GRASP WITH PATH-RELINKING FOR FACILITY LAYOUT

R. M. A. SILVA, M. G. C. RESENDE, P. M. PARDALOS, G. R. MATEUS, AND G. DE TOMI

Abstract. This paper proposes a mathematical formulation for the facility layout problem (FLP) based on the generalized quadratic assignment problem (GQAP). The GQAP is a generalization of the NP-hard quadratic assignment problem (QAP) that allows multiple facilities to be assigned to a single location as long as the capacity of the location allows. As a case study, we adapt the GRASP with path-relinking (GRASP-PR) heuristic introduced in Mateus et al. (2011) for the hospital layout problem (HLP). In the HLP we are given a set of areas in a hospital where medical facilities, such as surgery and recovery rooms, can be located and a set of medical facilities, each facility with a required area, that must be located in the hospital. Furthermore, we are given a matrix specifying, for each ordered pair of facilities, the number of patients that transition from the first to the second facility. The objective of the assignment is to minimize the total distance traveled by the patients. We illustrate our algorithm with a numerical example.

1. Introduction

The facility layout problem (FLP) consists in assigning facilities to locations such that the total area of the facilities assigned to a location does not exceed the available area of the location. Among all feasible assignments, we seek one that minimizes the sum of products of flows between facility pairs and distances between locations to which the facility pairs are assigned. More formally, let $N = \{1, \ldots, n\}$ denote the set of facilities and $M = \{1, \ldots, m\}$ the set of locations. Furthermore, denote by $A_{n\times n} = (a_{ii'})$ the flow between facilities $i \in N$ and $i' \in N$, such that $a_{ii'} \in \Re^+$ if $i \neq i'$ and otherwise $a_{ii'} = 0$, by $B_{m\times m} = (b_{jj'})$ the distance between locations $j \in M$ and $j' \in M$, such that $b_{jj'} \in \Re^+$ if $j \neq j'$ and otherwise $b_{jj'} = 0$, and by $C_{n\times m} = (c_{ij})$, the cost of assigning facility $i \in N$ to location $j \in M$, such that $c_{ij} \in \Re^+$. Let $z \in \Re^+$ be a scaling factor called the unit traffic cost, $q_i \in \Re^+$ be the area demanded by facility $i \in N$, and $Q_j \in \Re^+$, the total available area of location $j \in M$. The FLP can be modelled as a generalized quadratic assignment problem (GQAP). It consists in finding $X_{n\times m} = (x_{ij})$, with $x_{ij} = \{0, 1\}$, where facility $i \in N$ is assigned to location $j \in M$ if and only if $x_{ij} = 1$, such that the constraints

$$\sum_{j\in M} x_{ij} = 1, \forall i \in N, \tag{1}$$

$$\sum_{i\in N} q_i x_{ij} \leq Q_j, \forall j \in M, \tag{2}$$

Date: January 20, 2013.

Key words and phrases. Hospital layout, generalized quadratic assignment problem, GRASP, path-relinking, heuristics.

AT&T Labs Research Technical Report.

$$x_{ij} \in \{0,1\}, \forall i \in N, \forall j \in M$$

are satisfied and the objective function

$$\sum_{i\in N}\sum_{j\in M} c_{ij}x_{ij} + z\sum_{i\in N}\sum_{j\in M}\sum_{i'\in N, i'\neq i}\sum_{j'\in M} a_{ii'}b_{jj'}x_{ij}x_{i'j'}$$

is minimized. Constraints (1) guarantee that each facility is assigned to exactly one location, while constraints (2) ensure that location capacities are not violated.

The facility layout problem (FLP) was first modelled as a quadratic assignment problem by Koopmans and Beckmann (1957), as a linear integer programming problem by Lawler (1963), as a quadratic set covering problem by Bazaraa (1975), as a mixed integer programming problem by Kaufman and Broeckx (1978), and as a graph theory problem by Foulds and Robinson (1976) and Rosenblatt (1979), among others (Kusiak and Heragu, 1987).

The main drawback of the quadratic assignment model is that it does not take into account that facilities can have different dimensions and that more than one facility can be assigned to a single location as long as the location can accommodate them. Approaches based on integer programming can only handle very small instances of the FLP. This paper models the FLP as a GQAP, thus satisfying this more realistic requirement.

We present a GRASP with path-relinking heuristic for the hospital layout problem (HLP), a FLP where medical facilities, such as intensive care units, surgery rooms, radiology rooms, and physician offices, must be feasibly located in a hospital. A numerical example shows the applicability of the proposed GRASP-PR heuristic for this hospital layout problem.

The paper is organized as follows. In Section 2, we review the GRASP with path-relinking procedure proposed in Mateus et al. (2011). The numerical example is described in Section 3. Concluding remarks are made in Section 4.

