The dynamic space allocation problem: applying hybrid GRASP and tabu search metaheuristics

Geiza Cristina da Silva [a]
Laura Bahiense [b]
Luiz Satoru Ochi [c,*]
Paulo Oswaldo Boaventura-Netto [b]

[a] Department Department of Statistic, Federal University of Pernambuco, Recife, Pernambuco, Brazil, CEP 50.740-540.
E-mail: geiza.silva@gmail.com
[b] Production Engineering Program, COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil, CEP 21.941-972.
E-mail: laura@pep.ufrj.br, boaventu@pep.ufrj.br
[c,*] Computer Science Institute, Fluminense Federal University, Niterói, Rio de Janeiro, Brazil, CEP 24.210-240.
E-mail: satoru@ic.uff.br
[*] Corresponding author. Tel.: +55(21)26295681; Fax: +55(21)26295669.

Abstract

This work is devoted to the Dynamic Space Allocation Problem, where project duration is divided into a number of consecutive periods, each of them associated with a number of activities. The resources required by the activities have to be available in the corresponding workspaces and those sitting idle during a period have to be stored. This problem contains the Quadratic Assignment Problem (QAP) as a particular case, which puts it in the NP-hard class. In this context, the difficulty of identifying optimal solutions, even for instances of medium size, justifies the use of heuristic techniques. This work proposes a construction and a hybrid algorithm (HGT) based on the GRASP and Tabu search metaheuristics. Comparisons are presented for values obtained by HGT, pure GRASP versions, Tabu search and literature results. Computational results show the proposed methods to be competitive in relation to instances in the literature and to existing techniques.

Keywords: Dynamic Space Allocation Problem, Quadratic Assignment Problem, Metaheuristics, Combinatorial Optimization.

1 Introduction

The Dynamic Space Allocation Problem (DSAP) [8], quite recent in the literature, was inspired by the need to optimize the cost of reallocating resources when we assign activities to workspaces and resources to work or storage spaces during a multi-period planning horizon. The reallocation costs include both the distances between locations and preparation costs for moving equipment and other resources.

A given *project* is divided into consecutive time *periods*, each involving certain *activities* that have to be executed. A given resource is considered *necessary* during a given period when it is related to an activity of this period and considered *idle* otherwise. The place where the project is worked on is divided into *workspaces*, where activities are carried out and their associated resources are stored, and *depots*, where idle resources are stored. The set of periods, with their activities and the respective necessary and idle resources, is the project's *agenda*. The objective of the problem is to allocate the resources to spaces in such a way as to minimize the reallocation costs over the time periods associated with the project.

The DSAP, introduced by McKendall and Jaramillo in [8], is related to other well-known combinatorial optimization problems: the problem of assigning activities to workspaces (Quadratic Assignment Problem, QAP) [4,6], and the problem of allocating activities to multiple

time periods (Dynamic Facility Layout Problem, DFLP) [10] The Static Facility Layout Problem (SFLP) is a well-researched problem of finding positions of departments on the plant floor in order to avoid departments overlapping and to minimizing material handling cost (i.e., minimizing the sum of the product of the flow of materials, distance, and transportation cost per unit per distance unit for each pair of departments). When material flows between departments change during the planning horizon, the problem becomes the Dynamic Facility Layout Problem (DFLP). The solution of the DFLP is a group of layouts (one for each period), which minimizes the sum of the material handling cost and the rearrangement cost during the planning horizon.

Layout problems are known to be complex and are generally NP-Hard [2]. There are several research reviews describing facility layout problems carefully ([1], [3] and [6]). Loiola et al. [6] dealt with facility layout as a branch of the Quadratic Assignment Problem (QAP), an NP-hard combinatorial optimization problem first introduced by Koopmans and Beckmann [4] to model a facility location problem. In this context, the objective is to find a minimum cost assignment of facilities to locations considering the flow of materials between facilities and the distance between locations.

In general terms, the Dynamic Space Allocation Problem (DSAP), addressed by this paper, can be described as follows: it is the assigning of activities to workspaces and idle resources to storage spaces while minimizing the total distance traveled by the resources throughout the duration of the project [8]. The inputs of the DSAP are: the schedule of project activities and the required resources to perform the activities, the layout of the floor plan, the capacity of the storage spaces, and the distances between the locations. The output is a sequence of layouts containing the locations of the activities and their resources in the workspaces as well as the idle resources in the storage spaces.

The DFLP consider the minimization of the distances traveled by the materials/products during the planning horizon. The DFLP is related to the DSAP, since the departments require spaces (workspaces) to produce a product, and the materials (which can be considered as resources) travel from department to department. In the DFLP, the facility layout is redesigned every time a change in the flow pattern (i.e., for each period) occurs (e.g., a new product mix is required). Each period considers the relocation of the departments (e.g., old departments, new departments, or modified departments). That is, for each period, there exists a new set of activities requiring resources. Each of the new activities (departments) is allocated to a workspace such that total cost (i.e., the sum of the material handling costs and the relocation costs) is minimized. Therefore, the DSAP generalizes the DFLP and since DFLP is NP-hard, so is the DSAP.

