2008

Improved Tabu Search Heuristics for the Dynamic Space Allocation Problem

Alan McKendall

Available online at www.sciencedirect.com
ScienceDirect
Computers & Operations Research 35 (2008) 3347–3359
computers & operations research
www.elsevier.com/locate/cor

Improved Tabu search heuristics for the dynamic space allocation problem

Alan R. McKendall Jr.*

Department of Industrial and Management Systems Engineering, 325A Mineral Resources Building, West Virginia University, Morgantown, WV 26506, USA

Available online 12 March 2007

Abstract

The dynamic space allocation problem (DSAP) presented in this paper considers the task of assigning items (resources) to locations during a multi-period planning horizon such that the cost of rearranging the items is minimized. Three tabu search heuristics are presented for this problem. The first heuristic is a simple basic tabu search heuristic. The second heuristic adds diversification and intensification strategies to the first, and the third heuristic is a probabilistic tabu search heuristic. To test the performances of the heuristics, a set of test problems from the literature is used in the analysis. The results show that the tabu search heuristics are efficient techniques for solving the DSAP. More importantly, the proposed tabu search heuristic with diversification/intensification strategies found new best solutions using less computation time for one-half of all the test problems.

Keywords: Dynamic space allocation problem; Tabu search; Probabilistic tabu search; Diversification and intensification strategies; Metaheuristics

1. Introduction

The dynamic space allocation problem (DSAP) presented in this paper considers the task of assigning items (resources) to locations during a multi-period planning horizon such that the cost of rearranging the items is minimized. More specifically, when items are used to perform activities (required), they are assigned to workspaces. However, items are assigned to storage locations, when they are not used to perform an activity (idle). Therefore, the objective is to assign the required resources (activities) to workspaces and the idle resources to storage locations during a multi-period planning horizon such that total rearrangement cost is minimized. Total rearrangement cost is the sum of the transportation and preparation costs. Transportation cost is the sum of the products of all the distances the resources travel and unit costs (i.e., the costs of moving the resources one distance unit). In contrast, the cost of preparing the resources to be moved and unloading the resources is called preparation cost.

The DSAP presented in this paper was first defined by McKendall et al. [1]. It was used to efficiently assign maintenance (laydown, preventative maintenance, surveillance, etc.) activities and their required resources to workspaces as well as assign idle resources to storage locations in a reactor containment building during planned outages at a nuclear

* Fax: +1 304 293 4970.
E-mail address: armckendall@mail.wvu.edu.

0305-0548/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2007.03.002

power plant. More specifically, resources used to perform activities during planned outages are:

(1) plentiful, bulky, and difficult to move;
(2) consume a lot of space where space is very limited;
(3) require a single polar crane to be moved.

Therefore, assigning resources to work/storage spaces without sophisticated techniques resulted in high congestion and work crew interference. For instance, activities were delayed because resources (materials) were occupying areas where activities needed to be performed. As a result, the polar crane was needed to move the resources to other locations. However, oftentimes the polar crane was also needed to perform an activity. Therefore, either an activity is interrupted or other activities are delayed. Nevertheless, the polar crane was being over-utilized. In order to reduce congestion and work crew interference, McKendall et al. [1] defined the DSAP as the problem of assigning resources to work/storage spaces such that the total distance the resources travel throughout the duration of the outage (transportation cost) is minimized. The authors presented a mathematical model and two simulated annealing heuristics for the DSAP. However, McKendall and Jaramillo [2] presented five methods for constructing solutions for the DSAP, and the solutions were improved using a simple tabu search (TS) heuristic. Also, the authors added an additional restriction to the DSAP. That is, if a resource is idle for consecutive time periods, it must be assigned to the same storage location. There were two reasons for adding this restriction. First, it was used to reduce the number of times resources were moved. In other words, it was used to minimize preparation cost, since loading/unloading the resources to/from the polar crane may require a substantial amount of time. Second, restricting the assignment of idle resources to storage spaces decreases the size of the solution space. Therefore, the DSAP was easier to solve.

The assumptions for the DSAP presented in this paper are defined as follows.

(1) The locations of the work/storage spaces and the distances between locations are known.
(2) The resources required to perform each activity as well as the time periods the activities are performed are given.
(3) Each activity requires at least one resource and only one workspace.
(4) Only one activity can be performed in each workspace in each period, and the workspace is large enough to perform the activity and store the required resources.
(5) If an activity is performed in multiple periods, the activity must be assigned to the same workspace each period it is performed.
(6) The capacities of the storage locations are given.
(7) The objective is to minimize the sum of the transportation and preparation costs.

In assumption (2), the activities as well as the number of activities performed in each period are given. It is important to note that the number of activities performed in each period cannot exceed the number of workspaces, since the resource constrained project scheduling problem [3] is used to schedule the activities with respect to precedence constraints and restrictions on material handling equipment, work crews, materials, and workspaces. More specifically, the schedule of activities obtained from solving the resource constrained project scheduling problem is used as input for the DSAP. Considering assumption (3), if some activities do not require a resource to be performed, then these activities can be assigned to any available workspaces since they have no impact on the objective function value (OFV). Therefore, these activities can be ignored in the DSAP. Because of the following, assigning activities to workspaces can be modeled as a quadratic assignment problem (QAP).

- the number of activities in each period cannot exceed the number of workspaces,
- each activity is assigned to exactly one workspace (in assumption (3));
- only one activity can be performed in each workspace (in assumption (4));

Since activities are assigned to workspaces in multiple periods, this problem is not a traditional QAP (see [4] for more information about the QAP). In contrast, it is basically a dynamic facility layout problem (see [5] for a complete description of the DFLP). In order to minimize total rearrangement cost, utilization of the polar crane, congestion, and work crew interferences, assumption (5) is necessary. Also, it is not feasible to assign an activity to different workspaces each period it is performed. In assumption (6), the capacities of the storage locations are given. In other words, more than one idle resource can be stored in a storage location, but the capacity of the location cannot be exceeded. Therefore, the problem of assigning idle resources to storage locations is and can be modeled as a generalized quadratic assignment problem (GQAP) as defined in Lee and Ma [6]. The authors defined the GQAP as the problem of assigning pieces of

equipment to locations such that more than one piece of equipment can be assigned to a location without exceeding the capacity of the location. This problem is very similar to our problem of assigning idle resources to storage locations. However, since idle resources are assigned to storage locations in multiple periods, this makes our problem more general than the GQAP. As a result, the DSAP is a combination of the DFLP (a type of QAP) and a type of GQAP.