2. GRASP with path-relinking

A GRASP (Feo and Resende, 1989; 1995; Resende and Ribeiro, 2010) is a multi-start heuristic where at each iteration a greedy randomized solution is constructed to be used as a starting solution for local search. Local search repeatedly substitutes the current solution by a better solution in the neighborhood of the current solution. Each such replacement is called a *move*. If there is no better solution in the neighborhood, the current solution is declared a local minimum and the search stops. The best local minimum found over all GRASP iterations is output as the solution. One way to incorporate memory into GRASP is with path-relinking (Glover, 1996; Ribeiro and Resende, 2012). In GRASP with path-relinking (Laguna and Martí, 1999; Resende and Ribeiro, 2005), an elite set of diverse good-quality solutions is maintained to be used during each GRASP iteration. After a solution is produced with greedy randomized construction and local search, that solution is combined with a randomly selected solution from the elite set using the path-relinking operator. The combined solution is a candidate for inclusion in the elite set and is added to the elite set if it meets certain quality and diversity criteria.

Mateus et al. (2011) introduced a GRASP with path-relinking for the GQAP. Algorithm 1 shows pseudo-code for this algorithm. The algorithm takes as input the set N of facilities, the set M of locations, the flow matrix A, the distance

Data : $N, M, A, B, C, z, q_i, Q_j$.
Result: Feasible solution p^*.
$P \leftarrow \emptyset$;
while *stopping criterion not satisfied* **do**
  $p \leftarrow$ `GreedyRandomized`$(\cdot)$;
  if *elite set P has enough elements* **then**
    if *p is not feasible* **then**
      Randomly select a new solution $p \in P$;
    end
    $p \leftarrow$ `LocalSearch`(p);
    Randomly select a solution $q \in P$
    $r \leftarrow$ `PathRelinking`(p, q);
    if *elite set P is full* **then**
      if $c(r) \leq \max\{c(s) \mid s \in P\}$ *and* $r \not\approx P$ **then**
        replace the element most similar to r among all
        elements with cost worse than r;
      end

    else if $r \not\approx P$ **then**
      $P \leftarrow P \cup \{r\}$;
    end

  else if *p is feasible and* $p \not\approx P$ **then**
    $P \leftarrow P \cup \{p\}$;
  end
end
return $p^* = \mathbf{argmin}\{c(s) \mid s \in P\}$;

Algorithm 1: A GRASP with path-relinking heuristic for the FLP.

matrix B, the assignment cost matrix C, the scaling factor z, the facility demands $q_i,\ i \in N$, and the location capacities $Q_j,\ j \in M$, and outputs a feasible solution p^* specifying the location of each facility in the best solution found. After initializing the elite set P as empty, the GRASP with path-relinking iterations are computed until a stopping criterion is satisfied. This criterion could be, for example. a maximum number of iterations, a target solution quality, or a maximum number of iterations without improvement. During each iteration, a greedy randomized solution is generated and local search is applied using it as a starting point, resulting in a solution p. If the greedy randomized solution is infeasible, a feasible solution is randomly selected from the elite set and local search is applied on this solution. Path-relinking is applied between p and some elite solution q only if the elite set has at least a minimum number of elements. Otherwise, solution p is simply added to the elite set if it is sufficiently different from the solutions already in the elite set. To more precisely define the term *sufficiently different*, let the *symmetric difference* $\Delta(x, y)$ between two solutions x and y be defined as the number of moves needed to transform x into y or vice-verse. For a given level of difference δ, we say x is sufficiently different from all elite solutions in P if $\Delta(x, p) > \delta$ for all $p \in P$, which we indicate by the notation $x \not\approx P$. If the elite set is not yet full, the solution r resulting from path-relinking is added to the elite set if $r \not\approx P$. Otherwise, if r is not

of worse quality than any elite solution and $r \not\approx P$, then it will be added to the elite set in place of some elite solution. Among all elite solutions having cost no better than that of r, the one most similar to r, i.e. with smallest symmetric difference with respect to r, is selected to be removed from the elite set. At the end, the best elite set solution is output as the solution of the GRASP with path-relinking heuristic.

2.1. **Greedy randomized construction.** The construction procedure builds a solution one assignment at time. Suppose a partial solution is on hand, i.e. a number of assignments have already been made. To make the next assignment, the procedure needs to select a new facility and a location. Locations are made available, one at time. The procedure randomly determines whether to use a new location or a previously chosen location, favoring a new location when the previously chosen locations have insufficient or barely sufficient available capacity. If the procedure determines that a previously chosen location is to be selected, it then determines which facilities can be assigned to that location with the maximum available capacity and randomly selects one of these facilities to be assigned. Of the locations that can accommodate this facility, one is selected at random and the assignment is made. On the other hand, if there is no previously chosen location with sufficient capacity or if the available capacity is barely sufficient, a new location is selected at random from the set of yet unchosen locations.