Figure 1 shows a small DSAP instance, where the agenda has four periods, five activities and nine resources. The space layout has three workspaces (W_1 to W_3), along with three depots (D_1 to D_3). This type of layout, with the rows of workspaces and depots alongside each other, is considered in every instance of the literature. The Manhattan distance metric is used in determining distances. Each space can receive up to three resources. Lastly, in all instances we found in the literature, the resource requirements of an activity spanning multiple periods do not change from one period to another, although in some practical applications an activity can require a different set of resources for each period. Figure 1 shows this situation.

Period	**Activities (Necessary resources)**	**Idle resources**
1	A_1 (6,7) A_2 (1,5)	2,3,4,8,9
2	A_2 (1) A_3 (3,4)	2,5,6,7,8,9
3	A_4 (2,8)	1,3,4,5,6,7,9
4	A_4 (2) A_5 (6,9)	1,3,4,5,7,8

W_1	W_2	W_3
D_1	D_2	D_3

Figure 1: A small DSAP instance: agenda and space layout

For a given solution to be considered feasible, the following conditions must be met:

1. During a given period, exactly one activity can be carried out in a workspace.
2. At any time, a given activity is always carried out in the same workspace.
3. The capacity of a workspace must be sufficient to contain the resources required by its activity.
4. The depot capacities also have to be respected.

Figure 2 shows an optimal solution of cost 13 for this instance, found using Cplex.

During period 1, resources 1 and 5, used by activity A_2, were allocated to workspace W_1 and resources 6 and 7, used by activity A_1, were allocated to workspace W_3, while the idle resources 2 and 8, 3 and 4 and 9 were allocated to depots D_1, D_2 and D_3, respectively, as shown in the figure. The first period is free of traveling cost by convention.

During period 2, the resources for A_1 become idle and are allocated to D_3, covering one distance unit. At the same time, A_3 is allocated to W_2, which means that its resources 3 and 4 have to come from D_2 (one distance unit). Besides, resource 5 of activity A_2 is made idle in this period, and is allocated to D_1 (one distance unit). The subtotal distance thus equals (2 resources × 1 unit distance + 2 resources × 1 unit distance + 1 resource × 1 unit distance) = 5.

The third period finds only A_4 being carried out at W_1, where A_2 is deactivated, with its resources going to W_3 (one distance unit). Consequently, resources 2 and 8 travel from D_1 to W_3, and resource 1 moves in the opposite direction. The subtotal distance thus equals (2 resources × 1 unit distance + 2 resources × 1 unit distance + 1 resource × 1 unit distance) = 5.

Lastly, period 4 has activity A_4 continuing to be executed (in W_1) requiring only resource 2, making idle resource 8 (one distance unit), while activity A_5 is carried out in W_3, receiving its resources from D_3 (one distance unit). The subtotal distance thus equals (1 resource × 1 unit distance + 2 resources × 1 unit distance) = 3.

The sum of the distances traveled by all resources is 13, which is equal to the optimal problem solution value.

A_2(1,5)		A_1(6,7)
2,8	3,4	9

Period 1

A_2(1)	A_3(3,4)	
2,8,5		6,7,9

Period 2

A_4(2,8)		
1,5	3,4	6,7,9

Period 3

A_4(2)		A_5(6,9)
1,5,8	3,4	7

Period 4

Figure 2: An optimal solution for the sample instance

In this paper, we present a construction algorithm and a hybrid heuristic based on GRASP (greedy randomized adaptive search procedure) and Tabu search metaheuristics, in order to obtain near-optimal solutions for the DSAP. Section 2 contains a brief review of the literature and a mathematical formulation is presented in Section 3. Section 4 presents the proposed heuristics and Section 5 shows the computational results obtained using a set of test problems taken from the literature. Section 6 provides conclusions and suggestions for future research.

2 Brief review of the literature

The DSAP is a recently formulated problem for which there are few contributions in the literature. It was proposed by McKendall et al. [7], which associated it with the minimization of costs for transporting given resources such as equipment, tools and replacement parts, in situations encountered during maintenance operations at nuclear power plants. These costs have been frequently associated, in theoretical instances, with the distances traversed by the resources. The authors presented a mathematical formulation for exact resolution and two heuristic algorithms based on simulated annealing.

McKendall Jr. and Jaramillo [8] presented five construction algorithms and a Tabu search strategy (Tabu I). A new constraint was added to the formulation to keep the idle resources that remain idle for consecutive time periods in the same location, a reasonable assumption that reduces the search space. Tabu I keeps two tabu lists, one for displacements of activities and the other for movements of resources. At each iteration, the best movement is chosen and the resulting solution becomes the current one for the next iteration. Comparison with previously used methods showed that combining the construction algorithms with Tabu I give better results, both in solution quality and computing time.