Assumptions (1)–(6) were also used to define the DSAP presented in McKendall et al. [1] and McKendall and Jaramillo [2]. However, the objective in both papers was to minimize transportation cost only, as mentioned earlier. In addition, McKendall and Jaramillo [2] gave the additional assumption that a resource must be assigned to the same storage location if idle in consecutive periods. This restriction was used to reduce preparation cost and the size of the solution space. However, in this paper, this assumption is omitted, and the preparation cost is explicitly considered in the objective function. Therefore, the size of the solution space presented in this paper and in McKendall et al. [1] is larger than the one presented in McKendall and Jaramillo [2].

Besides managing resources during outages at electric power plants, the DSAP has other potential applications. Similarly, it can be used to manage resources during the overhaul of complex equipment in other environments besides electric power plants such as manufacturing as in Gharbi et al. [7]. Also, the DSAP can be used to assign construction materials to work/storage spaces during the construction of buildings, to mention a few. For more information about assigning construction activities and resources to locations, see Zouein and Tommelein [8]. In summary, the DSAP has many applications and is a very difficult problem to solve, since it is a combination of two computational intractable problems: a type of QAP and a type of GQAP.

Therefore, the purpose of this paper is to present three efficient heuristics for the DSAP. The first heuristic is a simple basic TS heuristic. The second heuristic adds diversification and intensification strategies to the first, and the third heuristic is a probabilistic TS heuristic. In Section 2, a mathematical model is presented for the DSAP presented in this paper. Then the TS heuristics are presented in Section 3. In Section 4, the computational results are given on the performances of the proposed heuristics with respect to solution quality and computation time on a set of test problems taken from the literature. Last, Section 5 provides conclusions and future research directions.

2. Mathematical model for the DSAP

The mathematical model presented below for the DSAP is a modification of the model presented by McKendall et al. [1]. Here preparation cost is added to the objective function. Other small modifications are made in order to make the DSAP more general. Let

T is the set of time periods in the planning horizon such that $p \in T = 1, 2, \ldots, P$;
L is the set of all locations such that $k, l \in L = 1, 2, \ldots, N$;
W is the set of workspaces such that $w \in W$ and $W \subset L$;
S is the set of storage locations such that $s \in S$, $S \subset L$, and $W \cup S = L$;
A_p is the set of activities performed in period p;
R_j is the set of resources required to perform activity j;
I_p is the set of idle resources in period p;
Z is the set of all resources (required and idle) such that $r \in Z = 1, 2, \ldots, R$. In other words, Z is the union of the set of all required resources ($\bigcup_{\forall j} R_j$) and the set of all idle resources ($\bigcup_{\forall p} I_p$).
a_{prkl} is the sum of the cost of loading resource r at location k and the cost of unloading it at location l in period p;
d_{kl} is the distance from location k to l;
c_{prkl} is the unit cost of moving resource r from location k to l in period p;
q_r is the space requirements (e.g., number of items, volume, etc.) of resource r;
C_{ps} is the capacity (e.g., number of items, volume, etc.) of storage location s in period p;

$$x_{prk} = \begin{cases} 1 & \text{if resource } r \text{ is assigned to location } k \text{ in period } p, \\ 0 & \text{otherwise.} \end{cases}$$

$$y_{jw} = \begin{cases} 1 & \text{if activity } j \text{ is assigned to location } w, \\ 0 & \text{otherwise.} \end{cases}$$

Then the DSAP can be formulated as the following integer program:

$$\text{Minimize} \quad \sum_{r=1}^{R}\sum_{k=1}^{N}\sum_{\substack{l=1\\ l\neq k}}^{N}\sum_{p=1}^{P-1} a_{prkl}x_{prk}x_{p+1,rl} + \sum_{r=1}^{R}\sum_{k=1}^{N}\sum_{\substack{l=1\\ l\neq k}}^{N}\sum_{p=1}^{P-1} c_{prkl}d_{kl}x_{prk}x_{p+1,rl} \tag{1}$$

$$\text{Subject to} \quad \sum_{\forall s\in S} x_{prs} = 1 \quad \forall p, \forall r \in I_P, \tag{2}$$

$$\sum_{\forall r\in I_p} q_r x_{prs} \leqslant C_{ps} \quad \forall s \in S, \forall p, \tag{3}$$

$$\sum_{\forall w\in W} y_{jw} = 1 \quad \forall j, \tag{4}$$

$$\sum_{\forall j\in A_p} y_{jw} \leqslant 1 \quad \forall w \in W, \forall p, \tag{5}$$

$$\sum_{r\in R_j} x_{prw} = |R_j| y_{jw} \quad \forall p, \forall j \in A_p, \forall w \in W, \tag{6}$$

$$x_{prk} = 0 \text{ or } 1 \quad \forall p, \forall r, \forall k \in L, \tag{7}$$

$$y_{jw} = 0 \text{ or } 1 \quad \forall j, \forall w \in W. \tag{8}$$

The objective function (1) minimizes the sum of the preparation and transportation costs. Constraint set (2) ensures that every idle resource in each period is assigned to a storage location. Constraint set (3) guarantees that the storage capacity for each storage location in each period is not exceeded. It is important to note that the capacity of a storage location may change from period to period based on the completion of certain activities. For instance, in the construction of multi-story buildings, when the interior walls are constructed, the size of the work/storage locations may change. Furthermore, assigning each activity to only one workspace is considered in constraint set (4). To ensure that at most one activity is assigned to a workspace in each period, constraint set (5) is used. Constraint set (6) ensures that the required resources needed to perform each activity are assigned to the workspace where each activity is performed. Lastly, the restrictions on the decision variables are given in (7) and (8).

The formulation given above for the DSAP can be easily modified (restricted) to solve other related problems such as the DFLP and a type of GQAP. First, if all the resources (machines) are required to perform an activity (belong to a department), all locations are workspaces, and each activity (department) is assigned to one and only one workspace, then the DSAP becomes a DFLP (see [5] for a detailed description of the DFLP). That is, objective function (1) as well as constraint sets (4)–(8) can be used to solve a DFLP with simple modifications. Second, if only a single period is considered ($p = 1$), all the resources (pieces of equipment) are idle resources, all locations are storage locations, and a storage location can be assigned to more than one idle resource such that its capacity is not exceeded, then the DSAP is restricted to a GQAP (see [6] for a detailed description of a GQAP). In other words, objective function (1) and constraint sets (2), (3), and (7) can be used to solve a type of GQAP with multiple periods ($P > 1$). Hence, the DSAP is the integration of two types of combinatorial optimization problems: a type of QAP (i.e., DFLP–QAP with multiple periods in the planning horizon) and a type of GQAP (i.e., GQAP with multiple periods in the planning horizon). Therefore, the DSAP is a computationally intractable problem, and only small size problems can be solved in reasonable computation time using the above formulation. A number of DSAP instances with six locations (three workspaces and three storage spaces), 20 periods, nine resources, and 20 activities were solved in reasonable time using the above formulation and a branch-and-bound algorithm (CPLEX Solver, version 6.0). However, only one DSAP instance with 12 locations (six workspaces and six storage locations), 10 periods, 18 resources, and 10 activities was solved to optimality using the above formulation, which took 57.3 h on a Pentium IV 1.8 GHz PC. As a result, three TS heuristics are developed for the DSAP and will be presented next.