The above procedure is not guaranteed to produce a feasible solution. The greedy randomized construction procedure, shown in Algorithm 2, addresses this difficulty by repeatedly applying the steps described above. The main loop in lines 1 to 21 is repeated a maximum number of times or until all facilities are assigned, i.e. when $F = \emptyset$. In each iteration of the procedure, the working sets are initialized in line 2 and the threshold probability is set to 1 in line 3. The purpose of the threshold is to control whether a new location should be selected. Since it is initially set to 1, then in the first iteration of the until loop in lines 4 to 19, the procedure always selects a new (first) location. At each iteration of the until loop, the threshold is updated in line 17 such that it will be more likely that a new location is selected when there are few facilities that can be assigned to locations in the current set R. The until loop consists of two stages. With probability equal to the threshold, the first stage (lines 5 to 9) selects a new location in line 6, updates the sets L and CL in line 7, and in line 8 determines the set T of facilities that can be assigned to some selected location. In the second stage (lines 10 to 18), the procedure randomly selects a facility from set T in line 11, updates the sets T, F, and CF in line 12, creates the location set R in line 13, randomly selects a location from that set in line 14, makes the assignment of the facility to the location in line 15, determines the set T of facilities that can be assigned to some selected location in line 16, and updates the threshold probability in line 17. The until loop is repeated until both sets T and L are empty in line 19. The while loop ends either with a valid assignment in line 25 (indicated by $F = \emptyset$) or with no solution found determined in line 23.

2.2. **Approximate local search.** The construction procedure of Subsection 2.1 produces a feasible solution p that is not guaranteed to be locally optimal. A local search procedure is applied starting at p to find an approximate local minimum. The local search procedure makes use of two neighborhood structures which we call

Data : $\bar{t}$ = *maximum number of tries*
Result: Solution $x \in \mathcal{X}$
while $k < \bar{t}$ **and** $F \neq \emptyset$ **do**
  $F \leftarrow N$; $CF \leftarrow \emptyset$; $L \leftarrow M$; $CL \leftarrow \emptyset$; $T \leftarrow \emptyset$;
  Set `threshold` $\leftarrow 1$;
  repeat
    if $L \neq \emptyset$ **and** `random`$([0,1]) \leq$ `threshold` **then**
      Randomly select a location $l \in L$;
      Update sets $L \leftarrow L \setminus \{l\}$ and $CL \leftarrow CL \cup \{l\}$;
      Set $T \subseteq F$ to be all facilities with demands less than or equal to the maximum slack in CL;
    end
    if $T \neq \emptyset$ **then**
      Randomly select a facility $f \in T$;
      Update sets $T \leftarrow T \setminus \{f\}$; $F \leftarrow F \setminus \{f\}$; and $CF \leftarrow CF \cup \{f\}$;
      Create set $R \subseteq CL$ to be all locations having slack greater than or equal to demand of facility f;
      Randomly select a location $l \in R$;
      Assign facility f to location l;
      Set $T \subseteq F$ to be all facilities with demands less than or equal to the maximum slack in CL;
      Set `threshold` $\leftarrow 1 - \mid T \mid / \mid F \mid$;
    end
  until $T = \emptyset$ **and** $L = \emptyset$;
  $k \leftarrow k + 1$;
end
if $F \neq \emptyset$ **then**
  Solution not found;
else
  return assignment $x \in \mathcal{X}$;
end

Algorithm 2: Pseudo-code for `GreedyRandomized`: Greedy randomized construction procedure.

1-move and *2-move*. A solution in the 1-move neighborhood of p is obtained by changing one facility-to-location assignment in p. Likewise, a solution in the 2-move neighborhood of p is obtained by simultaneously changing two facility-to-location assignments in p.

One way to carry out a local search in these neighborhoods is to evaluate moves in the 1-move neighborhood and move to the first improving solution. If no 1-move improving solution exists, 2-move neighborhood solutions are evaluated and a move is made to the first improving solution. Another way to carry out the local search is to evaluate all 1-move and 2-move neighborhood solutions and move to the best improving solution. In both variants, the search is repeated until no improving solution in the neighborhoods exists. We propose a tradeoff approach here. Instead

of evaluating all of the 1-move and 2-move neighborhood solutions, we sample these neighborhoods and populate a candidate list with improving solutions. One of the solutions from the candidate list is randomly selected and a move is made to that solution. The search is repeated until no improving solution is sampled. Because solutions are sampled, not all neighbors may be evaluated. Consequently, the best solution found may not be a local minimum. Mateus et al. (2011) call this solution an *approximate local minimum*.

