generalized assignment problem. *Parallel Computing*, 46, 98- 119.

[23] Degroote, H., Velarde, J., & Causmaecker, P. (2018). Applying algorithm selection- a case study for the generalized assignment problem. *Electric Notes in Discrete Mathematics*, 69, 205- 212.

[24] Chakravarthy, V. K., Ramana, V., & Umashankar, C. (2018). Investigation of task bottleneck generalized assignment problems in supply chain optimization using heuristic techniques. *IOSR Journal of Business and Management*, 20 (5), 41- 47.

[25] Fadaei, S., & Bichler, M. (2017). Generalized assignment problem: truthful mechanism design without money. *Operations Research Letters*, 45 (1), 72-76

[26] Sahni, S., & Gonzalez, T. (1976). P-complete approximation problems. *Journal of the Association for Computing Machinery*, 23, 556- 565.

[27] Yagiura, M., Komiya, A., Kojima, K., Nonobe, K., Nagamochi, H., Ibaraki, T., & Glover, F. (2007). A path- relinking approach for the multi- resource generalized quadratic assignment problem. *Engineering stochastic local search algorithms, designing, implementing and analyzing effective heuristics*, 4638, 121- 135.

[28] Gavish, B., & Pirkul, H. (1991). Algorithms for the multi- resource generalized assignment problem. *Management Science*, 37 (6).

[29] Yagiura, M., Iwasaki, S., Ibaraki, T., & Glover, F. (2004). A very large- scale neighborhood search algorithm for the multi- resource generalized assignment problem. *Discrete Optimization*, 1, 87- 98.

[30] Mitrovic- Minic, S., & Punnen, A. (2009). Local search intensified: Very large scale variable neighborhood search for the multi- resource generalized assignment problem. *Discrete Optimization*, 6, 370- 377.

[31] Moussavi, S. E., Mahdjoub, M., & Grunder, O. (2018). A hybrid heuristic algorithm for the sequencing generalized assignment problem in an assembly line. *IFAC- Papers OnLine*, 51(2), 695- 700.

[32] Qin, T., Peng, B., Benlic, U., Cheng, T. C. E., Wang, Y., & Lü, Z. (2015). Iterated local search based on multi- type perturbation for single- machine earliness/ tardiness scheduling. *Computers and Operations Research*, 61, 81- 88.

[33] Ibaraki, T., Imahori, S., Nonobe, K., Sobue, K., Uno, T., & Yagiura, M. (2008). An iterated local search algorithm for the vehicle routing problem with convex time penalty functions. *Discrete Applied Mathematical*, 156 (11), 2050- 2069.

[34] Subramanian, A., Drummond, L. M. D. A., Bentes, C., Ochi, L. S., & Farias, R. (2010). A parallel heuristic for the vehicle routing problem with simultaneous pickup and delivery. *Computers and Operations Research*, 37 811), 1899- 1911.

[35] Penna, P. H. V., Subramanian, A., & Ochi, L. S. (2013). An iterated local search heuristic for the heterogeneous fleet vehicle routing problem. *Journal of Heuristics*, 19 (2), 201- 232.

[36] Michallet, J., Prins C., Amodeo, L., Yalaoui, F., & Vitry, G. (2014). Multi- start iterated local search for the periodic vehicle routing problem with time

Windows and time spread constraints on services. *Computers and Operations Research*, 41, 196- 207.

[37] Stützle, T. (2006). Iterated local search for the quadratic assignment problem. *European journal of Operations Research*, 174 (3), 1519- 1539.

[38] Avcı, M., & Topaloglu, S. (2017). A multi- start iterated local search algorithm for the generalized quadratic multiple knapsack problem. *Computers and Operations Research*, 83, 54- 65.

