2017

A Tabu Search Heuristic for a Generalized Quadratic Assignment Problem

Alan McKendall

West Virginia University, alan.mckendall@mail.wvu.edu

Chihui Li

Follow this and additional works at: https://researchrepository.wvu.edu/faculty_publications

Part of the [Operations Research, Systems Engineering and Industrial Engineering Commons](#)

Digital Commons Citation

McKendall, Alan and Li, Chihui, "A Tabu Search Heuristic for a Generalized Quadratic Assignment Problem" (2017). *Faculty & Staff Scholarship*. 3058.

https://researchrepository.wvu.edu/faculty_publications/3058

This Article is brought to you for free and open access by The Research Repository @ WVU. It has been accepted for inclusion in Faculty & Staff Scholarship by an authorized administrator of The Research Repository @ WVU. For more information, please contact ian.harmon@mail.wvu.edu.

A tabu search heuristic for a generalized quadratic assignment problem

Alan McKendall & Chihui Li

To cite this article: Alan McKendall & Chihui Li (2017) A tabu search heuristic for a generalized quadratic assignment problem, Journal of Industrial and Production Engineering, 34:3, 221-231, DOI: [10.1080/21681015.2016.1253620](https://doi.org/10.1080/21681015.2016.1253620)

To link to this article: <http://dx.doi.org/10.1080/21681015.2016.1253620>

Published online: 18 Nov 2016.

Submit your article to this journal [↗](#)

Article views: 30

View related articles [↗](#)

View Crossmark data [↗](#)

A tabu search heuristic for a generalized quadratic assignment problem

Alan McKendall and Chihui Li

Department of Industrial & Management Systems Engineering, West Virginia University, Morgantown, WV, USA

ABSTRACT

The generalized quadratic assignment problem (GQAP) is the task of assigning a set of facilities to a set of locations such that the sum of the assignment and transportation costs is minimized. The facilities may have different space requirements, and the locations may have varying space capacities. Also, multiple facilities may be assigned to each location such that space capacity is not exceeded. In this paper, an application of the GQAP is presented for assigning a set of machines to a set of locations on the plant floor. Construction algorithms and a simple tabu search heuristic are developed for the GQAP. A set of test problems available in the literature was used to evaluate the performances of the TS heuristic using different construction algorithms. The results show that the simple TS heuristic is effective for solving the GQAP.

ARTICLE HISTORY

Received 23 December 2015
Revised 22 September 2016
Accepted 17 October 2016

KEYWORDS

Generalized quadratic assignment problem; tabu search; metaheuristic; construction algorithms

1. Introduction

The machine layout problem is the task of assigning N machines to N locations, on the plant floor of a manufacturing facility, such that the sum of the material handling and installation/assignment costs is minimized. For this problem, the plant floor is represented as an array of N equal size grid units, each having enough capacity to store any of the N machines. Therefore, this problem can be modeled as a quadratic assignment problem (QAP). The QAP was introduced by Koopmans and Beckmann [1], and was proven to be NP Hard by Sahni and Gonzales [2]. See Burkard et al. [3] and Loiola et al. [4] for an extensive review of the solution techniques for the QAP.

A generalization of the machine layout problem defined above is to assign one or more machines to each location on the plant floor such that the plant floor may be represented as an array of unequal-area grids. More specifically, M machines, which may have different space (area) requirements, are assigned to N locations of varying space capacities ($M > N$) such that the capacities of the locations are not exceeded. This extended machine layout problem can be modeled as a generalized quadratic assignment problem (GQAP). The GQAP is the task of assigning a set of facilities to a set of locations such that the sum of the assignment and transportation costs is minimized. The facilities may have different space requirements, and the locations may have varying space capacities. Also, multiple facilities may be assigned to each location such that space capacity is not exceeded. Lee and Ma [5] first introduced the GQAP to solve the problem of assigning a set of equipment to a set of

locations. More specifically, they considered the problem where a manufacturing company decided to locate multiple equipment types at different manufacturing sites such that the sum of the costs of moving intermediate parts between sites, where a sequence of operations are performed on the parts, and the costs of assigning the equipment to the sites is minimized. Cordeau et al. [6] presented an application which uses the GQAP to manage containers in a storage yard (assign M container groups to N storage areas). Similarly, Cordeau et al. [7] considered the service allocation problem as a GQAP with side constraints. Unal and Uysal [8] used the GQAP to design a curriculum at a university. Although there are many applications of the GQAP, in this paper, the GQAP is used to assign a set of machines to a set of locations such that the capacities of the locations are not violated while minimizing the sum of the assignment and transportation (or material handling) costs.

The formulation of the GQAP is given below and is an adaptation of the model presented by Lee and Ma [5].

$$\text{Minimize } z = \sum_{i=1}^M \sum_{k=1}^N a_{ik} x_{ik} + \sum_{i=1}^M \sum_{j=1}^M \sum_{k=1}^N \sum_{l=1}^N c_{ijkl} f_{ij} d_{kl} x_{ik} x_{jl} \quad (1)$$

$$\text{s.t. } \sum_{k=1}^N x_{ik} = 1, i = 1, \dots, M \quad (2)$$

$$\sum_{i=1}^M r_i x_{ik} \leq C_k, k = 1, \dots, N \quad (3)$$

$$x_{ik} = \{0, 1\}, i = 1, \dots, M, k = 1, \dots, N \quad (4)$$

where M is the number of machines, N is the number of locations, a_{ik} is the cost of assigning (installing) machine i to (at) location k , f_{ij} is the units of materials transported from machine i to machine j , d_{kl} is the distance from location k to location l , c_{ijkl} is the unit cost per distance unit of moving materials from machine i (at location k) to machine j (at location l), r_i is the space requirement of machine i , and C_k is the amount of space capacity available at location k . The decision variables are defined as:

$$x_{ik} = \begin{cases} 1, & \text{if machine } i \text{ is assigned to location } k, \\ 0, & \text{otherwise.} \end{cases}$$

Objective function (1) minimizes the sum of the assignment/installation and material handling costs. Constraint set (2) ensures that each machine is assigned to only one location. Constraint set (3) ensures that the space capacity of each location is not exceeded, and the restrictions on the decision variables are given in (4).

The term in objective function (1) used to obtain material handling cost has a quadratic term (i.e. product of two variables). As a result, the mathematical formulation (1)–(4) is nonlinear and is called a nonlinear programming model. The model is linearized by substituting w_{ijkl} for $x_{ik}x_{jl}$. Then, replace objective function (1) with

$$\text{Minimize } z = \sum_{i=1}^M \sum_{k=1}^N a_{ik}x_{ik} + \sum_{i=1}^M \sum_{j=1}^M \sum_{k=1}^N \sum_{l=1}^N c_{ijkl}f_{ij}d_{kl}w_{ijkl} \quad (1a)$$

and add the following constraints:

$$x_{ik} + x_{jl} - 1 \leq w_{ijkl} \quad \forall i, j \neq i \text{ and } \forall k, l \neq k \quad (5)$$

$$w_{ijkl} \geq 0 \quad \forall i, j \neq i \text{ and } \forall k, l \neq k \quad (6)$$

As a result, the linearized model (i.e. a mixed integer linear programming model) for the GQAP consists of objective function (1a) subject to constraints (2)–(6). This model will be used in the next section to solve a small GQAP instance.

