2005

Simulated Annealing Heuristics for Managing Resources during Planned Outages at Electric Power Plants

Alan McKendall

James Noble

Cerry Klein

Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 32 (2005) 107–125
computers & operations research
www.elsevier.com/locate/dsw

Simulated annealing heuristics for managing resources during planned outages at electric power plants

Alan R. McKendall Jr[a,*], James S. Noble[b], Cerry M. Klein[b]

[a] *Department of Industrial & Management Systems Engineering, 325A Mineral Resources Building, PO Box 6070, West Virginia University, Morgantown, WV 26506, USA*
[b] *Department of Industrial & Manufacturing Systems Engineering, E3437 Engineering Building East, University of Missouri-Columbia, Columbia, MO 65211, USA*

Received 17 December 2002

Abstract

This paper presents a mathematical model and simulated annealing heuristics for assigning activities to workspaces and resources (e.g., equipment, parts, and toolboxes) to work/storage spaces during planned outages at electric power plants. These assignments are made such that the distance resources (toolboxes) travel throughout the duration of the outage is minimized. This problem is defined as the dynamic space allocation problem. To test the performance of the proposed techniques, a data set is generated and used in the analysis. The results show that the simulated annealing heuristics perform well with respect to solution quality and computational time.

Keywords: Dynamic space allocation problem; Simulated annealing; Integer programming model; Outage planning; Electric power plants

1. Introduction

This research is motivated by the challenge to reduce congestion and work crew interference inside a reactor containment building during planned outages at a nuclear power plant. During planned outages, activities (laydown, preventative maintenance, surveillance, etc.) are scheduled with respect to constraints on resources, space, and logic relationships between activities such that outage duration is minimized. Once the schedule of the outage activities are obtained, laydown managers and building

* Corresponding author. Fax: +1-304-293-4970.
E-mail address: armckendall@mail.wvu.edu (A.R. McKendall).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/S0305-0548(03)00206-5

coordinators assign activities to workspaces and idle resources (e.g., toolboxes) to storage spaces when they are not used to perform an activity. These assignments are made based on the knowledge and past experiences of the plant managers and coordinators. Since these assignments are often determined based on limited information and without the aid of an algorithm or efficient technique, poor assignments are made causing high congestion and work crew interference. Therefore, this paper focuses on assigning activities to workspaces and resources to work/storage spaces such that the total distance the resources travel throughout the duration of the outage (transportation cost) is minimized. This problem is defined as the dynamic space allocation problem (DSAP). If the total distance resources travel is reduced, then travel time is reduced. According to Askin [1, p. 204], reduced material movement translates into reductions in required aisle space, higher productivity and safety, reduced storage space and utility requirements, simplified material control and scheduling, and less overall congestion. Hence, when minimizing the distance resources travel, other objectives are achieved simultaneously. More importantly, outage cost is reduced.

In the building construction literature, Zouein [2] defined the layout planning problem as the task of assigning site space to resources such that they can be accessible and functional during construction. In addition, they defined dynamic layout planning as creating layouts that change over time as construction progresses. The authors used a construction method to heuristically assign site space to resources with respect to minimizing transportation and relocation costs subject to two-dimensional geometric constraints. Transportation costs are based on the travel distance of a resource from storage to the location where it is used, and relocation costs are based on the travel distance of a resource from one storage space to another storage space. The authors stated that their results are desirable when solving a problem for which no closed-form mathematical solution exists. Other papers in this area that addresses the dynamic layout planning problem are Tommelein [3] and Lin [4].

In light of the literature review, there are no known papers that address the DSAP defined in this paper. Although the dynamic layout planning problem is similar to the DSAP, these problems are different for the following reasons.

- The layout configuration for the DSAP remains the same throughout the planning horizon (i.e., the workspaces and the storage locations are the same for each period); however, it changes in the dynamic layout planning problem (e.g., constructing the interior walls of a building restricts the availability of space).
- In the DSAP, the resources are renewable. In contrast, the resources are consumable (non-renewable) materials in the dynamic layout planning problem.
- In the DSAP, transportation cost is defined as the distance resources travel (i.e., the sum of the travel distances of the resources from/to storage to/from workspaces, from storage locations to other storage locations, and from workspaces to other workspaces) throughout the duration of an outage. In the dynamic layout planning problem, the total distance resources travel is the sum of the transportation (travel distances of resources from storage to workspaces) as well as relocation costs (travel distances of resources from storage to storage locations), and the travel distances of resources from workspaces to workspaces is not considered, since the resources are non-renewable.
- The dynamic layout planning problem only assigns construction resources to locations. In contrast, the DSAP assigns outage activities and their required resources (toolboxes) to workspaces and idle resources (toolboxes) to storage spaces.

In this paper, a mathematical formulation and two simulated annealing heuristics are developed for the DSAP. In Section 2, the definition, assumptions, and mathematical formulation for the DSAP are given, and a small problem instance is solved using the formulation to illustrate the complexity of this problem. Then two simulated annealing heuristics for the DSAP are presented in Section 3. In Section 4, the computational results of the proposed techniques on a set of randomly generated test problems are given. Finally, Section 5 provides conclusions and future research directions.

2. The dynamic space allocation problem model

2.1. Problem definition

When scheduling outage activities (e.g., laydown, preventative maintenance, surveillance) during planned outages at electric power plants (specifically, in a reactor containment building at a nuclear power plant), several resources are needed to perform these activities. Through interviewing personnel and observing planned outages, the authors concluded that the most constraining resources were the pedestal and polar cranes, work crews (laborers, operators, engineers, etc.), toolboxes, and space. A brief description of how these constraining resources are utilized during planned outages follows. First, larger objects (e.g., materials, equipment, toolboxes) are moved into the reactor containment building through the equipment hatch using the pedestal crane. Once these objects are moved into the building, the polar crane is used to move the objects to work/storage spaces. However, workers are used to move smaller objects into the building through the equipment hatch, and they are also used to move these objects to work/storage spaces. Furthermore, work crews (operators, engineers, etc.) are needed to operate the equipment and to perform maintenance and surveillance activities. The most constraining resource, space, is used to store, stage, and move materials, as well as to perform outage activities. After the outage activities are performed, the objects are moved out of the building into a warehouse until the next outage.