In [9], a modified mathematical formulation and three Tabu-type algorithms are proposed. The mathematical formulation has objective function and constraints modified in order to consider the costs of transporting and loading/unloading resources in different locations during the time horizon. These modifications make the new instances very different from those ones proposed with the first formulation. Moreover, they did not present in [9] new instances that are capable of taking advantage of the model generalization.

The first Tabu algorithm, Tabu II, is a simple tabu search, which differs from Tabu I in terms of the individual manipulation of idle resources.

The second heuristic algorithm, Tabu III, recalculates the sizes of the tabu lists at each iteration and uses intensification and diversification strategies. The diversification is guided by evaluation functions for inefficient movements of activities and idle resources. A modified objective function based on these functions is used to evaluate these movements. The intensification strategy consists in not allowing movements yielding reduction percentiles over a threshold value.

The third heuristic algorithm, Tabu IV, is similar to Tabu II but evaluates and orders every possible movement, with the best ones M being selected to create a *candidate movement list (CML)*. The first *CML* member is then accepted with probability p. If it is accepted, the resulting solution becomes current for the next iteration. Otherwise, the same strategy is applied to the next movement, and so on, until a movement is selected. If the strategy fails, the best *CML* movement is chosen. Computational results show the second heuristic algorithm as the best among those compared in the paper.

3 Mathematical formulation

The formal DSAP definition considers the following parameters and indexes:

Indexes:

J = set of activities (j = 1, 2, ..., $|J|$);

R = set of resources (r = 1, 2, ..., $|R|$);

P = set of periods (p = 1, 2, ..., $|P|$);

N = total location number (workspaces and depots);

L = location set, $|L| = N$ (k, l = 1, 2, ..., N);

W = workspace set, $W \subset L$ ($w \in W$);

S = depot set, $S \subset L$ *(*$s \in S$*)* and $W \cup S = L$*;*

R_{jp} = set of resources required by activity j in period p;

I_p = set of idle resources during period p;

A_p = set of activities in period p.

Parameters:

d_{kl} = distance between locations k and l*;*

C_s = capacity of depot s.

Our decision variables are x_{rkp} which equals 1 if the resource r is assigned to the location k in period p, and 0 otherwise, and y_{jw} which equals 1 if the activity j is assigned to the workspace w, and 0 otherwise. The mathematical formulation proposed in [7], modified in order to consider the generalized situation where the resource requirements of an activity spanning multiple periods can change from one period to another, is presented below:

Minimize:

(1) $$\sum_{r=1}^{|R|}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{p=1}^{|P|-1} d_{kl}\, x_{rkp}\, x_{rlp+1}$$

subject to:

(2) $$\sum_{s\in S} x_{rsp} = 1, \forall r \in I_p, \forall p \in P,$$

(3) $$\sum_{r\in I_p} x_{rsp} \le C_s, \forall s \in S, \forall p \in P,$$

(4) $$\sum_{w\in W} y_{jw} = 1, \ \forall j \in J,$$

(5) $$\sum_{j\in A_p} y_{jw} \le 1, \forall w \in W, \forall p \in P,$$

(6) $$\sum_{r\in R_{jp}} x_{rwp} = \left|R_{jp}\right| y_{jw}, \forall j \in A_p, \forall w \in W, \forall p \in P,$$

(7) $$x_{rkp} = 0 \text{ or } 1, \forall r \in R, \forall k \in L, \forall p \in P,$$

(8) $$y_{jw} = 0 \text{ or } 1, \forall j \in J, \forall w \in W.$$

The objective function (1) minimizes the distance traveled by the resources over the project time periods. Constraints (2) guarantee that every idle resource in each period will be assigned to a single depot and constraints (3) make sure that depot capacity is respected throughout each period. Constraints (4) and (5) specify, respectively, that every activity be assigned to a single workspace and that every workspace have at most one assigned activity. Constraints (6) guarantee that every resource required by a given activity in each period is assigned to the same workspace where this activity is going on. Finally, constraints (7) and (8) state that the decision variables are binary.

The addition of index p to the required resources set of activity j, in order to define R_{jp} instead of R_j, made the model capable of dealing with the situation where an activity j requires different sets of resources in consecutive periods. Trough this little change the model became more general and the solution methodologies were not affected by this adjustment.

This formulation is clearly quadratic, but it can be linearized by substituting expression (1a) for (1), where $z_{rklp,p+1}$ is a binary decision variable. Additionally, constraints (9-11) have to be included [7]:

(1a) $$\sum_{r=1}^{|R|}\sum_{k=1}^{N}\sum_{l=1}^{N}\sum_{p=1}^{|P|-1} d_{kl}\, z_{rklp,p+1}$$

(9) $$x_{rkp} + x_{rlp+1} - 1 \le z_{rklp,p+1}, \forall r \in R, \forall k,l \in L, l \ne k, \forall p \in P,$$

(10) $$x_{rkp} + x_{rlp+1} \geq 2z_{rklp,p+1}, \forall r \in R, \forall k,l \in L, l \neq k, \forall p \in P,$$

(11) $$z_{rklp,p+1} = 0 \text{ or } 1, \forall r \in R, \forall k,l \in L, l \neq k, \forall p \in P.$$

4 Proposed heuristic algorithms

This work presents a construction algorithm and a hybrid heuristic based on GRASP and Tabu search metaheuristics, together with the isolated implementation of these two methods. These algorithms have achieved good results for a number of combinatorial optimization problems [8, 9] and the combination of their ideas in hybrid heuristics have been fruitful for difficult problems such as QAP.