As stated previously, the QAP was proven to be NP Hard (Sahni and Gonzales [2]), and the GQAP is a generalization of the QAP. Therefore, the GQAP is computationally intractable, and it is impossible to solve large-size problems, in reasonable time, using the formulation presented above. Therefore, heuristics are presented to solve large-size problems quickly. It is important to note that since space is modeled one-dimensionally, instead of two-dimensionally, the solutions (or assignments) generated may be impractical for certain real-world problems. For example, consider a GQAP where machines 1 and 2 are assigned to location 1. The one-dimensional space (i.e. area) requirements of the machines are 9 and 6 units of space, and the one-dimensional space (i.e. area) capacity of location 1 is 16 units of space. Obviously this assignment is feasible, since 15

units of space are required, where 16 units of space are available. However, this assignment may be infeasible when considering the actual dimensions of the machines, i.e. two-dimensional space (length $\times$ width). Consider the case where the same machines require a 3×3 and a 2×3 block of space (total area of 15 units of space), and the location is a 2×8 block of space (total area of 16 units of space). The assignment is infeasible, since the 3×3 machine cannot fit within the 2×8 location. To overcome this limitation, a more general GQAP needs to be considered. Alternatively, once a solution is obtained using the one-dimensional GQAP, the assignments of machines to locations are obtained. These assignments along with the dimensions of the machines and locations can be used as input data for the continuous facility layout problem (FLP) presented in McKendall et al. [9]. The FLP considers two-dimensional space requirements for the machines. As a result, the output is the actual locations of the machines either within their assigned locations or reassigned locations as in the example mentioned previously, since the original assignment is infeasible.

Unlike the QAP, the GQAP literature is very limited. In other words, there are only a few papers which consider the GQAP. Lee and Ma [5] presented the first formulation for the GQAP. Also, the authors presented three methods for the linearization of the formulation, and a branch and bound algorithm to optimally solve the GQAP. Hahn et al. [10] presented a new algorithm based on a reformulation linearization technique (RLT) dual ascent procedure to optimally solve the GQAP. Similarly, Pessoa et al. [11] presented two exact algorithms for the GQAP which combine a previously proposed branch and bound scheme with a new Lagrangean relaxation procedure over a known RLT formulation. These exact methods are unable to solve large size problems in reasonable time. Therefore, approximation (or heuristic) methods were presented for the GQAP. Cordeau et al. [6] presented a linearization of the GQAP formulation as well as a memetic heuristic for the GQAP, which combines genetic algorithms (Holland [12]) and tabu search (Glover [13]). Mateus et al. [14] proposed several GRASP (greedy randomized adaptive search procedures) with path-relinking heuristics for the GQAP using different construction, local search, and path-relinking procedures (GRASP-PR). Since these heuristic methods presented for the GQAP may be difficult for researchers and practitioners to implement (i.e. to understand, code, or find good heuristic parameter settings), the aim of this research was to present simple but effective solution techniques. Therefore, in this paper, simple construction algorithms and a simple tabu search (TS) heuristic is presented for the GQAP, which obtain good solutions.

There are many successful and diverse applications of the TS heuristic. Here are just a few of them. McKendall and Jaramillo [15] presented a TS heuristic for the dynamic space allocation problem. The problem was to assign maintenance activities and their required

resources to workspaces and their idle resources to storage locations inside the reactor containment building during planned outages at a nuclear power plant. McKendall and Hakobyan [16] developed a TS heuristic to design the layout of unequal-area facilities on the plant floor during a multi-period planning horizon. McKendall and Liu [17] presented TS heuristics with different diversification and intensification strategies for the dynamic FLP. More recently, Liang et al. [18] compared a TS heuristic with their multiple-level variable neighborhood search technique for an orienteering problem. Yousefikhoshbakht et al. [19] presented a modified TS heuristic to solve the vehicle routing problem with simultaneous pickup and delivery. Wang et al. [20] developed a TS heuristic to minimize talent cost and operating cost during film production. In this paper, a TS heuristic is developed for the proposed GQAP.

The paper is organized as follows. In Section 2, an illustrative example is presented and solved using the mathematical formulation presented above. Afterward, the construction algorithms and a simple TS heuristic are presented for the GQAP in Section 3. In Section 4, the computational results of the proposed heuristics on a set of test problems taken from the literature are given, and then concluding remarks are given in Section 5.

2. Illustrative example

Consider a GQAP where six machines are assigned to four locations on the plant floor. First, the space requirements of each machine are determined by obtaining the footprints, personnel space needed, and material storage requirements for each machine. The space calculations are summarized in Table 1 below. Notice machine 2 requires 120 ft² of space.

Next, the plant floor is divided into unequal-area grids based on barriers, columns, floor loadings, etc. The plant

floor configuration is given in Figure 1. The dimensions, area capacities, and centers of the locations (centroids) for the four locations (sites) are given in Table 2. For instance, location 2 is 20-feet long by 10-feet wide, and the centroid is at (25, 15). Therefore, location 2 has 200 ft² of capacity. Once the centroids are obtained, the distances between locations are calculated using the rectilinear distance measure. If (a_k, b_k) and (a_l, b_l) are the centroids for locations k and l , respectively, then the distance between the locations is $|a_k - a_l| + |b_k - b_l|$. See matrix d_{kl} for distances between locations.

The amount of materials (f_{ij}) flowing between each pair of machines i and j per month is obtained from route sheets and is given in Table 3. Also, the total cost of installing each machine i to each location k is calculated, and then the monthly equivalent cost (a_{ik}) is obtained and given in Table 3. Assume $c_{ijkl} = 1$ for all i, j, k , and l . Recall, c_{ijkl} is the unit cost per distance unit of moving materials from machine i (at location k) to machine j (at location l).

Using the CPLEX 11.0 solver and the linearized mathematical formulation for the GQAP presented above, which consist of objective function (1a) subject to constraints (2)–(6), the optimal solution is obtained for the illustrative example. Since the variables w_{ijkl} are

Figure 1. Plant floor configuration for production area.

Table 1. Calculations of space requirement (r_i) for each machine i .

Machine	Footprint	Area (ft ²)			
		Equipment	Personnel	Material	Total (r_i)
1	5 ft × 10 ft	50	20	20	90
2	6 ft × 10 ft	60	20	40	120
3	5 ft × 6 ft	30	20	50	100
4	5 ft × 8 ft	40	20	50	110
5	5 ft × 10 ft	50	20	40	110
6	5 ft × 6 ft	30	20	20	70
Total area required					600

Table 2. Calculations of capacity (C_k) of each location k , and distances (d_{kl}) between pairs of sites k and l .

Location	Dimension	Area (C_k)	Centroid	Site	1	2	3	4
1	15 ft × 20 ft	300 sq ft	(7.5, 10)	1	0	22.5	17.5	27.5
2	20 ft × 10 ft	200 sq ft	(25, 15)	2	22.5	0	15	15
3	10 ft × 10 ft	100 sq ft	(20, 5)	3	17.5	15	0	10
4	10 ft × 10 ft	100 sq ft	(30, 5)	4	27.5	15	10	0
Production area	35 ft × 20 ft	700 sq ft						

Table 3. Material flows (f_{ij}) between machines and installation costs (a_{ik}) of machines at sites data.