There are many software products used to manage projects and to schedule outage activities (i.e., to determine start and finish times for the outage activities). After obtaining the schedule of outage activities, plant managers and coordinators assign outage activities and their resources (i.e., toolboxes) to workspace locations and idle resources (i.e., idle toolboxes) to storage locations. These assignments are made based on the knowledge and past experiences of the plant managers and coordinators. Since assignments are often determined based on limited information and without the aid of a sophisticated algorithm or technique, poor assignments are often made causing high congestion and work crew interference. As a result, this research focuses on developing efficient algorithms for the problem of efficiently assigning outage activities to workspaces and resources to work/storage spaces such that the distance the resources travel throughout the duration of the outage is minimized. In other words, at each time period an activity is started, it is assigned to a workspace as well as the resources required to perform that activity, and the idle resources are assigned to storage spaces. Furthermore, at each time period an activity is completed, its resources are either moved to storage spaces or assigned to other workspaces where activities, which require those resources, are beginning. This problem is defined as the DSAP. Below, the assumptions, notation and formulation for the DSAP are presented.

2.2. Problem assumptions

The assumptions for the DSAP are as follows:

1. The location of the work/storage spaces (i.e., layout configuration) is known;
2. The distances between locations are known;
3. The resources required to perform each activity are known;
4. The schedule of the outage activities are known a priori and are determined with respect to precedence relationships between activities as well as constraints on resources and workspaces such that outage duration is minimized;
5. Only one activity can be performed in each workspace in a given period, and the workspace assigned to an activity is large enough to perform the activity and store its required resources;
6. Each activity requires only one workspace and at least one resource;
7. Each activity is assigned to only one workspace (e.g., if an activity is performed in multiple periods, the activity is assigned to the same workspace for multiple periods);
8. The capacities of the storage spaces are known;
9. The objective is to minimize transportation costs (i.e., the total distance the resources travel throughout the duration of the outage).

2.3. Mathematical notation

The indices, parameters, and variables for the DSAP formulation are presented below.

Indices

j = activity number such that $j = 1, 2, \ldots, J$ where $J =$ total number of activities;
t = resource number such that $t = 1, 2, \ldots, T$ where $T =$ total number of resources;
p = time period such that $p = 1, 2, \ldots, P$ where P is the total number of periods;
$k, l \in L$ = the set of all spaces or locations (storage/workspaces) such that $L = (1, 2, \ldots, N)$ where $N =$ total number of storage/workspaces;
$w \in W$ = the set of workspaces where $W \subset L$;
$s \in S$ = the set of storage spaces where $S \subset L$ and $W \cup S = L$;
R_j = set of resources required to perform activity j;
I_p = set of idle resources in period p;
A_p = set of activities performed in period p;

Parameters

d_{kl} = distance between locations k and l;
C_s = maximum number of resources allowed in storage space s;
NR_j = number of resources required to perform activity j;

Decision variables

x_{ptk} = 1 if resource t is assigned to location k at time period p,
= 0 otherwise; and
y_{jw} = 1 if activity j is performed in workspace w,
= 0 otherwise;

2.4. Mathematical formulation

$$\text{Minimize} \quad \sum_{t=1}^{T}\sum_{k=1}^{N}\sum_{\substack{l=1\\ l\neq k}}^{N}\sum_{p=1}^{P-1} d_{kl}x_{ptk}x_{p+1,tl} \tag{1}$$

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

$$\sum_{\forall t\in I_p} x_{pts} \leqslant C_s \quad \forall s \in S,\ \forall p, \tag{3}$$

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

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

$$\sum_{t\in R_j} x_{ptw} = NR_j y_{jw} \quad \forall p,\ \forall j \in A_p,\ \forall w \in W, \tag{6}$$

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

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

The objective function (1) minimizes transportation cost (i.e., total distance the resources travel throughout the duration of the outage). Constraint set (2) ensures that every idle resource in each period is assigned to a storage space. Constraint set (3) guarantees that the storage capacity for each storage space in each time period is not exceeded. 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 restriction on the decision variables are given in (7) and (8).

2.5. Small problem instance

The mathematical formulation for the DSAP will be used to solve a small problem instance. The data for the problem instance is given in Table 1. For instance, activity 3 is performed in periods 1 and 2 and requires resources 2, 7, and 8. Also, resources 1, 4–6 are idle in period 1. The layout of the facility is given in Fig. 1 such that there are three workspaces (locations 1–3) and three storage spaces (locations 4–6). The rectilinear distance measure is used to determine the distances between the locations. For instance, the distances between locations 1 and 2 as well as 1 and 6 are 1 and 3 distance units, respectively. Furthermore, the maximum capacity of each storage space is three resources.

Table 1
Data for the small DSAP instance

Period	Activity	Required resources	Idle resources
1	1	9	1, 4, 5, 6
	2	3	
	3	2, 7, 8	
2	2	3	1, 5
	3	2, 7, 8	
	4	4, 6, 9	
3	5	1, 2, 9	4, 6, 8
	6	5	
	7	3, 7	
4	6	5	1, 2, 4, 6,
	7	3, 7	8, 9
5	7	3, 7	2, 4, 6, 8
	8	1, 5, 9	

Location 1 (Workspace 1)	Location 2 (Workspace 2)	Location 3 (Workspace 3)
Location 4 (Storage space 1)	Location 5 (Storage space 2)	Location 6 (Storage space 3)

Fig. 1. Layout configuration.