3. TS heuristics for the DSAP

Glover [9] was the first to introduce the TS heuristic. Since then, TS has been successfully used to solve many types of combinatorial optimization problems. For instance, Skorin-Kapov [10] was the first to use TS to solve the QAP. Also, Taillard [11], Battiti and Tecchiolli [12], Skorin-Kapov [13], Chiang and Kouvelis [14], and Chiang and Chiang [15] presented TS heuristics for the QAP, to mention a few. However, Kaku and Mazzola [16] presented the only TS heuristic for the DFLP, known to the author. Osman [17], Laguna et al. [18], Diaz and Fernandez [19], and Higgins [20] presented TS heuristics for the generalized assignment problem. Also, McKendall and Jaramillo [2] was the first to present a TS heuristic for the DSAP presented in this paper. Since TS heuristics with different variations performed well on related problems, three different TS heuristics are presented for the DSAP. First, a simple basic TS heuristic, called TS I, is developed for the DSAP. It is an improvement over the simple TS heuristic presented in McKendall and Jaramillo [2] for the DSAP, since it does not fix the resources that are idle in consecutive time periods to the same storage locations and preparation cost is explicitly considered in the objective function. Three initial solutions are constructed using the three best methods presented in McKendall and Jaramillo [2], and the solutions are improved using the TS I heuristic presented in this paper. As in the literature, in McKendall and Jaramillo [2] their TS heuristic performed better on higher quality initial solutions. Therefore, the second heuristic, called TS II, adds diversification and intensification strategies to TS I so that it can obtain high quality solutions even if only low quality initial solutions are available. As a result, it uses dynamic tabu list size, frequency-based memory, and an intensification strategy similar to the TS heuristic presented for the QAP in Chiang and Kouvelis [14]. The third heuristic is another modification of the TS I heuristic. Instead of accepting the best admissible move while searching neighborhoods of solutions, it randomly selects a move from the top M admissible moves. This is known as the probabilistic TS heuristic and is called the TS III heuristic, in this paper. A probabilistic TS heuristic for the QAP was presented by Chiang and Chiang [15]. Next, the proposed TS heuristics for the DSAP are presented but first the solution representation and neighborhood structure are defined.

A feasible solution of the DSAP is represented as follows:

$$\pi = (\pi^1, \pi^2, \ldots, \pi^P),$$

where π is the solution of the DSAP, $\pi^p = \{[\pi^p_A], [\pi^p_I]\}$ is the assignment of activities/idle resources to work/storage spaces in period $p \in (1, 2, \ldots, P)$, $\pi^p_A = [\pi^p_A(R_1), \pi^p_A(R_2), \ldots, \pi^p_A(R_K)]$ is the assignment of activities to workspaces in period p, $\pi^p_A(R_k)$ is the activity and its required resources assigned to location (workspace) k, K is the number of workspace locations where locations $1, 2, \ldots, K$ are workspaces, $\pi^p_I = [\pi^p_I(R_{K+1}), \pi^p_I(R_{K+2}), \ldots, \pi^p_I(R_N)]$ is the assignment of idle resources to storage locations in period p, $\pi^p_I(R_{K+s})$ is the set of idle resources assigned to location $K + s$ (storage location s) where $s \in (1, 2, \ldots, S)$, S is the number of storage locations where locations $K + 1, K + 2, \ldots, K + S$ are storage locations and $N = K + S$.

For example, consider the following DSAP instance with two periods in the planning horizon (i.e., $P = 2$) and six locations (i.e., $N = 6$) where locations 1–3 are workspaces (i.e., $K = 3$) and locations 4–6 are storage locations (i.e., $S = 3$). In period 1, activities 1, 2, and 3 are scheduled and require resources (1, 2), (3), (4, 5), respectively. As a result, the other resources, 6–9 are idle resources. In period 2, activities 1, 3, and 4 are scheduled and require resources (1, 2), (4, 5), and (6), respectively. Therefore, resources 3, 7, 8, and 9 are idle resources. Furthermore, the capacity of the storage locations is two resources. Therefore, a feasible solution for this DSAP instance is represented as follows:

$$\pi = (\{[(1, 2), (3), (4, 5)], [(6, 7), (8, 9), (\emptyset)]\}\{[(1, 2), (6), (4, 5)], [(7), (8, 9), (3)]\}).$$

In other words, activities 1 (1, 2), 2 (3), and 3 (4, 5) are assigned to workspace locations 1, 2, and 3, respectively, in period 1. More specifically, since activity 3 and its required resources (4, 5) are assigned to location (workspace) 3 in period 1, $\pi^1_A(R_3) =$ activity 3 (4, 5). Also, idle resources (6, 7) and (8, 9) are assigned to locations 4 (storage location 1) and 5 (storage location 2), respectively. However, location 6 (storage location 3) is empty. More specifically, idle resources 6 and 7 are assigned to location 4 (storage location 1) in period 1; therefore, $\pi^1_I(R_4) = (6, 7)$. Since activities 1 (1, 2) and 3 (4, 5) are assigned to workspace locations 1 and 3 in period 1, they must be assigned to the same locations in period 2 because of assumption (5). As a result, activity 4 (6) is assigned to workspace location 2 in period 2 (i.e., $\pi^2_A(R_2) =$ activity 4 (6)). In addition, idle resources (7), (8, 9), and (3) are assigned to locations 4, 5, and 6, respectively.

Table 1
Distances from locations k to l

From/To	1	2	3	4	5	6
1	–	1	2	1	2	3
2	1	–	1	2	1	2
3	2	1	–	3	2	1
4	1	2	3	–	1	2
5	2	1	2	1	–	1
6	3	2	1	2	1	–

If additional cost information is given such as

- the unit cost of moving a resource one distance unit is \$1,
- the cost of loading/unloading a resource is \$2, and
- the distances between locations (see Table 1)

then the total rearrangement cost $f(\pi)$ can be determined. Since resources 3 and 6 are moved from locations 2 and 4 in period 1 to locations 6 and 2 in period 2, the transportation cost is \$4 (resources 3 and 6 travel 2 distance units each) and preparation cost is \$4 (two resources are moved and loaded/unloaded). Thus, total rearrangement cost is \$8 (i.e., $f(\pi) = 8$).

Once a feasible solution π is available, the neighborhood of π, $N(\pi)$, can be explored to obtain the next solution. In other words, an operation or move m is performed on π such that a different solution π_{next} is obtained. Since the DSAP is a combination of two problems, moves are defined for each subproblem. For assigning activities to workspaces, there are three possible moves.