Machine	1	2	3	4	5	6	Machine	Site			
								1	2	3	4
1	0	33	41	9	16	56	1	700	1600	1900	1400
2	0	0	49	91	78	23	2	1300	1800	1700	800
3	0	0	0	41	38	44	3	800	1400	3000	1100
4	0	0	0	0	6	17	4	3000	800	700	1500
5	0	0	0	0	0	68	5	1200	1500	1300	1800
6	0	0	0	0	0	0	6	1700	800	1200	1100

used only to linearize the objective function, the values of these variables do not give any useful information and is not given here. However, $x_{13} = x_{21} = x_{34} = x_{42} = x_{51} = x_{61} = 1$, and all other decision variables are zero. The total cost of the solution (z^*) is \$17165 which is the sum of \$8000 (total installation cost) and \$9165 (total material handling cost). More specifically, machines 2, 5, and 6 are assigned to location 1. Machines 4, 1, and 3 are assigned to locations 2, 3, and 4, respectively. See display of optimal solution (or assignment) in Figure 2 below. Since each machine is assigned to a location, and the capacity of each location is not exceeded, the optimal solution obtained is feasible. See Table 4. For instance, machines 2, 5, and 6 require 120, 110, and 70 ft² of area, respectively. Thus, a total of 300 ft² of area is required for location 1, which has 300 ft² of area. Therefore, the remaining capacity (unused area remaining) is zero.

As discussed earlier, since the proposed GQAP considers one-dimensional space (area), the feasibility of the solution (i.e. the assignments of machines to each location) using two-dimensional space needs to be considered. Recall, the dimensions (length $\times$ width given in feet) of the machines and the locations were given in Tables 1 and 2, respectively. Notice, each of the locations are able to accommodate the assigned machine(s). For example, consider machines 2, 5, and 6 assigned to location 1. Machines 2, 5, and 6 require a 6×10 , 5×10 , and a 5×6 block of space, and location 1 is a 15×20 block of

Figure 2. Layout of production area.**Table 4.** Details of optimal solution.

Location	Area (ft ²)	Machine	Area (ft ²) required	Remaining capacity (ft ²)
1	300	2, 5, 6	300	0
2	200	4	110	90
3	100	1	90	10
4	100	3	100	0
Total	700		600	100

space. Therefore, the dimension of location 1 can accommodate the dimensions of the machines 2, 5, and 6. In addition, location 1 should be able to accommodate the dimensions of the corresponding personnel and materials. Hence, the solution is feasible.

In Table 4, notice that the total area required for all machines is 600 ft², and the total production area available is 700 ft², where 100 ft² is not used. As in Cordeau et al. [6], this information can be used to categorize the problem as easy or hard. More specifically, if the percentage of location area required is high, the problem is categorized as a hard problem; otherwise, the problem is considered easy. Since the percentage of location area required is 85.7 ($100 \times 600 / 700$), the illustrative example is categorized as a hard problem. Cordeau et al. [6] defined this as F = tightness of the capacity constraints, instead of the percentage of location area required. More importantly, when the problem is tightly constrained (i.e. percentage of location area required is high), there are less feasible solutions in the solution space, which will be discussed and illustrated below. However, when the problem is loosely constrained (i.e. percentage of location area required is low), there are more feasible solutions in the solution space. Cordeau et al. [6] also used M = number of machines and N = number of locations to further categorize problems. Besides the latter, Lee and Ma [5] also considered the range (wide or roughly the same) of the area requirements for the machines and the range of the location capacities to further categorize the problems. For the illustrative example, the area requirements for the machines range from 50 to 120 ft² (Range = 70), and the location capacities range from 100 to 300 ft² (Range = 200), in which both may be considered as wide, relatively speaking. Nevertheless, wide ranges for the machine area requirements and location capacities could further restrict the number of feasible solutions in the solution space, adding even more complexity to the problem. Next, construction algorithms and a TS heuristic is presented for the GQAP.

3. Solution techniques

3.1. Solution representation

Since the mathematical model can only be used to solve small-size problems in reasonable time, heuristics are developed for the GQAP. First, additional notation is defined for the combinatorial optimization problem

(COP) formulation of the GQAP so that heuristics can be used to solve the problem. The solution is represented as:

$$S = (s(1), s(2), \dots, s(M))$$

where $s(i) = k$, which is equivalent to saying machine i is assigned to location k . The solution is feasible if the following constraints hold,

$$\sum_{\forall i \text{ s.t. } s(i)=k} r_i \leq C_k \quad \text{for } k = 1, \dots, N \quad (7)$$

The total cost of the solution is obtained using the following objective function.

$$TC(S) = \sum_{i=1}^M a_{is(i)} + \sum_{i=1}^M \sum_{\substack{j=1 \\ j \neq i}}^M c_{ijs(i)s(j)} f_{ij} d_{s(i)s(j)} \quad (8)$$

For example, the optimal solution for the illustrative example presented above (see Tables 1–3) is represented as $S = (3, 1, 4, 2, 1, 1)$. That is, $s(1) = 3$, $s(2) = 1$, $s(3) = 4$, $s(4) = 2$, $s(5) = 1$, and $s(6) = 1$. More specifically, machines 2, 5, and 6 are assigned to location 1, machine 4 to location 2, machine 1 to location 3, and machine 3 to location 4. For location 1 (i.e. $k = 1$), consider constraint (7).

$$\sum_{\forall i \text{ s.t. } s(i)=1} r_i \leq C_1 \text{ or } \sum_{i=2,5,6} r_i \leq C_1$$

Recall, the space requirements for machines 2, 5, and 6 are $r_2 = 120$, $r_5 = 110$, $r_6 = 70$, and the space capacity of location 1 is $C_1 = 300$. Therefore,

$$\sum_{i=2,5,6} r_i = 120 + 110 + 70 = 300 \leq C_1 = 300$$

Thus, the capacity constraint for location 1 holds. It is easy to validate that the constraints for locations 2, 3, and 4 also hold. As a result, the solution $S = (3, 1, 4, 2, 1, 1)$ is feasible. Next, the total cost of the solution (\$17165) is obtained using objective function (8), where total installation cost is \$8000 and total material handling cost is \$9165.

This formulation which minimizes objective function (8) subject to constraint (7) is called the COP formulation. It has less variables (only variables $s(i)$), constraints, and solutions compared to the mathematical programming formulation presented earlier. See comparison of models in Table 5 where $M = 6$ and $N = 4$, as in the illustrative example. Notice the number of constraints in the mathematical model is based on the number of constraints for constraints (2), (3), and (5) where the restrictions on the variables are not considered. Also, the number of

possible solutions for both models considers all possible sets of values for each variable, whether the solutions are feasible or infeasible. More importantly, the solution space is much smaller for the COP model, only 4096 solutions, compared to the mathematical model, with 16,777,216 solutions. Therefore, it is much more efficient using the COP model, as opposed to the mathematical model, to solve the GQAP. Next, a construction algorithm for the GQAP is presented.

3.2. Construction algorithms

The following construction algorithm is used to generate a solution for the GQAP.