Before using the DSAP formulation to solve the small problem instance, the nonlinear (quadratic) objective function needs to be linearized so that a branch and bound technique can be used to solve this problem. The standard linear programming transformation is utilized to linearize this term. Therefore, the linearized objective function,

$$\sum_{t=1}^{T}\sum_{k=1}^{N}\sum_{\substack{l=1\\ l\neq k}}^{N}\sum_{p=1}^{P-1} d_{kl} w_{p+1,\,ptkl} \tag{1'}$$

is substituted for objective function (1) where $w_{p+1,\,ptkl}$ is a zero/one decision variable, and the constraints

$$x_{ptk} + x_{p+1,tl} - 1 \leqslant w_{p+1,\,ptkl} \quad \forall p,\ \forall t,\ \forall k, l \in L(k \neq l), \tag{9}$$

$$x_{ptk} + x_{p+1,tl} \geqslant 2w_{p+1,\,ptkl} \quad \forall p,\ \forall t,\ \forall k, l \in L(k \neq l), \tag{10}$$

$$w_{p+1,\,ptkl} = 0 \text{ or } 1 \quad \forall p,\ \forall t,\ \forall k, l \in L(k \neq l), \tag{11}$$

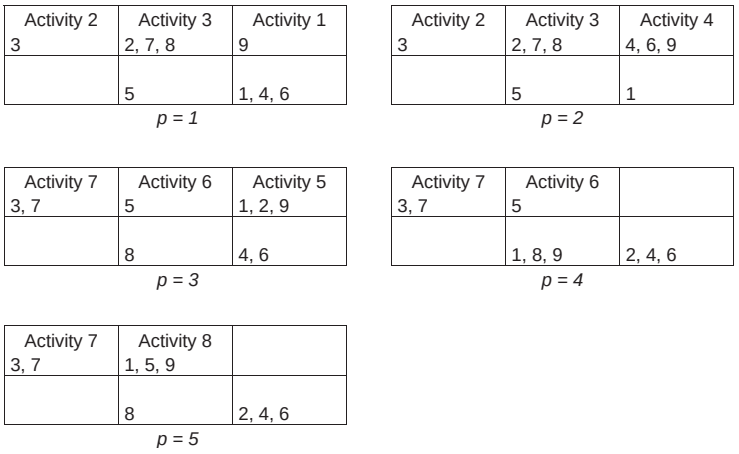

Fig. 2. Optimal solution to the small problem instance.

are added to the formulation. Now the formulation for the DSAP is a zero/one linear integer program and can be solved using a branch and bound technique.

The DSAP formulation is solved optimally using a branch and bound algorithm (CPLEX solver, version 6.0). The optimal solution is given in Fig. 2 and has a cost of 16 distance units. In period 1, activities 2, 3, and 1 as well as their required resources are assigned to workspaces 1, 2, and 3, respectively. Also, the idle resources 1, 4, and 6 are assigned to storage space 3, and the idle resource 5 is assigned to storage space 2. This is the initial assignment of resources to locations; therefore, the cost of this assignment is zero. However, the cost of the assignment of resources to locations is 2 distance units, in period 2, since resources 4 and 6 travel one distance unit from storage space 3 in period 1 to workspace 3 in period 2. In other words, the distance the resources travel between periods 1 and 2 is 2 distance units. In addition, the cost of the assignments in periods 3, 4, and 5 are 7, 5, and 2 distance units, respectively. Therefore, the total distance the resources travel is 16 distance units.

The DSAP instance above required only a few seconds of computational time. However, when several DSAP instances with 10 locations and 10 periods were considered, low quality, though feasible, solutions were obtained for some of the problems, yet for others a feasible solution could not be obtained, after 48 h of computational time on a Pentium IV 1.8 GHz PC. As with any combinatorial optimization problem, as the problem size increases, the computational time increases exponentially.

The computational complexity of the DSAP can be examined by considering the complexity of its sub-problems. The DSAP consists of two problems: the workspace allocation problem (WAP)—the problem of assigning activities to workspaces (i.e., assign the required resources to perform the activities to workspaces); and the storage space allocation problem (SAP)—the problem of assigning idle resources to storage spaces. Assignments are made such that the total distance resources travel during the planning horizon is minimized. The WAP is a generalization of the quadratic assignment problem, since there are multiple periods. The formulation for this problem is given in Section 2.4. More specifically, the WAP formulation consists of objective function (1) as well as constraint sets

(4)–(8), where the locations k and l need to be restricted to the set of workspaces, W, in objective function (1) and constraint set (7). For the DSAP instance given above, the number of possible WAP solutions in the first period is 3! (i.e., the number of ways activities 1–3 can be assigned to the three workspaces is 3 factorial). Since activities 2 and 3 are assigned to workspaces in the first period, the number of possible WAP solutions in the second period is one. That is, in period 2, activities 2 and 3 are assigned to the same workspaces as in period 1, and activity four is assigned to the same workspace as activity 1 in period 1. In periods 3, 4, and 5, the numbers of possible WAP solutions are 6, 1, and 2, respectively. As a result, the number of possible WAP solutions for the DSAP instance is 72, which is the product of the number of possible solutions in each period. In contrast, the SAP is a set partitioning problem. In other words, the set of idle resources in each period is partitioned into sets of resources assigned to specific storage spaces. The number of resources assigned to each set is based on the capacities of the storage spaces. The SAP formulation consist of objective function (1) as well as constraint sets (2), (3), and (7), where the locations k and l need to be restricted to the set of storage spaces, S, in objective function (1) and constraint set (7). For the DSAP example given above, the number of possible SAP solutions in the first period is 78 (i.e., the number of possible ways the idle resources 1,4–6 can be assigned to the three storage spaces is 78). In periods 2, 3, 4, and 5, the numbers of possible SAP solutions are 9, 27, 510, and 78, respectively. Thus, the number of possible SAP solutions is $753,990,120$, which is the product of the number of possible solutions in each period. Obviously the solution space for the SAP is much larger than the solution space for the WAP. Accordingly, the number of possible DSAP solutions is $54,287,288,640$, which is the product of the number of possible WAP solutions and SAP solutions. Therefore, the DSAP consists of two computationally intractable problems. As a result, it is impossible to obtain optimal solutions for large scale DSAPs in reasonable computational time. Hence, two simulated annealing heuristics are developed to quickly obtain high quality solutions for the DSAP.