4.1 The construction algorithm

The construction algorithm proposed here has two phases. The first one aims to allocate activities to workspaces and, in the second one, idle resources are assigned to depots. In the first phase, we consider the set *J* of activities and the set *W* of workspaces. At each iteration, an *initial candidate list (ICL)* is drawn up of the best results to be considered for inclusion in the solution.

In the first phase, each feasible solution is iteratively constructed as follows. First, an activity $a \in J$ is randomly associated with a workspace. At each iteration, a *restricted list of candidates (RLC)* is obtained from the remaining *ICL*, and an activity-workspace pair is randomly chosen from *RLC* and included in the solution. *ICL* contains the activity not yet assigned, paired with the available workspaces. For each pair, the insertion cost in the partial solution is calculated. *ICL* is then sorted in non-decreasing cost order and the first Ψ *ICL* elements are selected to compose *RLC*. Ψ value is given by $\alpha \times min(n\_av, |W|)$, where *n_av* is the number of available workspaces and $\alpha \in [0,1]$ is the same GRASP parameter.

The second phase is the Randomized Storage Policy (RSP) proposed in [8] and adapted from RSP by considering the resource assignment made in the last period, denoted by RSPA.

RSP associates idle resources to depots as follows: after obtaining a complete activity allocation, each idle resource is allocated to the depot nearest to the activity requiring it. The first idle resources to be allocated are those first called up for use. In addition, a resource that will remain idle during the remaining periods will be allocated to the depot nearest to the workspace where the activity which last used it was allocated.

In the first period, RSPA initializes the allocation of idle resources to depots as in RSP but, in the sequence, the allocation from the previous period is considered when making a new allocation. Then, if a given resource remains idle, its previous allocation is maintained for the new period. The remaining idle resources are allocated as in RSP.

4.2 Neighborhood structures

Once a feasible solution *s* is available, two natural neighborhood structures proposed in [7] are used:

- *NA*(*s*) is related to the assignment of activities in *s*, and
- *NR*(*s*) is related to the assignment of idle resources in *s*.

When considering the neighborhood *NA*(*s*), an *interchange move* changes the workspaces of two activities in one or more periods, and a *relocation move* removes an activity from a workspace and reassigns it to an available (empty) location.

Figure 3 illustrates, and highlights in bold, an interchange move and a relocation move in the neighborhood *NA*(*s*), based on the example presented in Figure 2. The interchange move occurs at period 1, where activities A_1 and A_2 have their workspaces interchanged: A_1 changes from workspace W_3 to workspace W_1 and A_2 changes from W_3 to W_1, always satisfying the feasibility

conditions presented before. The relocation move occurs at period 4, where activity A_5 is removed from workspace W_3 and it is relocated to workspace W_2.

A_1(6,7)		**A_2(1,5)**
2,8	3,4	9

Period 1

	A_3(3,4)	**A_2(1)**
2,8,5		6,7,9

Period 2

A_4(2,8)		
1,5	3,4	6,7,9

Period 3

A_4(2)	**A_5(6,9)**	
1,5,8	3,4	7

Period 4

Figure 3: Interchange and relocation move in *NA* for the sample instance

When considering neighborhood *NR*(*s*), an *interchange move* interchanges the depots of two (or more) idle resources in all consecutive periods where the resources remain idle. On the other hand, a *relocation move* removes one idle resource from a depot, in all consecutive periods where the resource remains idle, and then reassigns it to a different depot, if the depot capacity is not exceeded.

Figure 4 illustrates, and highlights in bold, an interchange move and a relocation move in the neighborhood *NR*(*s*), based on the example presented in Figure 2. The interchange move occurs at consecutive periods 3 and 4, where resources 1 and 3 interchange their respective depots. The relocation move occurs at consecutive periods 1 and 2, where the idle resource 2 is reassigned from depot D_1 to depot D_2.

A_2(1,5)		A_1(6,7)
8	3,4,**2**	9

Period 1

A_2(1)	A_3(3,4)	
8,5	**2**	6,7,9

Period 2

A_4(2,8)		
3,5	**1**,4	6,7,9

Period 3

A_4(2)		A_5(6,9)
3,5,8	**1**,4	7

Period 4

Figure 4: Interchange and relocation move in *NR* for the sample instance

4.3 Proposed hybrid GRASP and tabu heuristic algorithm

In the *Hybrid GRASP* and *Tabu* heuristic algorithm (HGT), the construction phase yields an initial solution using the construction algorithm, presented in 4.1. In the local search phase, all possible activity and idle resource movements are performed as in 4.2, independently of one another, in order to find the best feasible move (either an activity or an idle resource move). This best move is executed and the current solution is updated. The search terminates when the current solution can no longer be improved.