[39] Guan, J., Lin, G., & Feng, H. (2018). A multi- start iterated local search algorithm for the uncapacitated single allocation hub location problem. *Applied Soft Computing*, 73, 230- 241

[40] Meignan, D., & Knust, S. (2019). A neutrality- based local search for shift scheduling optimization and interactive reoptimization. *European Journal of Operational Research*, 279 (2), 320- 334.

[41] Mozdgir, A., Mahdavi, I., Badeleh, I. S., & Solimanpur, M. (2013). Using the Taguchi method to optimize the differential evolution algorithm parameters for minimizing the workload smoothness index in simple assembly line balancing. *Mathematical and Computer Modelling*, 57 (1-2), 137- 151.

Gülçin Bektur *received her MSc (2013) and PhD (2018) degrees from Department of Industrial engineering, Eskisehir Osmangazi University, Turkey. Her research areas include optimization and heuristic search. She is an Assistant Professor at the Department of Industrial engineering, Iskenderun Technical University.*

http://orcid.org/0000-0003-4313-7093

Appendices

Table A1. Five agent and 75 or 100 jobs (* denotes optimal solutions)

				MILP			ILS			TS		
n	pij1	c	hij	Solution		CPU	Solution	CPU	%Error	Solution	CPU	%Error
75	U[5,25]	1,4	1	**697,35**	*	0,49	**697,35**	45,22	0	699,77	85,79	0,35
75	U[5,25]	1,4	0,7	**728,3**	*	1,08	**728,3**	27,41	0	729,64	40,42	0,19
75	U[5,25]	1,6	1	**574,27**	*	1,31	**574,27**	48,98	0	575,07	82,75	0,14
75	U[5,25]	1,6	0,7	**814,46**	*	1,2	**814,46**	20,13	0	**814,46**	42,42	0
75	U[15,25]	1,4	1	**1153,9**	*	0,81	**1153,9**	35,26	0	**1153,9**	82,19	0
75	U[15,25]	1,4	0,7	**1219,56**	*	1,7	**1219,56**	22,19	0	1220,53	45,42	0,08
75	U[15,25]	1,6	1	**1157,64**	*	1,42	**1157,64**	39,84	0	1169,14	88,46	1
75	U[15,25]	1,6	0,7	**978,86**	*	1,03	**978,86**	20,46	0	**978,86**	42,46	0
75	U[25,35]	1,4	1	**1806,47**	*	3,56	**1806,47**	41,21	0	1812,16	85,64	0,32
75	U[25,35]	1,4	0,7	**1915,14**	*	2,57	**1915,14**	25,12	0	**1915,14**	41,53	0
75	U[25,35]	1,6	1	**1770,45**	*	2,7	**1770,45**	41,54	0	1775,76	86,79	0,3
75	U[25,35]	1,6	0,7	**1921,74**	*	1,25	**1921,74**	21,89	0	1948,96	48,86	1,42
100	U[5,25]	1,4	1	**814,47**	*	2,8	814,96	81,69	0,06	818,29	102,85	0,47
100	U[5,25]	1,4	0,7	**917,57**	*	1,76	**917,57**	67,88	0	917,57	85,69	0
100	U[5,25]	1,6	1	**860,59**	*	5,53	**860,59**	82,87	0	861,32	105,18	0,09
100	U[5,25]	1,6	0,7	**1008,28**	*	0,89	**1008,28**	61,95	0	1008,28	80,25	0
100	U[15,25]	1,4	1	**1522,29**	*	5,44	**1522,29**	89,22	0	1522,49	109,18	0,02
100	U[15,25]	1,4	0,7	**1616,18**	*	1,43	**1616,18**	69,28	0	1628,33	81,43	0,76
100	U[15,25]	1,6	1	**1537,62**	*	2,18	**1537,62**	87,87	0	1545,82	106,17	0,54
100	U[15,25]	1,6	0,7	**1645,26**	*	2,06	**1645,26**	68,51	0	1649,96	88,88	0,29
100	U[25,35]	1,4	1	**2451,96**	*	5,5	**2451,96**	86,54	0	2463,93	101,6	0,49
100	U[25,35]	1,4	0,7	**2514,61**	*	42,54	**2514,61**	64	0	**2514,61**	87,67	0
100	U[25,35]	1,6	1	**2435,7**	*	4,41	**2435,7**	83,75	0	2451,57	107,62	0,66
100	U[25,35]	1,6	0,7	**2645,9**	*	3,59	**2645,9**	55,88	0	2647,86	80,86	0,08