3. Simulated annealing heuristics for the DSAP

Many large combinatorial optimization problems have been solved successfully by applying simulated annealing (SA) heuristics. Kirkpatrick [5] was the first to use SA to solve combinatorial optimization problems. Wilhelm [6] and Heragu [7] applied SA heuristics for solving the quadratic assignment problem. Chen [8] as well as Adil [9] applied SA heuristics for the cell formation problem. Since SA heuristics perform well for a number of related problems, it is applied to solve the DSAP problem. In this section, two simulated annealing (SA) heuristics are presented for the DSAP.

The SA heuristic starts with an initial solution y^0, call it the current solution (i.e., let $y^c = y^0$), and its transportation cost is obtained, $TC(y^c)$. At the current iteration, a neighboring solution of the current solution is obtained by performing a move (or operation). This solution is denoted as y'. If the cost of the neighboring solution is better than the cost of the current solution, then the neighboring solution is selected as the current solution at the next iteration. More specifically, if $TC(y^c) - TC(y') > 0$, then set $y^c = y'$ for the next iteration. If the cost of the neighboring solution is worse than the cost of the current solution (i.e., $TC(y^c) - TC(y') < 0$), then the neighboring solution is selected as the current solution for the next iteration with respect to an acceptance probability. Otherwise, keep the current solution for the next iteration. At each iteration, the best solution and its

$y^0 =$

[(9),	(3),	(2,7,8),	{1, 4, 5},	{6},	{∅}] Period 1
[(4, 6, 9),	(3),	(2, 7, 8),	{1, 5},	{∅},	{∅}] Period 2
[(1, 2, 9),	(5),	(3, 7),	{4, 6, 8},	{∅},	{∅}] Period 3
[(∅),	(5),	(3, 7),	{1, 2, 4},	{6, 8, 9}	{∅}] Period 4
[(1, 5, 9),	(∅),	(3, 7),	{2, 4, 6},	{8},	{∅}] Period 5

Fig. 3. Initial solution for the DSAP instance.

cost, denoted *Best_sol* and *Best_cost*, are saved and updated, if necessary. The heuristic is repeated for a certain number of iterations or until a stopping criterion is met.

An initial solution for the WAP is generated by assigning the first activity to the first workspace, the second activity to the second workspace, and so on, in the first period. In the second period, if one or more of the activities in the second period is performed in the first period, 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. An initial solution for the SAP is generated by first forming a set of idle resources which is assigned to the first storage space. The number of idle resources in the set is determined by the capacity of the storage space. Then the next set of resources is assigned to the second storage space, and so on. For instance, if there are seven idle resources in period 1 and the capacity of each storage space is 3, then the first three idle resources are assigned to storage space 1. The second three idle resources are assigned to storage space 2, and the last idle resource is assigned to storage space 3. This process is repeated for each period, until all of the idle resources are assigned to storage spaces. Therefore, the initial solution, denoted as y^0, for the DSAP can be represented as multiple sequences of sets of resources. The first half of the sets of resources in each sequence (or period) represent the solution for the WAP, which is the sets of resources, in parentheses ( ), required to perform the activities assigned to the workspaces. The second half of the sets of resources in each sequence represent the solution for the SAP, which is the sets of idle resources, in braces {}, assigned to the storage spaces. For the DSAP instance defined in Table 1 and Fig. 1 with a maximum capacity of three idle resources in each storage space, the initial solution y^0 is generated and given in Fig. 3.

In Fig. 3, the initial solution y^0 for the DSAP instance shows the location of each of the resources in each time period. For instance, in period 1, resource 9 required to perform activity 1 is assigned to workspace 1, and resource 3 required by activity 2 is assigned to workspace 2. The set of resources, 2, 7, and 8, required by activity 3 is assigned to workspace 3. The set of idle resources $\{1, 4, 5\}$ is assigned to storage space 1, and the set of idle resource $\{6\}$ is assigned to storage space 2. Storage space 3 is empty, since no idle resources are assigned to storage space 3. In period 2, activities 2 (3) and 3 (2, 7, 8) are assigned to workspaces 2 and 3, respectively, since they are assigned to these workspaces in period 1. Therefore, activity 4 (i.e., the set of resources 4, 6, and 9) is assigned to the first available workspace. That is, it is assigned to workspace 1. The set of idle resources $\{1, 5\}$ is assigned to storage space 1, and storage spaces 2 and 3 are empty. Hence, each sequence gives the location of the resources in each period. In period 2, the cost of the assignment of resources to locations is 3 distance units, since resource 4 travels 1 distance unit and resource 6 travels 2 distance units. For periods 3, 4, and 5, the cost of the assignments of resources to locations is 11, 6, and 5 distance units, respectively. Therefore, the transportation cost of the initial solution y^0 is 25 distance units, denoted as $TC(y^0) = 25$.

Once an initial solution y^0 is generated and set equal to y^c, a neighboring solution y' is obtained by performing an operation on the current solution y^c. For the WAP, a move is defined as follows:

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

Considering the initial assignment of resources in period 4 for the DSAP solution in Fig. 3, removing activity 6 (and its required resource 5) from workspace 2 and assigning it to workspace 1 is an example of the second type of move. An example of the third type of move is to interchange the locations of activities 6 (5) and 7 (3, 7) in period 4. That is, assign activity 6 (and its required resource 5) to workspace 3, and assign activity 7 (and its required resources 3 and 7) to workspace 2, in both periods 3 and 4. Also, in period 5, remove activity 7 (and its required resources 3 and 7) from workspace 3 and assign it to workspace 2. It is necessary to perform these moves for periods 3, 4, and 5, to maintain feasibility (i.e., to ensure assumption 7 and constraint 4 holds). In other words, if activity 7 is assigned to workspace 2 in period 4, then activity 7 must also be assigned to workspace 2 in periods 3 and 5. Since the three operations are necessary to maintain feasibility, the set of operations is defined as a single move. For the SAP, a move is defined as follows.

1. Interchange the locations (storage spaces) of two resources assigned to different storage spaces in one of the periods.
2. In one of the periods, remove a resource from a location (storage space) and assign it to a different location (storage space in which the capacity has not been met).