HGT uses Tabu search as an intensification step. Starting with the current solution, the tabu algorithm executes the following steps until a fixed given number of non-improving consecutive iterations is achieved:

- Evaluate all possible movements related to activities and, for each one, execute heuristic algorithm RSPA in order to reallocate the idle resources according to the new activities matrix obtained.
- Choose the best movement.
- Update current solution and Tabu list.

A dynamic FIFO Tabu list stores the recent moves performed in the activity set. A movement is classified as Tabu if it belongs to the Tabu list or if it is a reverse movement of another movement belonging to the Tabu list. The size of the list ranges between a lower bound l_i and an upper bound l_s. The time during which a given movement is considered Tabu is determined by

the list size. At the beginning, and at each η iterations without improvement, the new size of the list is randomly chosen inside the interval $[l_i, l_s]$.

An aspiration criterion allows a Tabu movement to be performed if the cost of the solution yielded by the movement, followed by the application of heuristic RSPA, is better than the best current one.

At the end of this process, the solution is updated according to the better value. The final result is the best solution obtained after running a fixed number of iterations.

The main differences of our tabu search with respect to previous ones are:

- The movements are made only with activities, while [8] and [9] execute the best movement also for idle resources.
- The use of RSPA heuristic algorithm to yield a partial solution for idle resources.

Heuristic *HGT* (ITER_GRASP, α, ITER_TABU, l_i, l_s, η)
1. **for** $k \leftarrow 1$ **until** ITER_GRASP **do**
2. *Sol* ← *constructionAlgorithm* (α);
3. *Sol* ← *localSearch* (*Sol*);
4. *Sol* ← *tabuSearch* (*Sol*, ITER_Tabu, l_i, l_s, η);
5 *updateSolution* (*Sol*, *bestSol*);
6. **endfor**
7. *return bestSol;*
end.

Figure 5: Hybrid GRASP / Tabu Algorithm

5 Computational results

The set of tested instances is available in [7]. It is composed of 96 instances (P01 to P96) containing problems with 6, 12, 20 and 32 locations and 9, 18, 30 and 48 resources, each one with 10, 15 and 20 periods. Half of the locations are workspaces and the other half are depots. Each depot has a maximum capacity of three resources, and the number of required resources per activity varies between 1 and 3. Lastly, the number of activities ranges between 6, for small instances, and 87, for larger instances.

In order to better evaluate the HGT algorithm, we also implemented pure versions of GRASP and Tabu search. To obtain pure GRASP, the tabu call was taken off the HGT algorithm, which left Steps 1 to 3 and 5 to 7 to be executed. For pure tabu, an initial solution was built by the algorithm described in Section 4.1 with $\Psi = 1$ (greedy solution). Then, in Figure 5, only Step 4 is considered.

All parameters involved (Table 1) in the proposed techniques were determined with the aid of preliminary testing. The parameters l_i, l_s and η have been rounded to integers using the floor operator. The algorithms were written in C, using the GCC 4.2.3 compiler with –O3 option. We used a computer with an Intel® Core™2 Quad Processor Q6600 with 2.40 GHz, 4 Gbytes of RAM, and the Linux 2.6.24-19 operating system.

Parameter	**HGT**	**GRASP**	**TS**
GRASP Iterations	20	100	–
α	1.0	1.0	–
Tabu Iterations	50	–	100
l_i	$1.1\times\sqrt{\lvert J\rvert}$	–	$1.1\times\sqrt{\lvert J\rvert}$
l_s	$\lvert W\rvert -1\times\sqrt{\lvert J\rvert}$	–	$\lvert W\rvert -1\times\sqrt{\lvert J\rvert}$
η	0.2 × Tabu Iterations	–	0.2 × Tabu Iterations

Table 1: Parameters in proposed HGT, GRASP and Tabu Search

Optimal solutions were reported in the references for 25 instances, P01−P24 and P27. We solved the formulation described in Section 2 using Cplex 11 and, besides the previous known optimal values, we were able to find optimal solutions to instances P25, P26, P28, P29, P30, P31, P32, P35, P36, P39 and P40. Surprisingly, for instances P25, P26, P28, P32, P35, P36, P39 and P40 the best known upper bounds (heuristic algorithm solutions) reported in [9] were smaller than the optimal solutions found by Cplex 11.

Table 2 reports the results for the instances solved by Cplex 11, comparing the solutions and computational times for Cplex 11 and literature, GRASP, Tabu (TS) and HGT algorithms. The first column presents the instances, the second column presents the solutions found by Cplex 11, followed in the third column by their computational times. The fourth column presents the best solutions found by the literature, followed by their computational times, the sixth column presents the best solutions found by GRASP procedure, followed in the seventh column by their computational times, the eighth column presents the best solutions found by Tabu procedure, followed in the ninth column by their computational times and finally, the tenth column presents the best solutions found using the HGT procedure, followed by their computational times. Best solutions are set off in **bold** typeface. There are some values reported by [9] that are lesser than those found by Cplex. They are indicated by *underlined italics*. The last line in Table 2 shows the average processing time and the average objective function values for each algorithm. Note that the average objective function value in the literature column is lesser than that found by Cplex.