- Step 1: Initialize the capacities of the locations (i.e. $C = \{C(1), C(2), \dots, C(N)\}$. Initialize the space requirements of the machines (i.e. $r = \{r(1), r(2), \dots, r(M)\}$).
- Step 2: Sort machines in set r in descending order with respect to $r(i)$. Break ties based on smallest index (i.e. lowest machine number). Then put the sorted machine numbers in the eligible machine set (EMS). For example, if $r = \{5, 3, 5\}$, then $r(1) = r(3) = 5 > 3 = r(2)$. Since index $1 < 3$, $EMS = \{1, 3, 2\}$.
- Step 3a: Set $k = 1$ where k = location number;
- Step 3b: If $k > N$ (N = number of locations), then go to step 5b.
- Else, go to position 1 of EMS (i.e. set $p = 1$).
- Step 4: Set i = the machine in position p of set EMS.
- Step 5a: If $r(i) \leq C(k)$, then
- (1) Assign machine i to location k (i.e. set $s(i) = k$), and set $C(k) = C(k) - r(i)$;
 - (2) Remove machine i from EMS. If EMS is empty, then go to step 5b;
 - (3) If $C(k) < r(i)$ for $Last(i)$ in set EMS, then set $k = k + 1$, and go to step 3b;
- Else, go to step 4.
- Else, set $p = p + 1$, and go to step 4.
- Step 5b: Terminate algorithm. If EMS is empty, then display feasible solution S . Else, display "No feasible solution!"

To illustrate the construction algorithm, consider illustrative example presented earlier (see Tables 1–3).

Iteration 1: In step 1, let $C = \{300, 200, 100, 100\}$ and $r = \{90, 120, 100, 110, 110, 70\}$. In step 2, obtain $EMS = \{2, 4, 5, 3, 1, 6\}$, and set $k = 1$ (start with location 1) in step 3a. In step 3b, since $k = 1 < 4 = N$, set $p = 1$ (start at position 1 in EMS). Since machine 2 is in position 1, set $i = 2$ in step 4. In step 5a, since $r(2) = 120 < 300 = C(1)$, set $s(2) = 1$ (assign machine 2 to location 1), obtain $C(1) = C(1) - r(2) = 300 - 120 = 180$, and remove machine 2 from EMS. Thus, $EMS = \{4, 5, 3, 1, 6\}$. Since $C(1) = 180 > 70 = r(6)$, go to step 4.

Iteration 2: In step 4, set $i = 4$. In step 5a, since $r(4) = 110 < 180 = C(1)$, set $s(4) = 1$, obtain $C(1) = C(1) - r(4) = 180 - 110 = 70$, and obtain $EMS = \{5, 3, 1, 6\}$; Since $C(1) = 70 = r(6)$, go to step 4.

Table 5. Comparison of the models.

$M = 6, N = 4$	Math model	COP model
Number of variables	$M(N) = 24$	$M = 6$
Number of constraints	$M + N + M(M-1)(N(N-1)) = 370$	$N = 4$
Number of solutions	$2^{M(N)} = 16,777,216$	$N^M = 4096$

Iteration 3: In step 4, set $i = 5$. In step 5a, since $r(5) = 110 > 70 = C(1)$, set $p = 1 + 1 = 2$, and go to step 4.

Iteration 4: In step 4, set $i = 3$. In step 5a, since $r(3) = 100 > 70 = C(1)$, set $p = 2 + 1 = 3$, and go to step 4.

Iteration 5: In step 4, set $i = 1$. In step 5a, since $r(1) = 90 > 70 = C(1)$, set $p = 3 + 1 = 4$, and go to step 4.

Iteration 6: In step 4, set $i = 6$. In step 5a, since $r(6) = 70 = C(1)$, set $s(6) = 1$, obtain $C(1) = C(1) - r(6) = 70 - 70 = 0$, and obtain $EMS = \{5, 3, 1\}$. Since $C(1) = 0 < 90 = r(1)$, set $k = k + 1 = 2$, and go to step 3b.

Iteration 7: In step 3b, since $k = 2 < 4 = N$, set $p = 1$, and set $i = 5$ in step 4. In step 5a, since $r(5) = 110 < 200 = C(2)$, set $s(5) = 2$, obtain $C(2) = C(2) - r(5) = 200 - 110 = 90$, and obtain $EMS = \{3, 1\}$. Since $C(2) = 90 = r(1)$, go to step 4.

Iteration 8: In step 4, set $i = 3$. In step 5a, since $r(3) = 100 > 90 = C(2)$, set $p = 1 + 1 = 2$, and go to step 4.

Iteration 9: In step 4, set $i = 1$. In step 5a, since $r(1) = 90 = C(2)$, set $s(1) = 2$, obtain $C(2) = C(2) - r(1) = 90 - 90 = 0$, and obtain $EMS = \{3\}$. Since $C(2) = 0 < 100 = r(3)$, set $k = 2 + 1 = 3$, and go to step 3b.

Iteration 10: In step 3b, since $k = 3 < 4 = N$, set $p = 1$, and set $i = 3$ in step 4. In step 5a, since $r(3) = 100 = C(3)$, set $s(3) = 3$, obtain $C(3) = C(3) - r(3) = 100 - 100 = 0$, and obtain $EMS = \{\}$. Since EMS is empty, go to step 5b. In step 5b, the algorithm is terminated, and the solution $S = \{2, 1, 3, 1, 2, 1\}$ is displayed. In other words, machines 2, 4, and 6 are assigned to location 1, machines 1 and 5 are at location 2, machine 3 is at location 3, and location 4 is not assigned a machine.

The algorithm presented above (called construction algorithm 1) either yields a feasible solution or no solution. A solution may be difficult to obtain when the feasible solution space is restricted due to a relatively high percentage of location area required (i.e. problem is tightly constrained), as discussed earlier. In addition, the feasible solution space is further restricted when there are wide ranges in the machine requirements and the location capacities. As a result, the above algorithm can be modified such that in step 2 the machines can be ordered in ascending instead of descending order. Also, as opposed to starting at the first location ($k = 1$), the algorithm starts at the last location ($k = N$), and k is reduced until all machines are assigned to locations. The different combinations of these modifications result in four different construction algorithms. However, for construction algorithm 2 the machines are ordered based on their machine number in step 2. For construction algorithm 3, in step 2 the machines are ordered randomly and assigned to locations randomly. Therefore, a solution can always be generated for the GQAP using one of these construction algorithms. However, only construction algorithms 1, 2, and 3 will be used to generate diverse initial solutions for the proposed TS heuristic. Next, a local neighborhood search technique is presented for the GQAP, which is embedded within the proposed TS heuristic.

3.3. Local neighborhood search technique

The local neighborhood search technique, called the steepest descent local search heuristic, is embedded within the proposed TS heuristic and can be used to improve the solutions obtained from the construction algorithms presented above. The steepest descent local search heuristic for the GQAP is given as follows.

- Step 1: Construct a solution, $S_0 = (s(1), s(2), \dots, s(M))$, using either construction algorithm 1, 2, or 3, and obtain its cost, $TC(S_0)$ using objective function (8).
- Step 2: Evaluate all feasible solutions obtained from all possible drop/add operations on S_0 and all possible pairwise exchange operations on S_0 .
- Step 3: Pick best operation/move (i.e. move*) which gives best feasible neighboring solution, S , with respect to total cost, $TC(S)$. If $TC(S) < TC(S_0)$, set $S_0 = S$, $TC(S_0) = TC(S)$, and go to step 2. Else, terminate heuristic and display solution S_0 .