In Fig. 3, an example of the first type of move is to interchange the location of idle resources 1 and 6 in storage spaces 1 and 2, respectively, in period 4. More specifically, assign idle resources 1 and 6 to storage spaces 2 and 1, respectively, in period 4. An example of the second type of move is to remove idle resource 1 from storage space 1 and assign it to storage space 3, in period 4. Another example is to remove idle resource 6 from storage space 1 and assign it to storage space 2, in period 5. Since the number of possible moves for the SAP is much larger than the number of possible moves for the WAP, most of the attempted moves during the execution of the SA heuristics are for the SAP.

In some cases, it may be necessary to assign specific outage activities to specific workspaces. For example, if outage activity 7, in the DSAP instance defined above (Table 1 and Fig. 1), is a maintenance activity performed on a large machine and can only be performed in workspace 3 (location 3), then activity 7 is initially assigned specifically to workspace 3. Using the DSAP formulation to obtain the optimal solution for this problem, this would require setting the decision variable $y_{73} = 1$. As a result, the assignment of activities to workspaces and idle resources to storage spaces changes, and the transportation cost increases to 18 distance units (from 16 distance units). Therefore, restricting activities to specific workspaces reduces the complexity of the problem, since the number of feasible solutions is reduced. With respect to the heuristic solution procedure, restricting activities to specific workspaces reduces the number of WAP moves; thus, reducing the complexity of the problem. For example, if activity 2 can only be performed in workspace 2, then

there is only one WAP move in periods 1 and 2 (i.e., interchange the locations of activities 1 and 3 in period 1 and interchange the locations of activities 3 and 4 in period 2). Since assigning specific outage activities to specific workspaces reduces the complexity of the DSAP, it is not considered in this research. However, the SA heuristics presented in this research can easily be modified to consider this case.

The acceptance probability is defined as the probability of accepting a non-improving neighboring solution as the current solution for the next iteration. It is defined as

$$P(\Delta TC) = e^{-\Delta TC/T_c} \text{ and } T_c = T_0\alpha^{r-1} \quad \text{for } r = 1, 2, \ldots, R,$$

where $\Delta TC = TC(y^c) - TC(y')$, T_c represents the current temperature, T_0 is the initial temperature, $r-1$ is the number of temperature reductions, and α is called the cooling ratio and is usually set at 0.90 as in Wilhelm [6] and Heragu [7]. If a randomly generated number x, between 0 and 1, is such that $x < P(\Delta TC)$, then accept the non-improving neighboring solution as the current solution for the next iteration. Otherwise, reject the non-improving neighboring solution, and keep the current solution. At the initial temperature, the SA heuristic has a higher acceptance probability, which allows the acceptance of non-improving solutions. Therefore, this allows the heuristic to explore the solution space without quickly converging to a poor local optimum. After steady state has been reached (after a certain number of iterations), the temperature is reduced. As the temperature reduces, the heuristic has a lower acceptance probability, thus, enabling the SA heuristic to converge to a high quality local optimum.

The SA heuristic for the DSAP, called SA I, is given below.

Step 0: Define the SA parameters: $T_0 =$ initial temperature, $\alpha =$ cooling ratio, $AM =$ attempted number of moves at each temperature, $p =$ probability of performing a SAP move, $Max\_iter =$ maximum number of consecutive iterations without improvement.

Step 1: (a) Initialize the temperature change counter: $r = 1$.

(b) Initialize the number of iterations without improvement counter: $i = 0$.

Step 2: (a) Generate an initial solution y^0 and assign it to the current solution (i.e., set $y^c = y^0$).

(b) Obtain the cost of the current solution, $TC(y^c)$.

(c) Set the following parameters: $Best\_sol = y^c$ and $Best\_cost = TC(y^c)$.

Step 3. If $i \geqslant Max\_iter$, then stop and return *Best_sol* and *Best_cost*. Else, (1) Initialize counter for the number of attempted moves at each temperature: $j = 0$.

(2) Set the current temperature according to the annealing schedule, $T_c = T_0\alpha^{r-1}$.

Step 4: (a) Obtain a neighboring solution y' of y^c by randomly selecting a period t and randomly selecting either a WAP or SAP move in period t. The probability of selecting a SAP move is p.

(b) Update $j = j + 1$.

(c) Calculate the cost of y', $TC(y')$.

(d) Calculate the change in the total cost $\Delta TC = TC(y^c) - TC(y')$.

Step 5: If $\Delta TC > 0$ or ($\Delta TC < 0$ and $x =$ random $(0,1) < P(\Delta TC) = e^{-\Delta TC/T_c}$, then (1) Set $y^c = y'$ and $TC(y^c) = TC(y')$. (2) If $Best\_cost > TC(y^c)$, then $Best\_cost = TC(y^c)$, $Best\_sol = y^c$, and $i = 0$. Else, update $i = i + 1$.

Else, update $i = i + 1$.

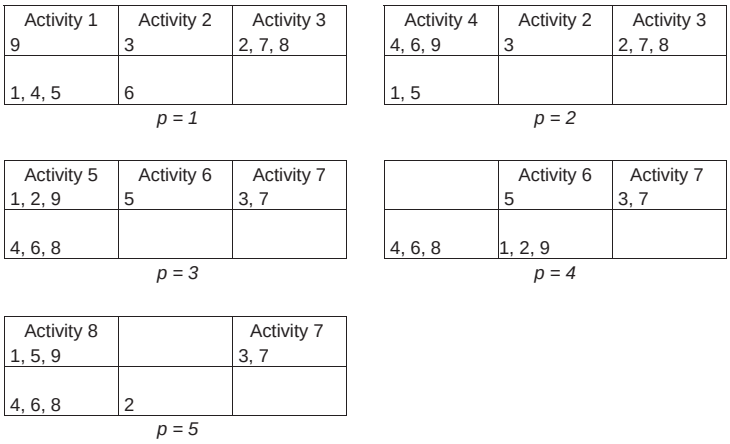

Fig. 4. Initial solution for the SA II heuristic.

Step 6. If $j \geqslant AM$, then update $r = r + 1$, and go to step 3. Else, go to step 4.