M(1) Interchange the locations (workspaces) of two activities in one or more periods.
M(2) In one or more periods, remove an activity from a location (workspace) and assign it to an available (empty workspace) location.
M(3) A combination of both M(1) and M(2).

For assigning idle resources to storage locations, the following moves are considered.

M(4) Interchange the locations (storage locations) of two idle resources assigned to different storage locations in a period.
M(5) In a period, remove an idle resource from a location (storage location) and assign it to a different location such that the capacity of the location is not exceeded.

These moves were defined and discussed in McKendall et al. [1]. In order to obtain π_{next}, all possible moves M(1)–M(5) are considered on feasible solution π, and all the possible solutions, obtained from performing all the possible moves on solution π for each period, is defined as $N(\pi)$. Furthermore, the best solution $\pi\sim$ in $N(\pi)$ is defined as π_{next}. In other words, π_{next} is the best solution in $N(\pi)$ with respect to the OFV (i.e., $f(\pi_{\text{next}}) \leqslant f(\pi\sim)$ for all $\pi\sim \in N(\pi)$). Then π is set to π_{next}, and the process is repeated until a non-improving solution π_{next} is obtained (i.e., $f(\pi_{\text{next}}) \geqslant f(\pi)$). It is obvious that this steepest descent local search technique may often terminate at a poor local optimum. Therefore, a basic TS heuristic is given below which allows iterating through non-improving solutions so that the global optimum may be obtained, but first the basic components of the TS heuristic are given.

One of the main differences between the simple local search technique discussed in the previous paragraph and the TS heuristic (as well as other meta-heuristics) is that the TS heuristic does not usually terminate at a poor local optimum. More importantly, the TS heuristic iterates through non-improving solutions. That is, at iteration k, if π_{next} is the best (non-improving) solution in $N(\pi)$ obtained from performing move m^* such that $f(\pi_{\text{next}}) > f(\pi)$, then π is set to π_{next}, and the process is continued until one (or more) *stopping criterion* (criteria) is (are) met. Examples of some stopping rules are:

(1) Terminate after a predefined number of iterations has been performed.
(2) Terminate after heuristic has ran for a predetermined number of time units.
(3) Terminate after there has been no improvement in the best solution for a predefined number of consecutive iterations.

If the current solution π is a non-improving solution, π_{next} at iteration $k+1$ will cycle back to solution π obtained at iteration $k-1$. In order to prevent cycling in the TS heuristic, *short-term* (*recency*) *memory* is used to forbid moves which might lead to recently visited solutions. More specifically, to prevent cycling, the most recent moves are *forbidden* (or declared *tabu*) for a certain number of iterations. The number of iterations the move is tabu is defined as the *tabu list size* (e.g., *TLS*), and the list of recent moves and its TLS is maintained in the *tabu list* (e.g., *Tabu-A*$[i][j]$, for $i<j$). However, if a move $m(i, j)$, which gives the best solution ever, is tabu restricted, then the tabu restriction may be overridden, and the move is performed. The condition that allows the tabu restriction placed on a move to be overridden is called an *aspiration criterion*. In summary, the best allowable (*admissible*) move $m*(i, j)$ selected is either a tabu move that gives the best solution ever or the best move that is not classified as tabu (non-tabu). At each iteration, the best solution found and its costs are saved if necessary.

A basic TS heuristic for the DSAP, called TS I, is given below.

Step 0: Construct three initial solutions using MFA, MRC I, and MRC II construction algorithms presented in McKendall and Jaramillo [2]. Then perform the following steps for each of the initial solutions.

Step 1: For an initial solution obtained in the previous step, assign the initial solution to π (current solution). Set the best solution $\pi_{\text{best}}=\pi$ and $z_{\text{best}}=f(\pi_{\text{best}})$. Initialize the activity tabu list, *Tabu-A*$[i][j]$ for $i<j$, and the idle resource tabu lists for each period p, *Tabu-I*$[i][j][p]$ for $i<j$. Also, initialize the iteration counter k (i.e., $k=0$), the counter l used to track the number of consecutive iterations without improving the best solution (i.e., $l=0$), tabu list size for activity moves (TLS_A), tabu list size for idle resource moves (TLS_I), and N (maximum number of consecutive iterations allowed without improving the best solution).

Step 2: Set $k=k+1$, and find the best admissible move m^* (either an activity or idle resource move) which gives the solution π_{next} from $N(\pi)$.

Step 3: Set $\pi=\pi_{\text{next}}$, and update tabu list which keeps track of (either the activity or idle resource) move $m*(i, j)$ or $m*(i, j, p)$ such that the entry in tabu list is updated to $k+TLS_{\text{A}}$ or $k+TLS_{\text{I}}$ for $i<j$.

Step 4: If $f(\pi)<f(\pi_{\text{best}})$, then set $\pi_{\text{best}}=\pi$ and $l=0$. Else, set $l=l+1$.

Step 5: If $l>N$, then terminate heuristic. Otherwise, go to step 2.

The setting of the heuristic parameters N, TLS_{A}, and TLS_{I} were obtained experimentally. The parameter N is set to a value large enough so that "good" solutions can be obtained. For setting the tabu list size parameters TLS_A and TLS_I, careful consideration were taken so that the values were not too low or too high. In other words, if the TLSs are too low, then cycling to recently visited solutions can occur. On the other hand, if the TLSs are too high, then the possible number of admissible moves may be too restrictive. Therefore, it may be more advantageous to vary the TLSs as in Taillard [11], Skorin-Kapov [13], and Chiang and Kouvelis [14] for the QAP. Next, a TS heuristic consisting of dynamic TLSs, a different diversification strategy, and an intensification strategy are presented for the DSAP. This TS heuristic is called TS II in this paper.

In the TS II heuristic presented in this paper for the DSAP, the TLSs are variables. More specifically, the TLSs in each iteration k vary between a lower bound (LB) and an upper bound (UB). That is, $LB_A \leqslant TLS_{Ak} \leqslant UB_A$ and $LB_I \leqslant TLS_{Ik} \leqslant UB_I$. The actual TLS varies from one iteration to another depending on $PR(\pi)$, the percent reduction of $f(\pi_{\text{next}})$ from $f(\pi)$. At each iteration k, use the following rules for setting the tabu list size TLS_{Ak} or TLS_{Ik}.