The drop/add operation ($u, v; v'$) represents exchanging location v assigned to machine u with location v' (drop v and add v'). For example, if $S_0 = \{2, 1, 3, 1, 2, 1\}$, as obtained from using construction algorithm 1 to solve the above illustrative example, then the drop/add operation (1, 2; 4) produces the solution $\{4, 1, 3, 1, 2, 1\}$. In other words, machine 1 assigned to location 2 is reassigned to location 4. Constraint (7), where $k = 4$, can be used to check for feasibility of the solution $\{4, 1, 3, 1, 2, 1\}$. Since solution is feasible, the objective function value (OFV) of the solution is obtained using (8). All possible drop/add operations for each machine is considered. Since there are $N = 4$ locations, and each machine is already assigned to one location, there are $N - 1 = 3$ possible drop/add operations for each machine. Since there are $M = 6$ machines, there are $M(N - 1) = 6(3) = 18$ possible drop/add operations. See Table 6 for all possible drop/add operations on solution $S_0 = \{2, 1, 3, 1, 2, 1\}$. Notice only 3 out

Table 6. Solutions obtained from drop/add operations.

#	Operation	Solution	OFV	#	Operation	Solution	OFV
1	(1, 2; 1)	{1, 1, 3, 1, 2, 1}	–	10	(4, 1; 2)	{2, 1, 3, 2, 2, 1}	–
2	(1, 2; 3)	{3, 1, 3, 1, 2, 1}	–	11	(4, 1; 3)	{2, 1, 3, 3, 2, 1}	–
3	(1, 2; 4)	{4, 1, 3, 1, 2, 1}	\$21,580	12	(4, 1; 4)	{2, 1, 3, 4, 2, 1}	–
4	(2, 1; 2)	{2, 2, 3, 1, 2, 1}	–	13	(5, 2; 1)	{2, 1, 3, 1, 1, 1}	–
5	(2, 1; 3)	{2, 3, 3, 1, 2, 1}	–	14	(5, 2; 3)	{2, 1, 3, 1, 3, 1}	–
6	(2, 1; 4)	{2, 4, 3, 1, 2, 1}	–	15	(5, 2; 4)	{2, 1, 3, 1, 4, 1}	–
7	(3, 3; 1)	{2, 1, 1, 1, 2, 1}	–	16	(6, 1; 2)	{2, 1, 3, 1, 2, 2}	–
8	(3, 3; 2)	{2, 1, 2, 1, 2, 1}	–	17	(6, 1; 3)	{2, 1, 3, 1, 2, 3}	–
9	(3, 3; 4)	{2, 1, 4, 1, 2, 1}	\$20,695	18	(6, 1; 4)	{2, 1, 3, 1, 2, 4}	\$20,495

of the 18 possible drop/add operations produced feasible solutions. This is not surprising, since the problem is tightly constrained and is categorized as a hard problem.

The pairwise exchange operation $(u, v; u', v')$ represents exchanging location v assigned to machine u with location v' assigned to machine u' . If $S_0 = \{2, 1, 3, 1, 2, 1\}$, then the pairwise exchange operation $(1, 2; 3, 3)$ produces the solution $\{3, 1, 2, 1, 2, 1\}$. In other words, machine 1 assigned to location 2 exchanges locations with machine 3 assigned to location 3. Constraints (7) where $k = 2$ and 3 can be used to check for feasibility of solution $\{3, 1, 2, 1, 2, 1\}$. Since solution is infeasible for $k = 2$, the OFV of the solution is not calculated using objective function (8). See Table 7 for all possible pairwise exchange operations on solution $S_0 = \{2, 1, 3, 1, 2, 1\}$. Since two machines are swapping locations and there are $M = 6$ machines, there are a combination of $M = 6$ pick two $(M(M-1)/2 = 15)$ possible pairwise exchange operations. However, if $v = v'$, the solution does not change and is not considered. For instance, if $S_0 = \{2, 1, 3, 1, 2, 1\}$, then the operation $(1, 2; 5, 2)$ produces the same solution $\{2, 1, 3, 1, 2, 1\}$; therefore, there are always less than $M(M-1)/2$ pairwise exchange operations when a location has more than one machine assigned to it. Again, notice only 1 out of the 15 possible pairwise exchange operations produced a feasible solution.

To illustrate the steepest descent local search heuristic, consider the illustrative example presented earlier (see Tables 1–3). In step 1 of iteration 1, the solution $S_0 = \{2, 1, 3, 1, 2, 1\}$ is obtained using construction algorithm 1, and the total cost (OFV) of the solution S_0 (i.e. $TC(S_0) = \$21,255$) is obtained using objective function (8). In step 2, all possible solutions are obtained for S_0 . See Tables 6 and 7. In step 3, $move^* = (4, 1; 5, 2)$ produces the feasible solution $S = \{2, 1, 3, 2, 1, 1\}$, which has the lowest cost (i.e. $TC(S) = \$18,050$). Since $TC(S) = \$18,050 < \$21,255 = TC(S_0)$, set $S_0 = S = \{2, 1, 3, 2, 1, 1\}$, $TC(S_0) = \$18,050$, and go to step 2. In step 2, all possible drop/add and pairwise exchange operations are performed on solution S_0 . In step 3 of iteration 2, $move^* = (3, 3; 4)$ produces the feasible solution $S = \{2, 1, 4, 2, 1, 1\}$, which has the lowest cost (i.e.

$TC(S) = \$17,460$). Since $TC(S) = \$17,460 < \$18,050 = TC(S_0)$, set $S_0 = S = \{2, 1, 4, 2, 1, 1\}$, $TC(S_0) = \$17,460$, and go to step 2. Again, all possible drop/add and pairwise exchange operations are performed on solution S_0 in step 2. In step 3 of iteration 3, $move^* = (1, 2; 3)$ produces the feasible solution $S = \{3, 1, 4, 2, 1, 1\}$, which has the lowest cost (i.e. $TC(S) = \$17,165$). Since $TC(S) = \$17,165 < \$17,460 = TC(S_0)$, set $S_0 = S = \{3, 1, 4, 2, 1, 1\}$, $TC(S_0) = \$17,165$, and go to step 2. In step 2, all possible drop/add and pairwise exchange operations are performed on solution S_0 . In step 3 of iteration 4, $move^* = (6, 1; 2)$ produces the feasible solution $S = \{3, 1, 4, 2, 1, 2\}$, which has the lowest cost (i.e. $TC(S) = \$17,240$). Since $TC(S) = \$17,240 > \$17,165 = TC(S_0)$, terminate heuristic and display solution $S_0 = \{3, 1, 4, 2, 1, 1\}$, $TC(S_0) = \$17,165$. This solution is called a local optimum; however, since the solution is equivalent to the solution obtained using the mathematical model, it is also a global optimum. See Table 8 for an outline of the results obtained using the steepest descent local search technique on the illustrative example.

As stated earlier, the proposed methods stay within the feasible solution space during the search for neighboring solutions. In contrast, the heuristics presented in the literature (memetic in [6] and GRASP-PR in [14]) for the GQAP allow infeasible neighboring solutions (i.e. capacity constraint(s) may not be satisfied) into the search space. Next, a simple TS heuristic is presented for the GQAP.