The second SA heuristic developed (SA II) performs steps 0–6 of the SA I heuristic. However, the generation of the initial solution (for the SAP) and the SAP moves are slightly different than in the SA I heuristic. The generation of the initial solution for the WAP for the SA II heuristic is the same as for the SA I heuristic. However, the generation of the initial solution for the SAP is different. When generating an initial solution for the SAP for the SA II heuristic, the first set of idle resources in period 1 is assigned to the first storage space. The number of idle resources in the set is determined by the capacity of the first storage space. Then the next set of idle resources is assigned to the second storage space, and so on, in period 1. In period 2, if one or more of the idle resources in period 2 is idle in period 1, assign these idle resources to the same storage spaces as in the first period. The other idle resources are assigned to the first available storage spaces. This process is repeated until all the idle resources in each period are assigned to storage spaces. For example, considering the DSAP instance in Table 1, idle resources 1, 4, and 5 are assigned to storage space 1 (location 4), and idle resource 6 is assigned to storage space 2 (location 5), in period 1. Since idle resources 1 and 5 are assigned to storage space 1 in period 1, they are assigned to storage space 1 in period 2. In periods 3–5, idle resources 4, 6, and 8 are assigned to storage space 1, and idle resources 1, 2, and 9 are assigned to the first available storage space (storage space 2) in period 4. Therefore, idle resource 2 is assigned to storage space 2, in period 5. See Fig. 4 for the initial solution for the SA II heuristic. The transportation cost of the initial solution is 25 distance units, which is the same as the cost of the initial solution for the SA I heuristic. However, this initial solution is much better, since the cost of moving resources from storage spaces to storage spaces in consecutive periods is at a minimum, and the number of SAP moves needed to improve the initial solution is greatly reduced. Also, since most of the cost comes from moving resources from/to workspaces to/from storage spaces in consecutive periods, performing a few WAP and SAP moves can drastically reduce the cost (recall, the number of possible SAP moves are much greater than the number of WAP moves).

In step 4a for the SA II heuristic, an SAP move is defined as follows:

1. Interchange the locations (storage spaces) of two idle resources assigned to different storage spaces in one or more periods.
2. In one or more periods, remove an idle resource from a location (storage space) and assign it to a different location (storage space in which the capacity has not been met).
3. A combination of both (1) and (2).

Considering the initial assignment of idle resources in period 4 in Fig. 4, exchanging the locations of resources 4 and 2 is an example of the third type of SAP move. That is, assign idle resources 2 and 4 to storage spaces 1 and 2, respectively, in time periods 4 and 5 (SAP move 1). Also, remove idle resource 4 from storage space 1, and assign it to storage space 2 (SAP move 2), in time period 3. Since resource 2 is not idle in period 3, the movement of this resource is not considered in the preceding periods. Furthermore, since resource 4 is not idle in period 2, the movement of this resource is not considered in the preceding period (period 1). As a result, when the idle resources 2 and 4 are moved from one location to another in period 4, the movements of these idle resources in periods 3 and 5 are considered so that the cost of moving the idle resources is minimized in consecutive periods. Therefore, the SA II heuristic is equivalent to the SA I heuristic with a look-ahead and look-back strategy.

4. Computational results

A set of test problems was developed for the DSAP to test the performance of the SA I and SA II heuristics. The test problems were generated based on the five following factors.

(1) $N =$ Number of locations (low value is 6, medium values are 12 and 20, and high is 32);
(2) $P =$ schedule changes or number of periods (low value is 10, medium is 15, and high is 20);
(3) $ADV =$ Activity duration variability (low value is 0.75 and high is 1.75);
(4) $ANRA =$ Average number of resources per activity (low value is 1.4 and high is 2.4);
(5) $WU =$ workspace utilization (low value is 50% and high is 90%).

Based on the five factors above, 96 test problems were generated such that 24 test problems were generated for each N. For the 6-location problems, a 2×3 layout is considered such that the first row is the set of workspaces and the second row is the set of storage spaces (same as Fig. 1). The 10-location problems use a 2×5 layout where the first row is the set of workspaces and the second row is the set of storages spaces. The 20-location problems use a 4×5 layout such that rows 1 and 4 are the set of storage spaces and the two middle rows are the set of workspaces. Similarly, the 32-location problems use a 4×8 layout where rows 1 and 4 are storage spaces and rows 2 and 3 are workspaces. The distances between locations were calculated using the rectilinear distance measure. The maximum capacity of each storage space was set to three idle resources, and the minimum and maximum numbers of resources required by an activity are 1 and 3, respectively. Therefore, the total numbers of resources for the 6-, 12-, 20-, and 32-location problems are $9, 15, 30$, and 48, respectively.

In order to determine the parameter settings for the proposed heuristics, both theoretical and experimental techniques were used. The initial temperature T_0 was determined by randomly selecting three problems (at least one easy and one hard) from each set of 24 test problems (i.e., for each N-number of locations). For each problem, an initial solution is generated as discussed earlier, and 5000 iterations of a pairwise exchange heuristic were performed. The change in the objective function value (ΔTC) was recorded at each iteration. Defining the parameter $P(\Delta TC)$ (i.e., the acceptance probability) and using ΔTC with the highest frequency for $\Delta TC > 0$, as well as the formula $P(\Delta TC) = e^{-\Delta TC/T_0}$, T_0 was obtained for each problem, and the maximum T_0 was used for each set of 24 test problems. Since the T_0 obtained is small and the solution space is extremely large, T_0 was multiplied by 300 so that enough temperature reductions can be performed during the implementation of the heuristic. The same initial temperature obtained was used for both heuristics. Since the initial temperatures were multiplied by the constants 300, the formula $P(\Delta TC) = e^{-\Delta TC/(T_C/300)}$ was used to obtain the acceptance probability. Afterwards, the number of attempted moves at each temperature AM was set to different levels by using formulas related to the problem size for a number of the randomly generated test problems. Generally, AM is set to the product of the number of locations (N), the number of periods (P), and the capacity of the storage spaces (i.e., $AM = 3(N)(P)$) performed well. Therefore, this formula was used to obtain AM for both heuristics. Also, based on the initial temperature, the formula $T_c = T_0\alpha^{r-1}$, and performing several experimental runs with different values of α between 0.9 and 0.9999, the cooling ratio α was set to 0.9988. Since the solution space for the SAP is much larger than the solution space for the WAP, the probability p of performing an SAP move was set to 0.6, 0.7, and 0.9, for each test problem. Lastly, the maximum number of consecutive iterations without improvement Max_iter was obtained experimentally for each heuristic. Since the number of moves is reduced using the SA II heuristic, Max_iter is slightly smaller for most of the problems for this heuristic. This information as well as the settings for the initial temperature T_0 and the number of attempted moves at each temperature AM for both heuristics are given in Table 2. Although the SA heuristics performed well with the parameter settings obtained in this paper, it is possible to find better solutions for specific problems using different settings, since the heuristics are stochastic.