(1) If $PR(\pi)<0\%$, then $f(\pi_{\text{next}})$ is worse than $f(\pi)$. Set either $TLS_{Ak}=TLS_{A,k-1}$ or $TLS_{Ik}=TLS_{I,k-1}$.
(2) If $0\% \leqslant PR(\pi) \leqslant \alpha\%$, then $f(\pi_{\text{next}})$ is either the same or a reduction from $f(\pi)$ but not a significant reduction. Set either $TLS_{Ak}=LB_A+(UB_A-LB_A)PR(\pi)/\alpha$ or $TLS_{Ik}=LB_I+(UB_I-LB_I)PR(\pi)/\alpha$.
(3) If $\alpha\% \leqslant PR(\pi)<\beta\%$, then $f(\pi_{\text{next}})$ is a significant reduction from $f(\pi)$. Set either $TLS_{Ak}=UB_A$ or $TLS_{Ik}=UB_I$.
(4) If $PR(\pi) \geqslant \beta\% > \alpha\%$, then $f(\pi_{\text{next}})$ is a relatively large reduction from $f(\pi)$. Set either $TLS_{Ak}=LV_A$ or $TLS_{Ik}=LV_I$, for LV=large value.

These rules were adapted from Chiang and Kouvelis [14] for the QAP and modified for the DSAP.

Besides using dynamic TLSs, the TS II heuristic uses a different diversification strategy than TS I. In TS I, the diversification strategy is to use diverse starting solutions. However, frequency-based (long-term) memory is used in TS II, as in Skorin-Kapov [10] and Chiang and Kouvelis [14] for the QAP. More specifically, the tabu lists *Tabu-A*$[i][j]$ and *Tabu-I*$[i][j][p]$, for $i>j$, is used to keep track of the frequency of the moves $m(i, j)$, and the penalty functions defined below is used to penalize non-improving activity and idle resource moves,

respectively

$$f_A(i, j) = \begin{cases} 0 & \text{if } f(\pi_{\text{next}}) < f(\pi) \text{ or } m \in (\text{M}(4), M(5)), \\ p_A TabuA[i][j] \text{ for } i > j & \text{otherwise}, \end{cases}$$

$$f_I(i, j, p) = \begin{cases} 0 & \text{if } f(\pi_{\text{next}}) < f(\pi) \text{ or } m \in (\text{M}(1), M(2), \text{M}(3)), \\ p_I TabuI[i][j][p] \text{ for } i > j & \text{otherwise}, \end{cases}$$

where p_A is the penalty value for a non-improving activity move, p_I is the penalty value for a non-improving idle resource move.

Therefore, the following modified objective function is used to determine the cost for all $\pi\sim \in N(\pi)$ when the best admissible move m^* is a non-improving move:

$$F(\pi\sim) = f(\pi\sim) + f_A(i, j) + f_I(i, j, p).$$

The TS II heuristic uses an intensification strategy similar to the one used in Chiang and Kouvelis [14] for the QAP. The basic idea behind the intensification strategy is not to allow (fix) moves which result in large percentage reductions of at least $\gamma\%$ from the best found solution (i.e., $PR(\pi_{\text{best}}) \geqslant \gamma\%$), after η iterations have already been performed. It is important to invoke the intensification strategy after "good" solutions are obtained, since initially $PR(\pi_{\text{best}})$ may be large for a number of moves. Thus, exploring "poor" solutions, initially, is a waste of computation time. Nevertheless, these moves cannot be performed (i.e., not allowed) until the moves yield percent reductions $PR(\pi_{\text{best}})$ better than that at which they were fixed.

The TS II and III heuristics use the same algorithm presented in McKendall et al. [1] for constructing solutions. In the first period, the first activity, second activity, and so on, are assigned to the first workspace, second workspace, and so on, respectively. In the second period, if one or more activities are also performed in the first, assign these activities to the same workspaces as in the first period. The other activities are assigned to the first available workspaces. This process is repeated until all the activities in each period are assigned to workspaces. Next, the idle resources are assigned to storage locations. In period 1, the idle resources with the lowest numbers are assigned to the first storage location such that its capacity is not exceeded. The idle resources with the next lowest numbers are assigned to the second storage location without exceeding its capacity. Continue in this fashion until all idle resources are assigned to storage locations in period 1. This process is repeated for each period. This method is called the first assignment method, and it is obvious that this method often produces poor quality solutions.

Finally, the probabilistic TS heuristic for the DSAP is presented, called the TS III heuristic. Chiang and Chiang [15] presented a probabilistic TS heuristic for the QAP, and Lim et al. [21] presented one for a crane scheduling problem. The only difference between the TS I and TS III heuristics is how the admissible move m is selected. Recall in TS I, all the possible moves are evaluated, and the best admissible move $m*$ is selected to obtain π_{next}. However, in TS III, all the possible moves are evaluated and ranked, and the top M admissible moves are considered as candidate moves. Then the following procedure is used to select a move m from the candidate list of moves such that π_{next} is obtained from performing move m.

Step 1: Consider the first move m in the candidate list.
Step 2: Accept the move m with probability p. If the move is accepted, then this move is selected as the admissible move and exit procedure. Else, go to step 3.
Step 3: Go to the next move in the candidate list and set as m. If there are no more candidate moves in the list, select the best move from the list with respect to move cost. Else, go to step 2.

Since the TS I heuristic performs less operations per iteration than both the TS II and TS III, the TS I heuristic is ran long enough such that "good" solutions are obtained, and the heuristic runs until there has been no improvement in the best solution for N number of consecutive iterations. The computation times are recorded, and the run times for the TS I heuristic is used as the stopping criterion for the TS II and TS III heuristics so that all the heuristics have the same average run time and can be compared fairly. Although the TS II heuristic may perform more operations at each iteration, it is computationally efficient since it requires less iterations to obtain higher quality solutions. This will be illustrated and discussed in the next section.

4. Computational results

The set of test problems used to test the heuristics presented in McKendall et al. [1] and McKendall and Jaramillo [2] was used to test the performances of the proposed heuristics. The data set consists of 96 test problems which contain problems with 6, 12, 20, and 32 locations and 9, 18, 30, and 48 resources, respectively, each with 10, 15, and 20 periods. Half of the locations are workspaces and the other half are storage locations. Each of the storage locations has a maximum capacity of three resources, and the number of required resources per activity varies between 1 and 3. Also, the number of activities ranges between 6 (for smaller problems) and 87 (for larger problems). In order to compare the results of the heuristics presented in McKendall et al. [1] and McKendall and Jaramillo [2], which both considered minimizing total transportation cost, preparation costs are not considered in this analysis. The proposed heuristics were programmed using the C++ programming language, and the set of test problems were solved on a Pentium IV 2.4 GHz PC.