3.4. Tabu Search (TS) heuristic

The TS heuristic was first presented by Glover[13]. Also, see Glover [21–23]. The basic idea of TS is to improve a solution iteratively, using some guiding rules such as recency (short-term) memory as well as intensification and diversification strategies to obtain good solutions in complex solution spaces. The proposed TS heuristic uses the steepest descent local search heuristic presented above with short-term memory (or recency-based memory) to accept uphill moves. In other words, the most recent moves are forbidden, enabling the heuristic to climb out of the valley (i.e. accept non-improving moves after obtaining a local optimum) in search of the global optima. In the GQAP, if the best feasible solution in the neighborhood of the current solution S_0 is S , and S is obtained by the drop/add move $move^* = (u, v; v')$, then

Table 7. Solutions obtained from pairwise exchange operations.

#	Operation	Solution	OFV	#	Operation	Solution	OFV
1	(1, 2; 2, 1)	{1, 2, 3, 1, 2, 1}	–	9	(2, 1; 6, 1)	Same as S_0	–
2	(1, 2; 3, 3)	{3, 1, 2, 1, 2, 1}	–	10	(3, 3; 4, 1)	{2, 1, 1, 3, 2, 1}	–
3	(1, 2; 4, 1)	{1, 1, 3, 2, 2, 1}	–	11	(3, 3; 5, 2)	{2, 1, 2, 1, 3, 1}	–
4	(1, 2; 5, 2)	Same as S_0	–	12	(3, 3; 6, 1)	{2, 1, 1, 1, 2, 3}	–
5	(1, 2; 6, 1)	{1, 1, 3, 1, 2, 2}	–	13	(4, 1; 5, 2)	{2, 1, 3, 2, 1, 1}	\$18,050
6	(2, 1; 3, 3)	{2, 3, 1, 1, 2, 1}	–	14	(4, 1; 6, 1)	Same as S_0	–
7	(2, 1; 4, 1)	Same as S_0	–	15	(5, 2; 6, 1)	{2, 1, 3, 1, 1, 2}	–
8	(2, 1; 5, 2)	{2, 2, 3, 1, 1, 1}	–				

Table 8. Results of the steepest decent heuristic on illustrative example.

Iter #	S_0	$TC(S_0)$	$move^*$	S	$TC(S)$
1	{2, 1, 3, 1, 2, 1}	\$21,255	(4, 1; 5, 2)	{2, 1, 3, 2, 1, 1}	\$18,050
2	{2, 1, 3, 2, 1, 1}	\$18,050	(3, 3; 4)	{2, 1, 4, 2, 1, 1}	\$17,460
3	{2, 1, 4, 2, 1, 1}	\$17,460	(1, 2; 3)	{3, 1, 4, 2, 1, 1}	\$17,165
4	{3, 1, 4, 2, 1, 1}	\$17,165	(6, 1; 2)	{3, 1, 4, 2, 1, 2}	\$17,240

this move is tabu restricted for a certain number of iterations, called tabu list size (*tabusize*). The tabu status and tabu list size of each move are maintained in the tabu list structure $tabu[u][v]$. For example, consider the results obtained in Table 8. In iteration 1, the best move $move^* = (4, 1; 5, 2)$, which yields the best feasible solution (lowest OFV) in the neighborhood of the current solution S_0 . Therefore, $move^* = (4, 1; 5, 2)$ is tabu until $iter = tabusize + iter$ and update tabu list: $tabu[4][1] = tabu[5][2] = tabusize + iter$. That is, the move, exchanging location 1 assigned to machine 4 with location 2 assigned to machine 5, can be performed again when $iter = tabusize + iter + 1 = tabusize + 2$, since current iteration is iteration 1 (i.e. $iter = 1$). Similarly, in iteration 2, since $move^* = (3, 3; 4)$, $tabu[3][3] = tabusize + iter = tabusize + 2$.

Sometimes a move which is tabu restricted may improve the best solution found thus far. Therefore, the aspiration criterion is used to override the tabu restriction of a move when the move improves the best found solution thus far (S^*). More specifically, if $move^*$ had been performed recently, is tabu restricted, and it improves the best solution found thus far (i.e. $TC(S) < TC(S^*)$), then the aspiration criterion is used to override its tabu restriction. Any move which is acceptable (i.e. non-tabu move and tabu move overridden by aspiration criterion) is defined as an admissible move. Hence, $move^*$ is defined as the best admissible move. The best admissible move is either a non-tabu move that generates the lowest cost feasible solution out of all the possible non-tabu moves at the current iteration or a tabu move that produces the lowest cost feasible solution found thus far, considering all the iterations performed. A simple TS heuristic for the GQAP is outlined below.

- Step 1: Initialize parameters and counters:
 Set $tabu[u][v] = 0$, for all $u = 1, \dots, M$ and $v = 1, \dots, N$;
 Set *tabusize*, where *tabusize* = tabu tenure length;
 Set *MaxIter*, where *MaxIter* = maximum number of iterations before terminating heuristic;
 Set $iter = 0$, where *iter* = iteration number;
- Step 2: Obtain an initial solution $S_0 = (s(1), s(2), \dots, s(M))$ using either construction algorithm 1, 2, or 3. Obtain $TC(S_0)$ using objective function (8).
- Step 3: Set $S^* = S_0$, where S^* is the best solution found thus far, and set $TC(S^*) = TC(S_0)$;
- Step 4: Set $iter = iter + 1$;
 Find the best admissible move, $move^*$, which gives best feasible solution (S) in neighborhood S_0 .
 Recall, find best feasible solution (S) by considering all possible drop/add and pairwise exchange operations on S_0 .
- Step 5: Set $S_0 = S$ and $TC(S_0) = TC(S)$;
 If $TC(S_0) < TC(S^*)$, set $S^* = S_0$ and $TC(S^*) = TC(S_0)$.

Step 6: Update tabu list as either $tabu[u][v] = iter + tabusize$, if drop/add operation $(u, v; v')$ gives best admissible move $move^*$ or $tabu[u][v] = tabu[u'][v'] = iter + tabusize$, if pairwise exchange operation $(u, v; u', v')$ gives best admissible move $move^*$;

Step 7: Stopping criterion: If $iter < MaxIter$, go to Step 4. Else, terminate the heuristic and return solution S^* and total cost of solution $TC(S^*)$.

The only TS heuristic parameters are *tabusize* and *MaxIter*. Recall, *tabusize* is the number of iterations a move is declared tabu restricted. The value of this parameter is based mainly on the number of feasible solutions in the neighborhood of the current solution S_0 (or similarly, the number of moves which produces a feasible solution different from S_0). As stated previously, the total number of drop/add and pairwise exchange moves is $M(N-1) + M(M-1)/2$. This is referred to as the size of the neighborhood of the solution S_0 . Recall, many of the solutions produced from these moves (or operations) may be infeasible. Also, some pairwise exchange operations may produce the same current solution S_0 . As a result, the parameter *tabusize* is determined by considering the range of integer values between multiplying the size of the neighborhood by 0.005 and by 0.02. Considering each construction algorithm, the integer value, which obtains the best solution the quickest (in less iterations) in this range, is the value chosen for the parameter *tabusize*. Smaller values of *tabusize* tend to result in poor solutions, since the TS heuristic either spends too much time repeating the same moves, or may get trapped at a local optima (i.e. cycling). Otherwise, using a large value for *tabusize* results in restricting the solution space too much whereby many potentially good moves may be overlooked, resulting in poor solutions. The value of the parameter *MaxIter* is between 5000 and 5,000,000 (a multiple of 5000) based on the problem size and problem complexity (easy or hard). See computational section of the paper.