Pseudo-code for the approximate local search is shown in Algorithm 3. The procedure takes as input the starting solution π and two parameters, *MaxCLS* and *MaxItr*, which control the sampling. The repeat until loop in lines 1 to 13 is repeated until an approximate local minimum is produced. In line 2, the sampling counter *count* and the candidate list *CLS* are initialized. At each iteration of the inner loop in lines 3 to 9, the 1-move and 2-move neighborhoods of π are sampled without replacement by procedure `Move`(π) in line 4. If this neighbor is an improving feasible solution, it is inserted into *CLS* in line 6. This is repeated until either the candidate list is full or a maximum number of neighbors have been sampled. In lines 10 to 12, if the candidate list is not empty, an assignment $\pi \in CLS$ is randomly chosen. If the set *CLS* is empty after the sampling process, the procedure terminates returning π as an approximate local minimum in line 14. Otherwise, the procedure moves to a solution in *CLS*, repeating the outer loop.

```
   Data  : π, MaxCLS,MaxItr
   Result: Approximate local minimum π
 1 repeat
 2 |  count ← 0; CLS ← ∅;
 3 |  repeat
 4 |  |  π′ ← Move(π);
 5 |  |  if π′ is feasible and cost(π′) < cost(π) then
 6 |  |  |  CLS ← CLS ∪ {π′};
 7 |  |  end
 8 |  |  count ← count + 1;
 9 |  until |CLS| ≥ MaxCLS or count ≥ MaxItr;
10 |  if CLS ≠ ∅ then
11 |  |  Randomly select a solution π ∈ CLS;
12 |  end
13 until CLS = ∅;
14 return π;
```

Algorithm 3: Pseudo-code for `ApproxLocalSearch`: Approximate local search procedure.

2.3. **Path-relinking.** Motived by the fact that a single move from a solution x in the direction of a target solution x_t does not guarantee the feasibility of the new constructed solution, a new variant of path-relinking is proposed in Mateus et al. (2011).

Suppose that among the differences between x and x_t is the location assigned to facility f. In other words, while the location assigned to f in x_t is l, the location assigned to f in x is l', with $l \neq l'$. In this case, is not necessarily feasible to perform a move in x that assigns f to l. If the capacity Q_l is not violated, then the new solution is feasible. Otherwise, a repair procedure must be applied to try to make it feasible.

In this repair procedure, a facility set F is created with all not yet fixed facilities assigned to location l for which capacity is violated. Next, the set $T \subseteq F$ is constructed with all facilities in F having demands less than or equal to the maximum available capacity of locations in M. After a facility from T is randomly selected, set R consists of locations in M that can accommodate it. A location is selected from set R and the facility is assigned to it. This process is repeated until the capacity of location l has a nonnegative slack.

The path-relinking process is a sequence of steps from x_s to x_t. In each step a move is performed from the current solution x with or without repair. Next, a facility i is randomly selected from a set composed of all not yet fixed facilities corrected in the step. A facility is *corrected* when its location becomes the same as the one assigned to it in the target solution x_t. After facility i is fixed, the next step begins. This process continues until the target solution x_t is reached or when no feasible solution is obtained from x.

Algorithm 4 shows pseudo-code for the path-relinking procedure. The algorithm takes as input π_s and π_t, the starting and target solutions, respectively, and outputs the best solution π^* in path from π_s to π_t. Initially, the best solution in the path is set as π^* in line 1 and its corresponding objective function is assigned to f^* in line 2. In line 3, the current solution π' is initialized with π_s, and the working sets *Fix* and *nonFix* are respectively initialized empty and with N. The while loop in lines 5 to 35 is repeated until all facilities in π' are assigned to the same locations assigned to them in π_t, i.e. the set $\varphi(\pi', \pi_t) = \{i \in N \mid \pi'(i) \neq \pi_t(i)\}$ is empty, where $\pi'(i)$ and $\pi_t(i)$ are the locations assigned to facility i in solutions π' and π_t, respectively.

After the set $\mathcal{B}$ of best solutions is set to empty in line 6, each while loop iteration consists of two stages. The first stage in lines 7 to 20 implements the path-relinking step. It creates set $\mathcal{B}$ with the best feasible solutions constructed from the current solution π'. In line 6, $\mathcal{B}$ is initialized as empty. Each facility $v \in \varphi(\pi', \pi_t)$ is analyzed in lines 7 to 20 to create the set $\mathcal{B}$ with the best feasible solutions constructed from the current solution π'. Procedure `makeFeasible` is applied in line 8 to facility v to attempt to create a new solution $\bar{\pi}$ from π'. The application of `makeFeasible` to facility v can either result in a feasible or infeasible solution. In case `makeFeasible` returns a feasible solution $\bar{\pi} \notin \mathcal{B}$, $\bar{\pi}$ is added to $\mathcal{B}$ if $\mathcal{B}$ is not yet full. Otherwise, if $\mathcal{B}$ is full and solution $\bar{\pi} \notin \mathcal{B}$ is not worse than any elite solution, then $\bar{\pi}$ is added to $\mathcal{B}$ replacing some other elite solution.