Tables 2–5 summarize the results obtained by the TS I, II, and III heuristics, as well as the heuristics presented by McKendall and Jaramillo [2] using a TS heuristic and McKendall et al. [1] using SA heuristics. More specifically, column two gives the average run time for the proposed heuristics. Columns 3, 4, and 5 give the OFVs for the TS I heuristic using construction algorithm MFA, MRC I, and MRC II, respectively. The OFV for the best solution obtained using the TS I heuristic is given in the column labeled "TS I Best." In the columns labeled "TS II" and "TS III," the OFV of the best solution obtained using the TS II (1 run) and TS III (with 5 runs) are given, respectively. Under "Old TS" and "Time," the OFV of the best solution obtained using the TS heuristic in McKendall and Jaramillo [2] is given as well as the average computation time of the 5 runs, respectively. Under "SA" and "Time," the OFV of the best solution obtained using the SA heuristics presented in McKendall et al. [1] and average run times (of 18 runs) are given, respectively. In the last column, the percent deviation the OFV of the best solution obtained from the proposed heuristics is below the OFV of the best solution obtained from SA and Old TS is given, for each test problem, under "%". Also, the bold numbers indicate the best OFV obtained for each test problem.

Table 2
Heuristic results for problems with six locations

No.	Time	TS I MFA	TS I MRCI	TS I MRCII	TS I Best	TS II	TS III	Old TS	Time	SA	Time	%
1	0.03	**16**	**16**	**16**	**16**	**16**	**16**	**16**	0.01	**16**	0.29	0.0
2	0.04	**25**	**25**	**25**	**25**	**25**	**25**	**25**	0.01	**25**	0.32	0.0
3	0.03	**18**	**18**	**18**	**18**	**18**	**18**	**18**	0.01	**18**	0.27	0.0
4	0.03	**25**	**25**	**25**	**25**	**25**	**25**	**25**	0.01	**25**	0.27	0.0
5	0.06	**16**	**16**	**16**	**16**	**16**	**16**	**16**	0.01	**16**	0.43	0.0
6	0.08	**27**	**27**	**27**	**27**	**27**	**27**	**27**	0.01	**27**	0.45	0.0
7	0.09	**16**	**16**	**16**	**16**	**16**	**16**	**16**	0.01	**16**	0.49	0.0
8	0.06	**31**	**31**	**31**	**31**	**31**	**31**	**31**	0.01	**31**	0.39	0.0
9	0.07	**25**	**25**	**25**	**25**	**25**	**25**	**25**	0.03	**25**	0.48	0.0
10	0.08	**46**	**46**	**46**	**46**	**46**	**46**	**46**	0.03	**46**	0.57	0.0
11	0.07	**32**	**32**	**32**	**32**	**32**	**32**	**32**	0.03	**32**	0.81	0.0
12	0.08	**41**	**41**	**41**	**41**	**41**	**41**	**41**	0.03	**41**	0.59	0.0
13	0.10	**28**	**28**	**28**	**28**	**28**	**28**	**28**	0.03	**28**	0.59	0.0
14	0.07	**45**	**45**	**45**	**45**	**45**	**45**	**45**	0.03	**45**	0.51	0.0
15	0.08	**35**	**35**	**35**	**35**	**35**	**35**	**35**	0.04	**35**	0.49	0.0
16	0.07	**49**	**49**	**49**	**49**	**49**	**49**	**49**	0.03	**49**	0.51	0.0
17	0.29	**35**	**35**	**35**	**35**	**35**	**35**	**35**	0.06	**35**	1.32	0.0
18	0.19	**60**	**60**	**60**	**60**	**60**	**60**	**60**	0.05	**60**	1.05	0.0
19	0.20	**46**	**46**	**46**	**46**	**46**	**46**	**46**	0.05	**46**	1.16	0.0
20	0.18	**60**	**60**	**60**	**60**	**60**	**60**	**60**	0.05	**60**	0.99	0.0
21	0.18	47	47	47	47	47	47	47	0.06	**46**	1.01	**2.2**
22	0.18	**67**	**67**	**67**	**67**	**67**	**67**	**67**	0.06	**67**	1.13	0.0
23	0.16	**55**	**55**	**55**	**55**	**55**	**55**	**55**	0.07	**55**	1.03	0.0
24	0.12	**74**	**74**	**74**	**74**	**74**	**74**	**74**	0.05	**74**	1.1	0.0

Table 3
Heuristic results for problems with 12 locations

No.	Time	TS I MFA	TS I MRCI	TS I MRCII	TS I Best	TS II	TS III	Old TS	Time	SA	Time	%
25	0.29	35	33	32	32	**30**	31	35	0.19	34	2.39	−11.8
26	0.32	48	44	43	43	**42**	44	48	0.20	47	2.44	−10.6
27	0.19	**43**	45	45	**43**	44	**43**	**43**	0.20	**43**	2.43	0.0
28	0.17	55	55	55	55	**54**	**54**	55	0.19	55	2.31	−1.8
29	0.39	**29**	**29**	**29**	**29**	**29**	**29**	**29**	0.22	**29**	2.66	0.0
30	0.33	51	**50**	**50**	**50**	**50**	**50**	51	0.17	**50**	2.58	0.0
31	0.30	43	**42**	**42**	**42**	**42**	**42**	43	0.18	**42**	2.74	0.0
32	0.28	69	70	68	68	**66**	68	69	0.17	69	3.16	−4.3
33	0.66	55	56	**53**	**53**	**53**	**53**	55	0.52	59	4.98	−3.6
34	0.79	73	73	74	73	**72**	73	73	0.43	79	5.23	−1.4
35	0.49	74	73	75	73	**68**	70	74	0.65	73	4.39	−6.8
36	0.43	95	98	95	95	94	94	95	0.46	**90**	4.53	**4.4**
37	0.46	52	51	49	49	**47**	48	52	0.50	54	7.54	−9.6
38	1.02	87	80	88	80	**77**	81	87	0.49	83	8.76	−7.2
39	0.50	68	68	**67**	**67**	**67**	**67**	68	0.55	71	6.6	−1.5
40	0.58	108	108	108	108	105	**104**	108	0.43	108	6.68	−3.7
41	0.96	83	80	80	80	**78**	79	83	1.16	85	8.96	−6.0
42	0.77	108	109	112	108	105	**104**	108	1.41	113	10.09	−3.7
43	0.98	111	111	113	111	112	111	111	1.24	**110**	10.29	**0.9**
44	0.80	139	143	141	139	**137**	**137**	139	0.79	140	8.72	−1.4
45	1.83	68	67	69	67	**66**	67	68	0.91	74	12.1	−2.9
46	1.52	116	114	114	114	**111**	114	116	0.89	122	13.44	−4.3
47	1.30	117	118	118	117	**111**	112	117	1.00	116	11.06	−4.3
48	0.91	173	172	171	171	170	**169**	173	1.17	171	9.5	−1.2