Recall, the objective of this paper was to present simple construction algorithms and a simple TS heuristic for solving large-size GQAP instances. More importantly, these heuristics are simple enough for managers and researchers to understand and apply. Furthermore, the advantages of the proposed heuristics over the heuristics available in the literature for the GQAP (i.e. the memetic heuristic presented by Cordeau et al. [6] and GRASP-PR heuristics presented by Mateus et al. [14]) are as follows:

- (1) The proposed TS heuristic is much easier to understand and easier to implement/code compared to the memetic and GRASP-PR heuristics.
- (2) The proposed TS heuristic has only two heuristic parameters versus the nine for the memetic and more than ten parameters for GRASP-PR.
- (3) The proposed TS heuristic forces the search to stay within the feasible solution space, which

restricts the search space and does not require complicated procedures to “fix” or “repair” infeasible solutions to make them feasible as in the other methods.

Considering advantage (3) above, one might assume that limiting the search to only the feasible solution space may restrict the simple TS heuristic too much such that it is unable to obtain “good solutions.” On the contrary, the simple TS heuristic was able to compete with the other heuristics presented in the literature. This will be further discussed in the next section.

4. Computational results

A set of test problems taken from Cordeau et al. [6] is used in this paper to evaluate the performance of the proposed TS heuristic using the three construction algorithms presented in Section 3.2. The set of test problems contains problem instances with 20, 30, 35, 40, and 50 facilities (or machines) with locations ranging between 6 and 20 such that the percentages of location area required (i.e. tightness of the capacity constraint) ranges between 35 and 95. The proposed solution techniques were programmed using Visual Basic programming language, and the data-set was solved on a 2.92 GHz PC.

In order to find the best solutions possible, the heuristic parameter *MaxIter* is set to 5,000,000. See Table 9 for a summary of the results. The first column gives the description of the problem instance. For example, the first problem instance requires assigning 20 machines to 15 locations such that the tightness of the capacity constraint is 35. In column 2, the tabu list size is given for each problem instance, which was obtained as discussed in section 3.4. Columns 3, 5, and 7 give the total cost (OFV) of the initial solutions obtained using construction algorithms 1, 2, and 3, respectively. The initial

solutions were improved using the proposed TS heuristic and the final OFVs are given in columns 4, 6, and 8. The bolded solutions are the best found solutions available in the literature. Notice construction algorithms 1 and 3 were unable to generate initial solutions for two of the problem instances where the tightness of the capacity constraint is 95. Although construction algorithm 2 was able to generate initial solutions for the two problem instances, the proposed TS heuristic was unable to obtain the best found solution for one of the instances, namely problem instance 30-20-95. Even though, construction algorithm 2 was able to obtain initial solutions for all of the problem instances, the proposed TS heuristic was unable to find the best found OFVs available in the literature for 4 out of the 21 instances. The proposed TS heuristic with construction algorithm 1 obtained the best OFVs for all, with the exception of one, of the instances in which initial solutions were obtained. The proposed TS heuristic with construction algorithm 3 (randomly assigning machines to locations) performed best (i.e. able to find best OFV in literature for all problems in which initial solutions were obtained).

Next, a comparison of the proposed TS heuristic with the other algorithms available in the literature for the GQAP (i.e. the memetic heuristic presented by Cordeau et al. [6] and GRASP-PR heuristics presented by Mateus et al. [14]) is given. Cordeau et al. [6] programmed their memetic heuristic using C, and ran the test problems on a 1.2 GHz Sun workstation. Mateus et al. [14] programmed their GRASP-PR heuristics using Java, and ran the test problems on a 2.66 GHz Intel Xeon processor. Since Mateus et al. [14] terminated their GRASP-PR heuristics when the best OFVs available in the literature were obtained, the set of test problems was re-solved using the proposed TS heuristic with the same stopping criterion. However, the proposed TS heuristic was unable to obtain the best OFVs for some of the problem instances using

Table 9. Summary of results of proposed construction algorithms and TS heuristic.

Instance	tabu size	Initial	Final	Initial	Final	Initial	Final
		Sol 1	Sol 1	Sol 2	Sol 2	Sol 3	Sol 3
20-15-35	8	3,570,402	1,471,896	3,296,339	1,471,896	3,120,364	1,471,896
20-15-55	8	3,934,838	1,723,638	3,811,156	1,723,638	3,846,386	1,723,638
20-15-75	7	3,863,736	1,953,188	3,697,904	1,967,315	3,693,511	1,953,188
30-06-95	7	7,858,015	5,160,920	8,502,834	5,160,920	8,460,410	5,160,920
30-07-75	7	8,321,507	4,383,923	6,972,953	4,383,923	8,299,575	4,383,923
30-08-55	7	6,798,564	3,501,695	7,756,799	3,501,695	7,067,847	3,501,695
30-10-65	11	7,712,596	3,620,959	7,446,408	3,620,959	8,069,676	3,620,959
30-20-35	12	7,750,035	3,379,359	6,564,745	3,633,644	7,287,830	3,379,359
30-20-55	12	8,093,573	3,593,105	8,921,966	3,599,602	7,989,876	3,593,105
30-20-75	11	8,657,772	4,050,938	8,146,803	4,050,938	8,491,849	4,050,938
30-20-95	12			8,627,569	5,726,530		
35-15-35	12	10,753,280	4,457,348	10,150,194	4,456,670	10,594,321	4,456,670
35-15-55	10	10,997,026	4,639,128	10,790,017	4,639,128	9,472,924	4,639,128
35-15-75	11	11,570,933	6,301,723	12,170,135	6,301,723	11,621,892	6,301,723
35-15-95	12			12,581,614	6,670,264		
40-07-75	11	11,981,541	7,405,793	12,467,450	7,405,793	13,067,437	7,405,793
40-09-95	8	14,015,031	7,667,719	14,780,911	7,667,719	13,338,268	7,667,719
40-10-65	10	13,199,022	7,265,559	15,547,370	7,265,559	14,588,924	7,265,559
50-10-65	10	23,447,369	10,513,029	25,416,364	10,513,029	22,822,818	10,513,029
50-10-75	8	22,010,509	11,217,503	21,806,545	11,217,503	21,080,820	11,217,503
50-10-95	10	25,008,645	12,845,598	23,195,085	12,845,598	24,583,846	12,845,598

Table 10. Comparison of the proposed TS heuristic with other heuristics.