The proposed heuristics were programmed using the C++ programming language, and the set of test problems were solved on a Pentium IV 1.8 GHz PC. Each test problem was solved three times for each p (i.e., for $p = 0.6$, 0.7, and 0.9, three runs each were performed; thus, a total of 9 runs were performed) using both the SA I and the SA II heuristics, and the best solution for each of the three runs are given under the "SA I Results" and "SA II Results" columns in Tables 3 and 4 for the 6- and 12-location problems as well as for the 20- and 32-location problems, respectively. Also, the average computation times for the nine runs are presented in minutes under the "Time" columns to the right of the solutions. The bold numbers give the best solution for each test problem, and the asterisks (*) indicate that the solution obtained is the optimal. The optimal solution was found for the first 24 test problems and problem $P27$ using the DSAP formulation presented in this research and the CPLEX Solver (Version 6.0). Problems $P22$ and $P27$ required the most computational time, 1.8 and 57.3 h, respectively. The remaining test problems up to $P48$ ran for 72 h and obtained low quality solutions.

The SA I heuristic produced the optimal solution for 22 of the 25 problems for which the optimal solution is known, and the SA II heuristic produced the optimal solutions for all 25 problems. Out of the 96 test problems, the SA I and SA II heuristics produced the same results for 42 (43.75%)

Table 2
Parameter settings for the SA I and SA II Heuristics

Problem size			SA I Heuristic			SA II Heuristic		
N	P	Max C_s	T_0	AM	Max_iter	T_0	AM	Max_iter
6	10	3	15,000	180	1000	15,000	180	800
	15	3	15,000	270	1000	15,000	270	800
	20	3	15,000	360	1000	15,000	360	800
12	10	3	16,000	360	1200	16,000	360	900
	15	3	16,000	540	1200	16,000	540	900
	20	3	16,000	720	1200	16,000	720	900
20	10	3	17,000	600	1400	17,000	600	1300
	15	3	17,000	900	1400	17,000	900	1300
	20	3	17,000	1200	1400	17,000	1200	1300
32	10	3	18,000	960	1500	18,000	960	1500
	15	3	18,000	1440	1500	18,000	1440	1500
	20	3	18,000	1920	1500	18,000	1920	1500

of the problems, and the SA I heuristic out-performed SA II for only 3 (3.12%) of the problems. However, the SA II heuristic out-performed SA I for 51 (53.13%) of the test problems. The percent deviations of the best solution of SA I above the best solution of SA II are given under the "% Deviation" column. Out of the 51 test problems, 15 of the solutions for SA I were within 2% above the best-found solutions. Also, 27 and 9 of the solutions were within 2–5% and 5–10%, respectively. Therefore, heuristic SA II clearly out-performed the SA I heuristic for this data set.

One of the unique aspects of the proposed heuristics is the parameter p, the probability of performing a SAP move (i.e., idle resource move). The best solution for the SA I heuristic was obtained by $p = 0.90$, 0.70, and 0.60 for 79.8%, 39.6%, and 36.5% of the problems, respectively. This indicates that the SA I heuristic clearly performs better when only 10% of the moves are WAP moves, since the number of SAP moves are much greater than the number of WAP moves. However, the best solution for the SA II heuristic was obtained by $p = 0.90$, 0.70, and 0.60 for 50%, 64.6%, and 61.5% of the problems, respectively. Therefore, the SA II heuristic performs better when 30% of the moves are WAP moves (i.e., 70% of the moves are SAP moves). The reduction of SAP moves from 90% in the SA I heuristic to 70% in the SA II heuristic is justified, since a SAP move in the SA II heuristic is a combination of SAP moves in the SA I heuristic. Therefore, less SAP moves need to be performed.

The problems are listed such that the first, second, third and fourth problems in each set of four problems (separated by bold lines) are categorized as hardest, hard, easy, and easiest problems, respectively. The average number of resources per activity (ANRA) and workspace utilization (WU) determine this categorization. When the ANRA and WU are low, the average number of idle resources available is high, which gives the largest possible solution space. Therefore, this problem is the hardest. When the ANRA is medium and WU is low, the average number of idle resources is relatively high, but not as high as the previous case. Therefore, the problem is defined as hard. When