Table 4
Heuristic results for problems with 20 locations

No.	Time	TS I MFA	TS I MRCI	TS I MRCII	TS I Best	TS II	TS III	Old TS	Time	SA	Time	%
49	0.97	**45**	47	46	**45**	46	**45**	**45**	1.78	49	15.24	0.0
50	0.98	66	66	66	66	64	**63**	66	2.19	69	15.65	−4.5
51	0.92	60	60	57	57	56	**55**	60	2.48	57	14.15	−3.5
52	0.80	**98**	100	99	**98**	99	99	**98**	1.36	**98**	13.65	0.0
53	1.39	50	50	52	50	**49**	51	50	1.66	52	16.15	−2.0
54	0.93	72	72	71	71	**67**	69	72	1.73	72	15.89	−6.9
55	0.87	**63**	**63**	**63**	**63**	**63**	**63**	**63**	2.15	**63**	14.03	0.0
56	1.06	99	98	98	98	**97**	98	99	1.65	98	13.77	−1.0
57	3.12	71	69	69	69	**67**	**67**	71	5.42	76	24.03	−5.6
58	4.08	110	111	109	109	**106**	108	110	4.72	112	22.89	−3.6
59	3.71	103	105	107	103	**101**	**101**	103	4.17	103	23.11	−1.9
60	2.72	165	165	161	161	**159**	160	165	4.56	162	20.4	−1.9
61	4.63	87	84	**82**	**82**	**82**	**82**	87	3.97	84	24.64	−2.4
62	3.85	139	135	**129**	**129**	**129**	131	139	3.54	137	22.11	−5.8
63	2.62	126	126	126	126	**121**	124	126	5.51	127	23.34	−4.0
64	2.63	192	**190**	192	**190**	**190**	**190**	192	5.45	**190**	21.93	0.0
65	4.09	109	114	106	106	**105**	**105**	109	8.34	120	30.13	−3.7
66	7.80	157	157	163	157	**156**	157	157	8.85	167	35.78	−0.6
67	4.59	165	161	158	158	**157**	158	165	8.81	164	30.48	−4.3
68	3.25	237	240	240	237	**234**	**234**	237	8.34	244	25.22	−1.3
69	4.32	125	127	120	120	116	**112**	125	7.21	133	27.89	−10.4
70	10.10	200	195	191	191	181	**178**	200	8.21	191	43.09	−6.8
71	6.77	172	174	175	172	**170**	**170**	172	9.41	179	25.62	−1.2
72	4.92	277	279	268	268	**265**	268	277	8.71	274	24.41	−3.3

Table 5
Heuristic results for problems with 32 locations

No.	Time	TS I MFA	TS I MRCI	TS I MRCII	TS I Best	TS II	TS III	Old TS	Time	SA	Time	%
73	5.01	75	77	**74**	**74**	**74**	76	75	18.86	85	26.55	−1.3
74	4.08	102	99	105	99	**97**	101	102	11.60	108	25.89	−4.9
75	4.47	113	111	116	111	**110**	**110**	113	15.72	117	23.92	−2.7
76	3.09	161	157	161	157	156	**155**	161	15.19	160	21.42	−3.1
77	3.74	**73**	77	74	**73**	**73**	74	**73**	15.58	79	19.07	0.0
78	6.27	118	107	107	107	**101**	102	118	17.20	111	23.91	−9.0
79	4.52	**110**	115	115	**110**	**110**	**110**	**110**	16.16	116	24.83	0.0
80	3.37	183	179	**175**	**175**	**175**	179	183	15.14	182	25.14	−3.8
81	10.25	130	126	126	126	121	**119**	130	31.55	140	36.06	−8.5
82	9.25	184	**176**	**176**	**176**	181	180	184	32.96	183	38.82	−3.8
83	13.73	204	198	198	198	**192**	**192**	204	37.33	200	42.12	−4.0
84	9.52	294	285	**282**	**282**	**282**	288	294	30.25	287	32.35	−1.7
85	13.21	135	129	129	129	**125**	128	135	30.46	144	39.59	−7.4
86	20.76	207	205	204	204	**192**	**192**	207	40.24	209	52.19	−7.2
87	17.74	195	204	197	195	**193**	199	195	34.51	205	34.44	−1.0
88	13.15	318	304	305	304	**302**	303	318	30.34	**302**	33.78	0.0
89	17.19	175	193	180	175	**171**	173	175	70.48	195	61.9	−2.3
90	26.76	269	274	270	269	**262**	268	269	42.95	278	74.94	−2.6
91	15.39	297	298	290	290	**284**	288	297	67.12	293	62.04	−3.1
92	17.84	407	405	397	397	396	400	407	56.02	**395**	52.9	**0.3**
93	29.59	195	191	205	191	**189**	192	195	36.43	211	76.5	−3.1
94	34.07	298	302	294	294	**281**	285	298	63.21	298	65.38	−5.7
95	27.55	327	323	**318**	**318**	**318**	**318**	327	68.07	332	60.06	−2.8
96	25.49	487	499	465	465	**464**	469	487	71.92	485	52.55	−4.3

In Table 2, the results for the test problems with six locations are given. Although all the heuristics performed equally as well, the SA heuristic obtained the best solution for all 24 test problems while the other heuristics obtained 23 out of the 24. Therefore, the SA heuristic performs slightly better than the other heuristics. However, the average run time of 0.68 min for the SA heuristic is higher than the average run times of 0.11 and 0.03 for the proposed and the Old TS heuristics, respectively. Nevertheless, the run times for the first 24 problems are low for all the heuristics; therefore, SA is the preferred heuristic. Also, the best solutions obtained for the 6-location problems were proven to be the optimal solutions in McKendall et al. [1].

In Table 3, the results are shown for the test problems with 12 locations, and only the best solution for problem 27 has been proven to be the optimal. Nevertheless, the TS II heuristic obtained the best solutions for 18 of the 24 problems and 15 are new best solutions. The TS III, TS I, SA, and Old TS obtained the best solutions for 11, 6, 6, and 2 test problems, respectively. Therefore, the TS II heuristic clearly out-performs the other heuristics with respect to solution quality. Also, the average percent deviation the OFVs of the best solutions obtained from the proposed heuristics are below the OFVs of the best solutions obtained using SA and Old TS is 3.4%. Considering computation time, the average run times for the proposed TS heuristics, Old TS, and SA are 0.68, 0.59, and 6.40 min, respectively. It is obvious that the SA heuristic is computationally costly compared to the other heuristics, considering that 6.40 min is the average computation time of each of the 18 runs for each of the 24 test problems. Also, the Old TS heuristic required an average of 0.59 min for each of the 5 runs. The TS I and TS III heuristics required an average of 0.68 min for each of the 3 and 5 runs, respectively. However, the TS II heuristic required an average of 0.68 min of computation time with only one run for each of the 24 test problems. Hence, the TS II heuristic is preferred over all of the other heuristics for the test problems with 12 locations.