Instance	Solution	Memetic	GRASP-PR	TS
		Time (sec)	Avg time (sec)	Avg time (sec)
20-15-35	1,471,896	96	7.09	0.05
20-15-55	1,723,638	102	2.88	0.07
20-15-75	1,953,188	102	2.02	250.97
30-06-95	5,160,920	114	2.6	0.08
30-07-75	4,383,923	156	7.86	0.05
30-08-55	3,501,695	96	1.62	0.05
30-10-65	3,620,959	210	122.55	4.05
30-20-35	3,379,359	564	79.43	1970.23
30-20-55	3,593,105	462	25.28	1473.98
30-20-75	4,050,938	522	41.64	0.69
30-20-95	5,726,530	2652	55,580.31	9.98
35-15-35	4,456,670	456	307.65	1605.68
35-15-55	4,639,128	384	21.24	0.21
35-15-75	6,301,723	396	68.57	0.47
35-15-95	6,670,264	864	1551.78	1.03
40-07-75	7,405,793	180	59.66	0.58
40-09-95	7,667,719	1140	419.08	4.57
40-10-65	7,265,559	240	17.96	0.16
50-10-65	10,513,029	504	24.68	1.4
50-10-75	11,217,503	606	1384.36	3.29
50-10-95	12,845,598	1254	89.36	25.99
Avg time		528.6	2848.5	254.9

construction algorithms 1 and 2, as discussed above. As a result, the proposed TS heuristic uses two stopping conditions, namely, terminate heuristic when either the best OFV is obtained or $MaxIter = 5,000,000$. See Table 10 for a summary of the results. Column 2 gives the best OFVs obtained by the memetic heuristic, except for instance 30-20-95, where Cordeau et al. [6] obtained a better OFV of 5,710,645. Neither the GRASP-PR heuristics nor the proposed TS heuristic were able to obtain this OFV (or better). In column 3, the best computational time is given for each of the OFVs obtained in column 2 using the memetic heuristic. For each instance, Mateus et al. [14] made 200 independent runs of GRASP-PR, with the exception of instances 30-20-95, 35-15-95, and 50-10-75, where they made only 30, 59, and 120 runs, respectively. As stated previously, each run stopped when the OFV in column 2 was found. Although they gave the minimum, maximum, and average computational times, only the average computational times are given for the GRASP-PR in column 4. In column 5, the average run times required, for the proposed TS heuristic to solve each of the instances using construction algorithms 1, 2, and 3, are given. Note, the proposed TS heuristic performed only 3 runs compared to GRASP-PR performing a minimum of 30 and a maximum of 200 runs. Also, the overall average computational time of solving the problem instances using the proposed TS heuristic is 254.9 vs. 2848.5 s for the GRASP-PR heuristics. Since the memetic computational times were obtained using a different stopping criterion, it cannot be compared with the computational times of the proposed TS heuristic or the GRASP-PR heuristics.

5. Conclusion

In this paper, three construction algorithms and a simple TS heuristic were presented for the GQAP. Also,

a comparison of the proposed TS heuristic with the heuristics available in the literature was given. The proposed solution techniques performed well on a data-set available in the literature and were able to find the best found OFVs for all but one problem instance. The proposed solution techniques are easy to understand and implement (code), require setting only two heuristic parameters, and remains within the feasible solution space during its search for a “good” local optimum, unlike the methods presented in the literature. For future research, one could develop other meta-heuristics and hybrid heuristics for the GQAP that remain within the feasible solution space during the search process. These developed heuristics could be modified such that the solutions are allowed to go outside the feasible solution space during the search process. A comparison of the performances of these heuristics should be considered to determine the limitations of both types of methods (i.e. having the search remain within the feasible solution space or not).

Disclosure statement

No potential conflict of interest was reported by the authors.

Notes on contributors

Alan McKendall is an associate professor in the Department of Industrial and Management Systems Engineering at West Virginia University. He received his PhD and MS degrees in Industrial Engineering and MS degree in Applied Mathematics from the University of Missouri-Columbia. He received his BS degree in Mathematics from Southern University in New Orleans. His research interests include facilities layout, logistics, scheduling, data mining, and optimization.

Chihui Li is a senior industrial engineer in Michelin (China) Investment Limited. He received his BS degree in Industrial Engineering from Guangdong University of Technology, China, and his MS degree in Industrial Engineering from West Virginia University. His present research interests include facilities layout, lean manufacturing, and data mining.

References

- [1] Koopmans TC, Beckmann MJ. Assignment problems and the location of economic activities. *Econometrica*. 1957;25:53–76.
- [2] Sahni S, Gonzalez T. P-complete approximation problems. *J ACM*. 1976;23:555–565.
- [3] Burkard RE, Cela E, Pardalos PM, et al. The quadratic assignment problem. In: Du D-Z, Pardalos PM, editors. *Handbook of combinatorial optimization*. Boston (MA): Kluwer Academic Publishers; 1998. p. 241–338.
- [4] Loiola EM, de Abreu NMM, Boaventura-Netto PO, et al. A survey for the quadratic assignment problem. *Eur Oper Res*. 2007;176:657–690.
- [5] Lee C-G, Ma Z. The generalized quadratic assignment problem. Technical Report. Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, ON; 2004.
- [6] Cordeau J-F, Gaudioso M, Laporte G, et al. A memetic heuristic for the generalized assignment problem. *INFORMS J Comput*. 2006;18:433–443.

- [7] Cordeau J-F, Gaudioso M, Laporte G, et al. The service allocation problem at the Gioia Tauro maritime terminal. *Eur Oper Res.* **2007**;176:1167–1184.
- [8] Unal YZ, Uysal O. A new mixed integer programming model for curriculum balancing: application to a Turkish university. *Eur J Oper Res.* **2014**;238:339–347.
- [9] Mckendall AR, Noble JS, Klein CM. Facility layout of irregular-shaped departments using a nested approach. *Int J Prod Res.* **1999**;37:2895–2914.
- [10] Hahn PM, Kim B-J, Guignard M. An algorithm for the generalized quadratic assignment problem. *Comput Optim Appl.* **2008**;40:351–372.
- [11] Pessoa AA, Hahn PM, Guignard M, et al. Algorithms for the generalized quadratic assignment problem combining lagrangean decomposition and the reformulation-linearization technique. *Eur J Oper Res.* **2010**;206:54–63.
- [12] Holland JH. *Adaptation in natural and artificial systems.* Ann Arbor (MI): University of Michigan Press; **1975**.
- [13] Glover F. Future paths for integer programming and links to artificial intelligence. *Comput Oper Res.* **1986**;13:533–549.
- [14] Mateus GR, Resende MGC, Silva RMA. GRASP with path-relinking for the generalized quadratic assignment problem. *J Heuristics.* **2011**;17:527–565.
- [15] McKendall AR, Jaramillo JR. A tabu search heuristic for the dynamic space allocation problem. *Comput Oper Res.* **2006**;32:107–125.
- [16] McKendall AR, Hakobyan A. Heuristics for the dynamic facility layout problem with unequal-area departments. *Eur J Oper Res.* **2010**;201:171–182.
- [17] McKendall AR, Liu W-H. New tabu search heuristics for the dynamic facility layout problem. *Int J Prod Res.* **2012**;50:867–878.
- [18] Liang Y-C, Kulturel-Konak S, Lo M-H. A multiple-level variable neighborhood search approach to the orienteering problem. *J Ind Prod Eng.* **2013**;30:238–247.
- [19] Yousefikhoshbakht M, Didehvar F, Rahmati F. A combination of modified tabu search and elite ant system to solve the vehicle routing problem with simultaneous pickup and delivery. *J Ind Prod Eng.* **2014**;31:65–75.
- [20] Wang S-Y, Chuang Y-T, Lin BMT. Minimizing talent cost and operating cost in film production. *J Ind Prod Eng.* **2016**;33:17–31.
- [21] Glover F. Tabu search Part I. *ORSA J Comput.* **1989**;1:1900–2006.
- [22] Glover F. Tabu search part II. *ORSA J Comput.* **1990**;2:4–32.
- [23] Glover F. Tabu search: a tutorial. *Interfaces.* **1990**;20:74–94.