In the second stage (lines 21 to 34), the procedure first randomly selects a solution π from set $\mathcal{B}$ in line 22. Then, in line 25, it selects at random a facility $i \in I = \varphi(\pi', \pi_t) \setminus (\varphi(\pi', \pi_t) \cap \varphi(\pi, \pi_t))$, where I is defined in line 24 as the set containing all unfixed facilities whose locations were corrected in the previous path-relinking step. A facility is corrected when its location becomes the one assigned to it in the target solution. After fixing facility $i \in I$, sets *Fix* and *nonFix* are updated in

Data : Starting solution π_s, target solution π_t, and candidate size factor η
Result: Best solution π^* in path from π_s to π_t
1 $\pi^* \leftarrow \text{argmin}\{f(\pi_s), f(\pi_t)\}$;
2 $f^* \leftarrow f(\pi^*)$;
3 $\pi' \leftarrow \pi_s$; $Fix \leftarrow \emptyset$; $nonFix \leftarrow N$;
4 Compute difference $\varphi(\pi', \pi_t)$ between solution π' and π_t;
5 **while** $\varphi(\pi', \pi_t) \neq \emptyset$ **do**
6 $\mathcal{B} \leftarrow \emptyset$;
7 **for** $\forall v \in \varphi(\pi', \pi_t)$ **do**
8 Move the facility v in π' to the same location l assigned to v in π_t;
9 $\bar{\pi} \leftarrow$ `makeFeasible`(π', v);
10 **if** $\bar{\pi}$ *is feasible* **then**
11 **if** $|\mathcal{B}| \geq \eta \cdot |\varphi(\pi', \pi_t)|$ **then**
12 **if** $c(\bar{\pi}) \leq \max\{c(\pi) \mid \pi \in \mathcal{B}\}$ **and** $\bar{\pi} \notin \mathcal{B}$ **then**
13 replace the element most similar to $\bar{\pi}$ among all elements with cost worst than $\bar{\pi}$;
14 **end**
15
16 **else if** $\bar{\pi} \notin \mathcal{B}$ **then**
17 $\mathcal{B} \leftarrow \mathcal{B} \cup \{\bar{\pi}\}$;
18 **end**
19 **end**
20 **end**
21 **if** $\mathcal{B} \neq \emptyset$ **then**
22 Randomly select a solution $\pi \in \mathcal{B}$;
23 Compute difference $\varphi(\pi, \pi_t)$ between solution π and π_t;
24 Set $I = \varphi(\pi', \pi_t) \setminus (\varphi(\pi', \pi_t) \cap \varphi(\pi, \pi_t))$;
25 Randomly select a facility $i \in I$;
26 $Fix \leftarrow Fix \cup \{i\}$; $nonFix \leftarrow nonFix \setminus \{i\}$;
27 $\pi' \leftarrow \pi$;
28 **if** $f(\pi') < f^*$ **then**
29 $f^* \leftarrow f(\pi')$;
30 $\pi^* \leftarrow \pi'$;
31 **end**
32 **else**
33 **return** assignment π^*;
34 **end**
35 **end**
36 **return** assignment π^*;

Algorithm 4: Pseudo-code for `PathRelinking`: Path-relinking procedure.

line 26. Finally, the next path-relinking step solution π' is set as π in line 27 and, if $f(\pi') < f^*$, then the best cost f^* and best solution π^* are updated in lines 29 and 30, respectively. However, if in some path-relinking step no feasible solution

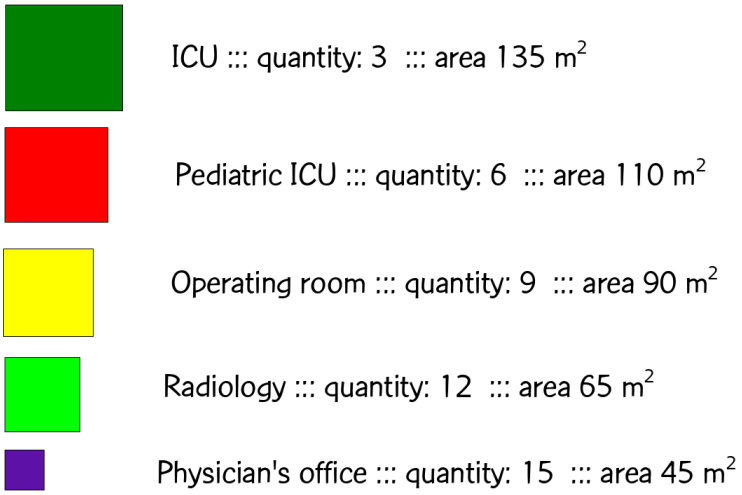


FIGURE 1. Medical facilities: types, quantities, and dimensions

TABLE 1. Unsymmetrical people flow for each pair of facilities

	ICU	PICU	OR	R	PO
Intensive care unit (ICU)	0	70	500	90	100
Pediatric ICU (PICU)	90	0	300	78	95
Operating room (OR)	700	250	0	200	50
Radiology (R)	80	60	300	0	380
Physician's office (PO)	200	150	600	800	0

is obtained from π', the while loop is interrupted, returning the current solution π^* as the result in line 33. If the target solution is reached, then π^* is returned in line 36.