In Table 4, the results for the 20-location test problems are given. The TS II heuristic obtained the best solutions for 18 of the 24 problems and 16 are new best solutions. The TS III, TS I, TS Old, and SA obtained the best solutions for 13, 6, 3, and 3, respectively. In addition, the average percent deviation the OFVs of the proposed heuristics are below the OFVs of the SA and TS old heuristics is 3.12%. The average run times of the proposed heuristics, TS old, and SA are 3.38, 5.01, and 22.65 min, respectively. Hence, the TS II heuristic out-performed the other heuristics.

In Table 5, the results for the test problems with 32 locations are given. The TS II heuristic obtained the best solutions for 20 of the 24 problems and 17 are new best solutions. The TS III, TS I, TS Old, and SA obtained the best solutions for 7, 7, 2, and 2, respectively. In addition, the average percent deviation the OFVs of the proposed heuristics are below the OFVs of the SA and TS old heuristics is 3.42%. The average run times of the proposed heuristics, TS old, and SA are 14, 36.22, and 41.93 min, respectively. Hence, the TS II heuristic out-performed the other heuristics.

In summary, the TS II, TS III, TS I, SA, TS Old obtained the best solutions for 79, 54, 42, 35, and 30 of the 96 test problems, respectively. Therefore, the TS II heuristic, which uses dynamic TLSs, frequency-based memory, and an intensification strategy, clearly out-performs all other heuristics. As mentioned previously, many more operations are performed per iteration for the TS II heuristic; therefore, it requires more computation time to perform each iteration. Yet, it is able to out-perform all the other heuristics requiring only one run, using less total run time. Also, the first assignment method is used to construct the initial solutions for the TS II heuristic; however, the results show that the performance of the TS II heuristic does not depend on the quality of its initial solution, unlike the TS I heuristic. In retrospect, it is believed that the strengths of the TS II heuristic come from the following.

(1) Its use of dynamic TLSs, which may reduce cycling.
(2) Its use of frequency-based memory for non-improving moves, which drives the search to other areas of the solution space.

Although the TS III heuristic did not perform as well as the TS II heuristic, it was more competitive than the others. Also, the TS I heuristic performed better than both the TS Old and SA heuristics. More importantly, the TS I and TS III heuristics (as well as the other heuristics) have a few advantages over the TS II heuristic.

(1) They are simple and easy to code.
(2) Require less time and effort to obtain the heuristic parameter settings.

Since the TS II heuristic requires many more heuristic parameter settings, a tremendous amount of effort was required in setting the parameters.

5. Conclusion

In this paper, three TS heuristics were presented to solve the DSAP. The first heuristic (i.e., the TS I heuristic) is a simple TS heuristic which uses static TLSs and multiple initial solutions as its diversification strategy. The second heuristic (i.e., the TS II heuristic) uses dynamic TLSs, frequency-based memory, and an intensification strategy, and the third heuristic (i.e., the TS III heuristic) is a probabilistic TS heuristic. In other words, the admissible move is randomly selected from a list of the top M moves, unlike in TS I where the best admissible move is selected. The proposed heuristics performed well on the data set used by McKendall et al. [1] and McKendall and Jaramillo [2]. More importantly, each of the proposed heuristics out-performed the heuristics available in the literature for the DSAP. The following recommendations are given for future research:

- Consider adding preparation cost data to the data set available in the literature and compare results obtained from the proposed heuristics and the old TS heuristic.
- Develop heuristics for the DSAP which require less heuristic parameters than the proposed TS II heuristic but perform equally as well or better.

References

[1] McKendall AR, Noble JS, Klein CM. Simulated annealing heuristics for managing resources during planned outages at electric power plants. Computers & Operations Research 2005;32(1):107–25.
[2] McKendall AR, Jaramillo JR. A tabu search heuristic for the dynamic space allocation problem. Computers & Operations Research 2006;33(3):768–89.
[3] Patterson JH, Huber WD. A horizon-varying zero–one approach to project scheduling. Management Science 1974;20(6):990–8.
[4] Koopmans T, Beckmann M. Assignment problems and the location of economic activities. Econometrica 1957;25:53–76.
[5] Rosenblatt MJ. The dynamics of plant layout. Management Science 1986;32(1):76–86.
[6] Lee C-G, Ma Z. The generalized quadratic assignment problem. Working paper, Department of Mechanical and Industrial Engineering, University of Toronto; 2005.

[7] Gharbi A, Pellerin R, Villeneuve L. A new constraint-based approach for overhaul project scheduling with work space constraints. International Journal of Industrial Engineering 1999;6(2):123–31.
[8] Zouein PP, Tommelein ID. Dynamic layout planning using a hybrid incremental solution method. Journal of Construction Engineering and Management 1999;125(6):400–8.
[9] Glover F. Future paths for integer programming and links to artificial intelligence. Computers & Operations Research 1986;1(3):533–49.
[10] Skorin-Kapov J. Tabu search applied to the quadratic assignment problem. ORSA Journal on Computing 1990;2:33–45.
[11] Taillard E. Robust taboo search for the quadratic assignment problem. Parallel Computing 1991;17:443–55.
[12] Battiti R, Tecchiolli G. The reactive tabu search. ORSA Journal on Computing 1994;6:126–40.
[13] Skorin-Kapov J. Extensions of a tabu search adaptation to the quadratic assignment problem. Computers & Operations Research 1994;21(8): 855–65.
[14] Chiang W-C, Kouvelis P. An improved tabu search heuristic for solving facility layout design problems. International Journal of Production Research 1996;34(9):2565–85.
[15] Chiang W-C, Chiang C. Intelligent local search strategies for solving facility layout problems with the quadratic assignment problem formulation. European Journal of Operational Research 1998;106:457–88.
[16] Kaku BK, Mazzola JB. A tabu-search heuristic for the dynamic plant layout problem. INFORMS Journal on Computing 1997;9(4):374–84.
[17] Osman IH. Heuristics for the generalized assignment problem: simulated annealing and tabu search approaches. OR Spektrum 1995;17: 211–25.
[18] Laguna M, Kelly JP, Gonzalez-Velarde JL, Glover F. Tabu search for the multilevel generalized assignment problem. European Journal of Operational Research 1995;82:176–89.
[19] Diaz JA, Fernandez E. A tabu search heuristic for the generalized assignment problem. European Journal of Operational Research 2001;132: 22–38.
[20] Higgins AJ. A dynamic tabu search for large-scale generalized assignment problems. Computers & Operations Research 2001;28:1039–48.
[21] Lim A, Rodrigues B, Xiao F, Zhu Y. Crane scheduling with spatial constraints. Naval Research Logistics 2004;51:386–406.