Table A2. Ten agent and 150 or 200 jobs

				MILP		ILS			TS		
n	pij1	c	hij	Solution	CPU	Solution	CPU	%Error	Solution	CPU	%Error
150	U[5,25]	1,4	1	**500,19**	3600	501,18	1656,85	0,2	501,68	2258,5	0,3
150	U[5,25]	1,4	0,7	568,62	3600	**562,13**	1258,45	-1,14	570,8	1874,6	0,38
150	U[5,25]	1,6	1	**487,87**	3600	489,23	1845,74	0,28	499,49	2156,8	2,38
150	U[5,25]	1,6	0,7	**585,68**	3600	588,74	1152,41	0,52	587,16	1745,7	0,25
150	U[15,25]	1,4	1	**1068,17**	3600	1069,5	1985,12	0,12	1071,63	2275,3	0,32
150	U[15,25]	1,4	0,7	1115,84	3600	**1114,14**	1158,46	-0,15	1120,07	1856,4	0,38
150	U[15,25]	1,6	1	**1062,19**	3600	1064,89	1585,45	0,25	1074,3	2041,9	1,14
150	U[15,25]	1,6	0,7	**1111,28**	3600	1114,23	985,12	0,27	1119,24	1765,2	0,72
150	U[25,35]	1,4	1	1683,83	3600	**1683,55**	1289,13	-0,02	1684,24	2256,5	0,02
150	U[25,35]	1,4	0,7	1766,41	3600	**1762,5**	1258,45	-0,22	1784,49	1946,4	1,02
150	U[25,35]	1,6	1	**1685,86**	3600	**1684,28**	1365,75	-0,09	1720,34	2178,7	2,04
150	U[25,35]	1,6	0,7	**1770,04**	3600	1770,95	1058,43	0,05	1798,02	1845,5	1,58
200	U[5,25]	1,4	1	**655,5**	3600	660,76	2156,52	0,8	**658,02**	2985,4	0,38
200	U[5,25]	1,4	0,7	**748,35**	3600	752,63	1985,63	0,57	762,46	2441,1	1,89
200	U[5,25]	1,6	1	**666,2**	3600	670,19	2045,12	0,6	669,13	2874,3	0,44
200	U[5,25]	1,6	0,7	769,65	3600	**766,38**	1756,42	-0,42	789,1	2045,5	2,53
200	U[15,25]	1,4	1	1425,66	3600	**1425,28**	2378,41	-0,03	1425,37	2796,1	-0,02
200	U[15,25]	1,4	0,7	1494,56	3600	**1494,46**	1974,23	-0,01	1499,15	2248,6	0,31
200	U[15,25]	1,6	1	1432,5	3600	**1431,54**	2685,42	-0,07	1498,45	2213,9	4,6
200	U[15,25]	1,6	0,7	1465,53	3600	1468,01	1545,78	0,17	1496,42	2016,2	2,11
200	U[25,35]	1,4	1	2251,16	3600	**2250,43**	1985,45	-0,03	2289,15	2845,3	1,69
200	U[25,35]	1,4	0,7	2338,99	3600	2339,41	1845,12	0,02	2339,95	2156,2	0,04
200	U[25,35]	1,6	1	2234,67	3600	**2232,55**	2156,45	-0,1	2256,78	2845,3	0,99
200	U[25,35]	1,6	0,7	2330,87	3600	2333,75	1585,12	0,12	**2328,86**	2045,9	-0,09