This path-relinking is different from the standard variant because given solutions x_s and x_t, their commons elements are not kept fixed a priori, such that a small portion of the solution space spanned by the remaining elements is explored. The new variant fixes one facility at time at each step.

3. An illustrative example of hospital layout

In this section, we provide an example to illustrate a hospital layout problem and show a corresponding solution produced with the GRASP with path-relinking heuristic described in Section 2.

3.1. **The layout problem instance.** Our example considers the medical departments listed in Figure 1: three intensive-care units (ICUs), six pediatric intensive-care units (PICUs), nine operating (OR) rooms, twelve radiology (R) rooms, and fifteen physician offices (PO). Table 1 shows the people flow for each pair of facilities.

Figure 2 presents a hypothetical plant for a hospital with three floors and an elevator. The first floor plant has four locations, numbered 1 to 4, with dimensions

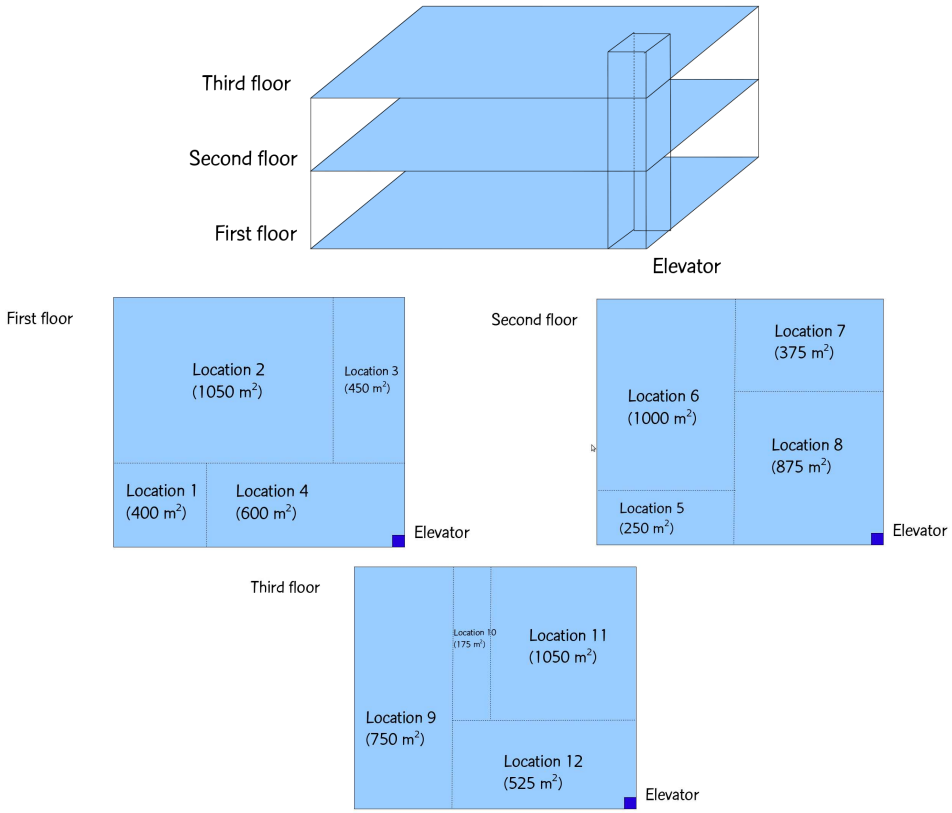


FIGURE 2. Hospital plant with three floors and twelve locations

400 m^2, 1050 m^2, 450 m^2, and 600 m^2, respectively. The second floor has four locations, numbered 5 to 8, with 250 m^2, 1000 m^2, 375 m^2, and 875 m^2, respectively. The third floor has four locations, numbered 9 to 12, with 400 m^2, 1050 m^2, 450 m^2, and 600 m^2, respectively.

Table 2 shows the Euclidean distances between each of the 12 hospital locations. Distances between pairs of locations on same floor are assumed to be the Euclidean distances between the centers of the locations (Figure 3a). Distances between locations on different floors are assumed to be the sums of the Euclidean distance between the center of the first location to the elevator on that floor, the distance traveled by elevator, and the Euclidean distance between the elevator on the other floor and the center of the second location (Figure 3b).