The computer reported in [9], a Pentium IV 2.4 GHz, has an estimated power of 4595 MFlops (http://www.activewin.com/reviews/hardware/processors/intel/p424ghz/benchs.shtml), while our computer has an estimated power of 44300 MFlops (http://techgage.com/print/intel_core_2_quad_q6600). Thus, in order to perform a fair comparison between the computational times, we use the rate of 4595/44300 Mflops to adjust the best literature times (fifth column of Table 2). All the computational times are expressed in seconds.

Inst.	Cplex	CPU(Cplex)	Lit.	CPU(Lit)	GRASP	CPU(GRASP)	TS	CPU(TS)	HGT	CPU(HGT)
P1	**16**	0.3	**16**	0.8	**16**	0.0	**16**	0.0	**16**	0.2
P2	**25**	0.3	**25**	1.0	26	0.0	26	0.0	**25**	0.2
P3	**18**	0.3	**18**	0.8	**18**	0.0	**18**	0.0	**18**	0.2
P4	**25**	3.5	**25**	0.8	26	0.0	26	0.0	**25**	0.3
P5	**16**	1.3	**16**	1.6	**16**	0.0	**16**	0.0	**16**	0.3
P6	**27**	5.5	**27**	2.1	**27**	0.0	**27**	0.0	**27**	0.2
P7	**16**	3.5	**16**	2.3	**16**	0.1	**16**	0.0	**16**	0.2
P8	**31**	0.9	**31**	1.6	**31**	0.0	**31**	0.0	**31**	0.2
P9	**25**	6.8	**25**	1.8	**25**	0.1	**25**	0.0	**25**	0.4
P10	**46**	19.0	**46**	2.1	**46**	0.1	**46**	0.0	**46**	0.4
P11	**32**	7.7	**32**	1.8	**32**	0.1	**32**	0.0	**32**	0.5
P12	**41**	15.0	**41**	2.1	43	0.1	43	0.0	**41**	0.4
P13	**28**	11.0	**28**	2.6	**28**	0.1	**28**	0.0	**28**	0.5
P14	**45**	18.9	**45**	1.8	**45**	0.1	**45**	0.0	**45**	0.4
P15	**35**	17.8	**35**	2.1	**35**	0.1	**35**	0.0	**35**	0.5
P16	**49**	4.5	**49**	1.8	**49**	0.1	**49**	0.0	**49**	0.5
P17	**35**	16.2	**35**	7.5	**35**	0.2	**35**	0.1	**35**	0.9
P18	**60**	62.3	**60**	4.9	62	0.2	62	0.1	62	0.5
P19	**46**	26.6	**46**	5.2	**46**	0.2	**46**	0.1	**46**	0.6
P20	**60**	57.8	**60**	4.7	63	0.2	63	0.0	**60**	0.4
P21	**46**	46.5	**46**	4.7	48	0.2	48	0.0	48	0.6
P22	**67**	103.2	**67**	4.7	**67**	0.2	**67**	0.0	**67**	0.5
P23	**55**	24.6	**55**	4.2	56	0.2	56	0.1	**55**	0.7
P24	**74**	32.6	**74**	3.1	**74**	0.2	74	0.0	**74**	0.5
P25	**31**	59,551.0	*30*	7.5	**31**	0.5	31	0.2	**31**	1.6

P26	**43**	20,663.1	*42*	8.3	**43**	0.6	43	0.1	**43**	1.8
P27	**43**	1,008.4	**43**	4.9	**43**	0.7	43	0.1	**43**	1.7
P28	**55**	582.3	*54*	4.4	**55**	0.6	55	0.1	**55**	1.2
P29	**29**	94,397.6	**29**	10.2	**29**	0.4	29	0.1	**29**	1.6
P30	**49**	27,609.2	50	8.6	**49**	0.5	49	0.2	**49**	1.6
P31	**42**	1,950.3	**42**	7.8	**42**	0.7	42	0.2	**42**	2.3
P32	**69**	3,716.4	*66*	7.3	**69**	0.5	69	0.1	**69**	1.2
P35	**73**	790,670.6	*68*	12.8	**73**	2.3	73	0.4	**73**	4.9
P36	**95**	57,012.0	*90*	117.9	**95**	1.6	95	0.2	**95**	3.2
P39	**68**	177,400.5	*67*	13.0	69	2.3	69	0.4	69	4.5
P40	**108**	211,216.3	*104*	15.1	**108**	1.7	108	0.2	**108**	3.0
Aver.	45.1	40,174.0	44.5	7.9	45.4	0.4	45.4	0.1	45.2	1.1

Table 2: Results of Cplex, literature, Tabu, GRASP and HGT

For the 36 instances where Cplex 11 was able to find the optimal solution, Tabu and GRASP procedures achieved the optimum value in 28 of them. HGT algorithm was able to get to the optimum with 33 instances. For the remaining suboptimal solutions, the maximum gap was 5.0%.