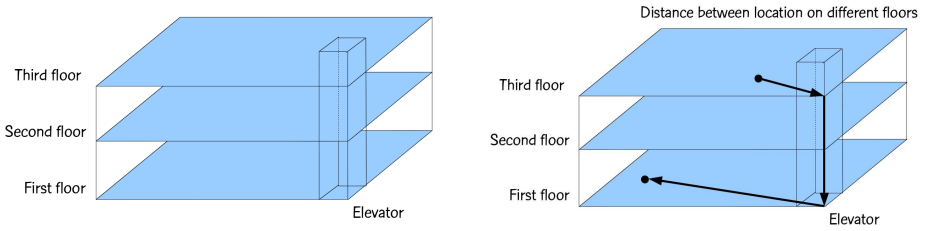
Finally, Table 3 shows the cost of assigning each medical facility to each of the twelve locations illustrated in Figure 2.

3.2. **The solution found by GRASP with path-relinking.** Figure 4 shows the assignment found with GRASP with path-relinking heuristic. We make the following observations regarding the solution.

- Since all the facilities could be accommodated in two floors, only two floors were used.

TABLE 2. Symmetrical distances between hospital locations.

	1	2	3	4	5	6	7	8	9	10	11	12
1	0	26.1	41	25	99.06	109.23	105.53	82.73	130.53	127.19	117.02	100.23
2	26.1	0	25	30.25	105.99	115.76	112.06	89.26	137.06	133.72	123.55	106.76
3	41	25	0	26.1	93.62	103.79	100.09	77.29	125.09	121.75	111.58	94.79
4	25	30.25	26.1	0	75.83	86	82.3	59.5	107.3	103.96	93.79	77
5	99.06	105.99	93.62	75.83	0	25	45.07	27.95	107.13	103.79	93.62	76.83
6	109.23	115.76	103.79	86	25	0	27.95	27.95	117.3	115.26	103.79	87
7	105.53	112.06	100.09	82.3	45.07	27.95	0 25	113.6	110.26	100.09	83.3	
8	82.73	89.26	77.29	59.5	27.95	27.95	25 0	90.8	87.46	77.29	60.5	
9	130.53	137.06	125.09	107.3	107.13	117.3	113.6	90.8	0	12.5	28.5	30.51
10	127.19	133.72	121.75	103.96	103.79	115.26	110.26	87.46	12.5	0	17.5	29.1
11	117.02	123.55	111.58	93.79	93.62	103.79	100.09	77.29	28.5	17.5	0	25.12
12	100.23	106.76	94.79	77	76.83	87	83.3	60.5	30.51	29.1	25.12	0



(a) (b)

FIGURE 3. Distance between locations

TABLE 3. Facility-location assignment cost

	1	2	3	4	5	6	7	8	9	10	11	12
ICU	1200	1050	1200	1300	1200	1050	1200	1300	1200	1050	1200	1300
PICU	1000	850	1000	1100	1000	850	1000	1100	1000	850	1000	1100
OR	800	650	800	900	800	650	800	900	800	650	800	900
R	600	450	600	700	600	450	600	700	600	450	600	700
PO	400	250	400	500	400	250	400	500	400	250	400	500

- The heuristic assigned facilities to floors 2 and 3. However, because of the symmetry of floors 1 and 3, a solution with equal cost would allocate all third floor facilities to the first floor.
- On the second floor, the facilities are concentrated around the elevator since patients often move between second floor ICUs and PICUs and third floor radiology facilities.
- The ICUs were located near the ORs since their inter-facility flows were high.
- Note that a single PICU was located on the third floor while a single ICU was located on the second. All radiology facilities are located on the third floor. Even though the flow between the ICU and radiology facilities is greater than the flow between PICU and radiology facilities, the lone ICU on the second floor had to be placed there since there was insufficient space for it in any location on the third floor. The lone PICU on the third floor was located there and not on the second floor since it is near the radiology facilities.

4. Concluding remarks

In this paper, we revisted the GRASP with path-relinking heuristic for the generalized quadratic assignment problem (GQAP) of Mateus et al. (2011) and applied the heuristic to solve a facility layout problem (FLP) as a generalized quadratic assignment problem (GQAP). We illustrate the solution method with a hypothetical hospital layout problem. To implement this approach, we require inter-facility flow rates. These will need to be established from data gathered in other hospital settings and adapted to the setting under consideration in the design. The approach can also be applied to other settings, such as sports facilities, shopping malls, and airports.