Table 3 shows the comparison between the best solution achieved by our Tabu, GRASP and HGT algorithms and the best results reported in [9], for the 60 instances where the optimal value is not known, i.e., P33, P34, P37, P38 and from P40 to P96. The first column shows the instance, the second column shows the best solution reported in [9], followed in the third column by its computational time. The fourth column shows the best cost obtained by the proposed algorithm, followed by its time, in the fifth. Finally, the sixth column indicates the algorithms that obtained this result. Lines Aver. in Table 3 show the average processing time and the average objective function values for each algorithm. All the computational times are expressed in seconds.

The GRASP algorithm proposed in this work was able to improve the known upper bounds 49 times, tied 2 times and decreased the best known upper bound, for instance in P62 by 12.4%. The maximum gap was 7.8%, also for P38. It yields, on average, solutions with a cost 1.6% less than the best known values from the literature.

12 Locations					
Inst.	**Lit_Best**	**CPU(Lit_Best)**	**Our_Best**	**CPU(Our_Best)**	**Best Result**
P33	53	17.2	**52**	0.8	GRASP, TS, HGT
P34	**72**	20.6	**72**	1.9	[9], GRASP, TS, HGT
P37	**47**	12.0	48	0.4	[9]
P38	**77**	26.6	83	0.3	[9]
P41	**78**	25.0	**78**	0.9	[9] , GRASP, TS, HGT
P42	104	20.0	**102**	0.7	GRASP, TS, HGT
P43	**110**	267.9	**110**	0.7	[9], GRASP, TS, HGT
P44	**137**	20.8	140	0.4	[9]
P45	**66**	47.6	**66**	1.2	[9], GRASP, TS, HGT
P46	**111**	39.6	116	1.1	[9]
P47	**111**	33.8	115	0.8	[9]
P48	**169**	23.7	171	0.5	[9]
Aver.	94.6	46.2	96.1	0.8	-

20 Locations					
Inst.	**Lit_Best**	**CPU(Lit_Best)**	**Our_Best**	**CPU(Our_Best)**	**Best Result**
P49	45	25.3	**44**	1.2	GRASP, TS, HGT
P50	63	25.5	**60**	1.2	GRASP, TS, HGT
P51	55	24.0	**53**	7.8	GRASP, HGT
P52	98	20.8	**89**	0.7	GRASP, TS, HGT
P53	49	36.2	**47**	1.7	GRASP, TS, HGT

P54	67	24.2	**62**	1.5	GRASP, TS, HGT
P55	63	22.6	**60**	1.8	GRASP, TS, HGT
P56	97	27.6	**89**	1.6	GRASP, TS, HGT
P57	67	81.2	**66**	4.0	GRASP, TS, HGT
P58	106	106.2	**100**	34.2	HGT
P59	101	96.6	**93**	3.7	TS, HGT
P60	159	70.8	**153**	2.4	GRASP, TS, HGT
P61	82	120.5	**74**	4.4	GRASP, TS, HGT
P62	129	100.2	**112**	3.1	TS, HGT
P63	121	68.2	**116**	7.1	TS, HGT
P64	190	68.5	**175**	4.9	TS, HGT
P65	105	106.5	**98**	89.4	HGT
P66	156	203.1	**150**	4.6	TS
P67	157	119.5	**145**	74.5	HGT
P68	234	84.6	**218**	33.5	GRASP, HGT
P69	112	112.5	**113**	7.0	TS, HGT
P70	178	262.9	**166**	14.6	TS, HGT
P71	170	176.2	**162**	9.1	TS, HGT
Aver.	113.2	88.0	106.3	15.2	-

32 Locations					
Inst.	**Lit_Best**	**CPU(Lit_Best)**	**Our_Best**	**CPU(Our_Best)**	**Best Result**
P72	265	128.1	**248**	52.0	GRASP, HGT
P73	74	130.4	**71**	54.8	GRASP, TS, HGT
P74	97	106.2	**90**	63.0	GRASP, TS, HGT
P75	110	116.4	**101**	107.6	HGT
P76	155	80.4	**144**	86.4	TS, HGT
P77	73	97.4	**70**	6.4	GRASP, TS, HGT
P78	101	163.2	**92**	11.4	TS, HGT
P79	110	117.7	**99**	128.4	HGT
P80	175	87.7	**163**	91.7	HGT
P81	119	266.8	**109**	342.1	HGT
P82	176	240.8	**162**	36.9	TS
P83	192	357.4	**183**	330.9	HGT
P84	282	247.8	**266**	210.1	HGT
P85	125	343.9	**117**	367.6	HGT
P86	192	540.5	**177**	343.4	HGT
P87	193	461.8	**181**	419.8	HGT
P88	302	342.3	**276**	236.6	HGT
P89	171	447.5	**159**	720.7	HGT
P90	262	696.7	**243**	25.2	TS
P91	284	400.7	**269**	570.0	HGT
P92	395	1458.4	**371**	380.7	HGT
P93	189	770.3	**176**	706.9	HGT
P94	281	887.0	**271**	568.3	HGT
P95	318	717.2	**301**	679.2	HGT
P96	464	663.6	**436**	49.7	TS
Aver.	204.2	405.9	191.0	272.4	-

Table 3: Best results of literature and our proposed algorithms

The Tabu algorithm proposed in this work yielded the best solutions in 48 instances, where the average gap reduction was 4.0% and the greatest reduction was 13.2%, for instance in P62. In four other instances, Tabu had the same performance as the best algorithm in the literature and, in the remaining 11 instances, the maximum gap was 7.8%, for P38.

Our proposed HGT algorithm was able to obtain lower cost solutions for 50 instances and to tie with the best algorithm in the literature in four instances. The average improvement obtained in the objective function was by 4.7%.

Considering all the 96 instances, the average percentual deviations from algorithms GRASP, TS and HGT to the best known solution was of 1.6, 2.2 and 2.8, respectively. Moreover, the optimal or best known solution was found for 79 instances by GRASP, for 80 instances by TS, for 86 instances by HGT, while the literature algorithm has found it only in 37 cases.

We can observe that, concerning time, Tabu search is more efficient than GRASP and HGT. On the average, nevertheless, these two algorithms are faster than those in the literature.

6 Conclusions and future work

The DSAP problem, relatively new in the literature, can model many important real-life problems where rearranging resources is a difficult or expensive task. It is also significant due to its relations with three well known hard combinatorial problems: QAP, DFLP and GQAP.

A new hybrid heuristic (HGT), based on GRASP and Tabu search, was proposed in this paper. In order to verify its efficiency, the computational results obtained were compared against the pure GRASP and Tabu search heuristics. The hybrid algorithm was able, on average, to yield solutions with better costs than the best solutions produced by GRASP or Tabu search.

When comparing the GRASP and the Tabu search algorithms, the latter was found to be the most efficient for the chosen parameters, both in processing time and in the quality of the solutions generated. We believe that GRASP can be improved through a more in-depth study of α value [12] or by using a reactive strategy, as was done with other problems of high complexity, as in [11].

It is important to note that, by using Cplex 11 over a previously defined formulation, we were able to correct the upper bounds P25, P26, P28, P32, P35, P36, P39 and P40, which are incorrectly presented in the literature. We were also able to solve to optimality some open instances.

As directions for future research we would suggest:

- Expanding this problem, merging it with the Resource Constrained Project Scheduling problem, in order to embrace other business problems.
- Testing other heuristic procedures, like VNS (variable neighborhood search), ILS (iterative local search) and hybrid versions using these metaheuristics;

Acknowledgements

We are grateful to CNPq/CT-Info and Universal, CAPES, FAPEMIG and FAPERJ for their support of this work.

References

[1] Drira A, Pierreval H, Hajri-Gabouj S, Facility layout problems: A survey, Annual Reviews in Control 31 (2007) 255–267.

[2] Garey M. R, Johnson D. S, Computers and intractability: A guide to the theory of NP-completeness, WH Freeman, (1979) New York.

[3] Gu J, Goetschalckx M, McGinnis L. F, Research on warehouse operation: A comprehensive review, European Journal of Operational Research 177 (2007) 1–21.

[4] Koopmans TC, Beckmann M. Assignment problems and the location of economic activities. Econometrica 1957;25:53-76.

[5] Lee C-G, Ma Z. (Department of Mechanical and Industrial Engineering, University of Toronto, Toronto, Ontario, M5S 3G8, Canada). The generalized quadratic assignment problem, Research Report; 2005.

[6] Loiola EM, Abreu NMMA, Boaventura-Netto PO, Hahn P., Querido TM. A survey for the quadratic assignment problem. European Journal of Operational Research 2007;176:657–690.

[7] McKendall Jr A, Noble J, Klein C. Simulated annealing heuristics for managing resources during planned outages at electric power plants. Computers & Operations Research 2005;32:107-125.

[8] McKendall Jr A, Jaramillo J. A tabu search heuristic for the dynamic space allocation problem. Computers & Operations Research 2006;33:768–789.

[9] McKendall Jr A. Improved tabu search heuristic for the dynamic space allocation problem. Computers & Operations Research 2008;35:3347–3359.

[10] Rosenblatt MJ. The dynamics of plant layout. Management Science 1986;32:76–86.

[11] Silva GC, Andrade MRQ, Ochi LS, Martins SL, Plastino A. New heuristics for the maximum diversity problem. Journal of Heuristics 2007;13:315-336.

[12] Resende MGC. Metaheuristic hybridization with Greedy Randomized Adaptive Search Procedures. In: Zhi-Long Chen and S. Raghavan, editors. TutORials in Operations Research. INFORMS; 2008, p. 295-319.