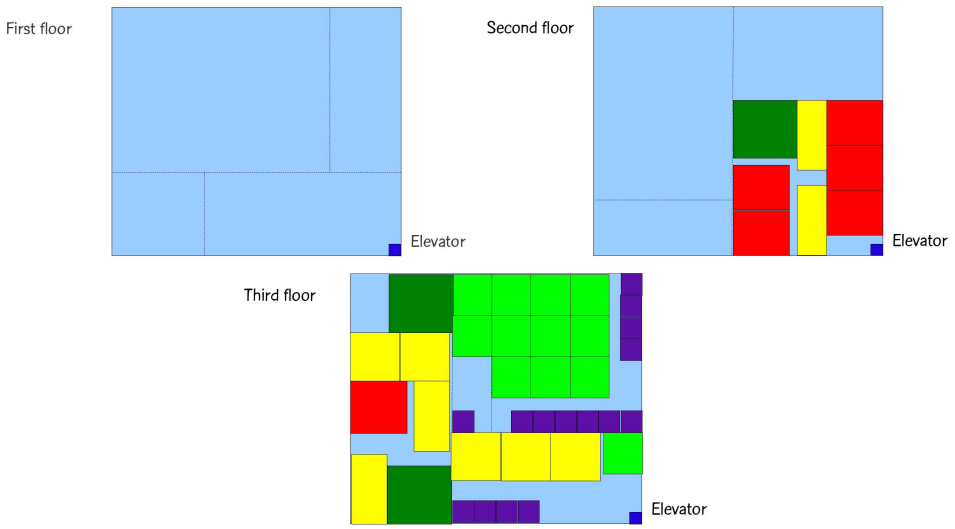


FIGURE 4. Assignment found by GRASP-PR in 1898.4 seconds on a 2.6 GHz machine.

REFERENCES

M. S. Bazaraa. Computerized layout design: A branch and bound approach. *AIIE Transactions*, 7(4):432–438, 1975.

T. A. Feo and M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *J. of Global Optimization*, 6:109–133, 1995.

L. R. Foulds and D. F. Robinson. A strategy for solving the plant layout problem. *Operational Research Quarterly*, 27(4):845–855, 1976.

F. Glover. Tabu search and adaptive memory programing – Advances, applications and challenges. In R. S. Barr, R. V. Helgason, and J. L. Kennington, editors, *Interfaces in Computer Science and Operations Research*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, pages 1–75. Kluwer, 1996.

L. Kaufman and F. Broeckx. An algorithm for the quadratic assignment problem using bender's decomposition. *European J. of Operational Research*, 2(3):207–211, 1978.

T. C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica: J. of the Econometric Society*, 25(1):53–76, 1957.

A. Kusiak and S. S. Heragu. The facility layout problem. *European J. of Operational Research*, 29(3):229–251, 1987.

M. Laguna and R. Martí. GRASP and path relinking for 2-layer straight line crossing minimization. *INFORMS J. on Computing*, 11:44–52, 1999.

E. L. Lawler. The quadratic assignment problem. *Management Science*, 9(4):586–599, 1963.

G. R. Mateus, M. G. C. Resende, and R. M. A. Silva. GRASP with path-relinking for the generalized quadratic assignment problem. *J. of Heuristics*, 17:527–565, 2011.

M. G. C. Resende and C. C. Ribeiro. GRASP with path-relinking: Recent advances and applications. In T. Ibaraki, K. Nonobe, and M. Yagiura, editors, *Metaheuristics: Progress as Real Problem Solvers*, pages 29–63. Springer, 2005.

M. G. C. Resende and C. C. Ribeiro. Greedy randomized adaptive search procedures: Advances and applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, pages 293–319. Springer, 2nd edition, 2010.

C. C. Ribeiro and M. G. C. Resende. Path-relinking intensification methods for stochastic local search algorithms. *J. of Heuristics*, 18:193–214, 2012.

M. J. Rosenblatt. The facilities layout problem: a multigoal approach. *J. Prod. Res.*, 17, 1979.

(R.M.A. Silva) Center for Informatics, Federal University of Pernambuco, Av. Jornalista Anibal Fernandes, s/n - Cidade Universitária, CEP 50.740-560, Recife, PE, Brazil.
E-mail address: rmas@cin.ufpe.br

(M.G.C. Resende) Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.
E-mail address: mgcr@research.att.com

(P. M. Pardalos) Department of Industrial and Systems Engineering, University of Florida, 303 Weil Hall, Gainesville, FL, 32611, USA.
E-mail address: pardalos@ufl.edu

(G.R. Mateus) Dept. of Computer Science, Federal University of Minas Gerais, CEP 31270-010, Belo Horizonte, MG, Brazil.
E-mail address: mateus@dcc.ufmg.br

(G.D. Tomi) Departamento de Engenharia de Minas e de Petróleo, Escola Politécnica da Universidade de São Paulo, Av. Prof. Mello Moraes, 2373, Butantã, CEP 05508-030, São Paulo, SP, Brazil.
E-mail address: gdetomi@usp.br