Table 3
SA I and SA II results for 6- and 12-location problems

Problem size			SA I results				SA II Results				(%) Deviation
N	*T*	Pb #	60%	70%	90%	Time	60%	70%	90%	Time	
6	10	P01	**16***	**16***	**16***	0.17	**16***	**16***	**16***	0.41	0
		P02	26	26	26	0.12	**25***	**25***	26	0.52	−4.00
		P03	**18***	**18***	**18***	0.12	**18***	**18***	**18***	0.42	0
		P04	**25***	**25***	**25***	0.12	**25***	**25***	**25***	0.42	0
		P05	18	17	**16***	0.19	**16***	**16***	**16***	0.66	0
		P06	32	32	**27***	0.19	**27***	**27***	**27***	0.71	0
		P07	**16***	**16***	**16***	0.25	**16***	**16***	**16***	0.73	0
		P08	**31***	**31***	**31***	0.16	**31***	**31***	**31***	0.61	0
	15	P09	27	**25***	27	0.29	**25***	**25***	27	0.66	0
		P10	49	47	49	0.24	**46***	**46***	49	0.89	−2.17
		P11	**32***	**32***	**32***	0.59	**32***	**32***	**32***	1.02	0
		P12	**41***	**41***	**41***	0.25	**41***	**41***	**41***	0.92	0
		P13	33	30	29	0.28	**28***	**28***	29	0.89	−3.57
		P14	**45***	**45***	46	0.23	**45***	**45***	46	0.79	0
		P15	**35***	**35***	36	0.25	**35***	**35***	36	0.72	0
		P16	**49***	**49***	**49***	0.25	**49***	**49***	**49***	0.77	0
	20	P17	**35***	36	**35***	0.54	**35***	**35***	**35***	2.09	0
		P18	**60***	**60***	65	0.58	**60***	**60***	65	1.51	0
		P19	**46***	**46***	47	0.65	**46***	**46***	47	1.67	0
		P20	**60***	**60***	**60***	0.42	**60***	**60***	**60***	1.56	0
		P21	53	53	**46***	0.44	**46***	**46***	**46***	1.58	0
		P22	71	70	**67***	0.52	**67***	**67***	**67***	1.73	0
		P23	**55***	**55***	**55***	0.48	**55***	**55***	**55***	1.57	0
		P24	**74***	**74***	**74***	0.45	**74***	**74***	**74***	1.74	0
12	10	P25	35	35	**34**	1.46	35	35	**34**	3.31	0
		P26	52	49	**47**	1.55	49	49	**47**	3.33	0
		P27	**43***	**43***	**43***	1.32	**43***	**43***	**43***	3.54	0
		P28	**55**	**55**	56	1.21	**55**	**55**	56	3.41	0
		P29	31	32	**29**	1.63	31	31	**29**	3.69	0
		P30	52	52	**50**	1.55	**50**	**50**	**50**	3.60	0
		P31	44	43	43	1.69	**42**	**42**	43	3.78	−2.38
		P32	**69**	70	**69**	1.99	**69**	**69**	**69**	4.32	0
	15	P33	65	65	60	3.54	**59**	60	60	6.42	−1.69
		P34	82	80	**79**	3.88	80	80	**79**	6.57	0
		P35	77	77	76	3.04	74	**73**	76	5.73	−4.11
		P36	95	95	97	2.73	**90**	**90**	97	6.33	−5.56
		P37	60	62	54	4.48	59	60	**54**	10.60	0
		P38	90	**83**	85	4.52	87	87	85	13.00	2.35
		P39	72	74	**71**	3.20	72	72	**71**	10.00	0
		P40	**108**	110	113	3.14	109	109	113	10.22	0.92

Table 3 (*continued*)

Problem size			SA I results				SA II Results				(%) Deviation
N	*T*	Pb #	60%	70%	90%	Time	60%	70%	90%	Time	
	20	P41	94	99	**85**	6.40	90	90	**85**	11.52	0
		P42	125	122	**113**	5.60	117	115	**113**	14.57	0
		P43	113	113	114	5.49	**110**	**110**	114	15.08	−2.73
		P44	142	142	145	4.98	142	**140**	145	12.45	−1.43
		P45	90	84	**74**	7.90	83	84	**74**	16.29	0
		P46	125	**122**	127	7.64	123	124	**122**	19.24	0
		P47	118	118	118	5.94	**116**	117	118	16.18	−1.72
		P48	174	172	175	5.36	**171**	175	**171**	13.64	−0.58

the ANRA and WU are high, the average number of idle resources available is low, which gives the smallest possible solution space. Therefore, this problem is defined as the easiest. A slightly harder problem (defined as easy) is the case where the ANRA is medium and WU is high (i.e., average number of idle resources available is relatively low). There is no indication that any one of the heuristics performs better for the harder or easier problems. However, as the problem size increases, the SA I heuristic does not perform well. This can be explained by the drawbacks of the stochastic SA I heuristic. For example, if a period and an SAP move is randomly chosen in the SA I heuristic such that the move of the idle resource(s) in previous and subsequent periods would yield the optimal solution, the probability of selecting these periods and performing the necessary moves in consecutive iterations are extremely small because of the randomness of the heuristic. However, the SA II heuristic has the capability of performing the necessary combination of SAP moves for the SA I heuristic with a single SAP move. Thus, enabling the SA II heuristic to obtain high quality solutions.

5. Conclusion

In this paper, the problem of assigning outage activities to workspaces and idle resources to storage spaces with respect to minimizing the distance resources travel during planned outages at a nuclear electric power plant was considered. This problem was defined as the dynamic space allocation problem (DSAP) and was modeled mathematically. Due to the complexity of the problem, two simulated annealing heuristics were presented for the DSAP. The first heuristic, called SA I, is a direct application of simulated annealing for the DSAP. The second heuristic, called SA II, uses a look-ahead and look-back strategy when performing SAP moves (i.e., idle resources moves). Although this strategy increases the computational time for larger problems (in most cases), SA II performed much better than SA I on the set of 96 randomly generated test problems.

The techniques presented in this paper can be easily modified to consider the management of resources during the implementation of construction projects (constructing bridges, multi-story buildings, etc.) and during the implementation of maintenance projects at manufacturing plants, to mention a few. Although the proposed heuristics (specifically, SA II) performed well, other solution

Table 4
SA I and SA II results for 20- and 32-location problems

Problem size			SA I results				SA II results				(%) Deviation
N	*T*	Pb #	60%	70%	90%	Time	60%	70%	90%	Time	
20	10	P49	53	53	**49**	5.83	52	52	**49**	24.64	0
		P50	72	74	71	7.79	70	**69**	71	23.50	−2.90
		P51	66	66	62	5.49	59	**57**	62	22.81	−8.77
		P52	101	102	101	5.01	99	**98**	101	22.29	−3.06
		P53	56	58	**52**	8.63	54	54	**52**	23.67	0
		P54	76	78	73	6.91	73	**72**	73	24.87	−1.39
		P55	68	65	69	6.58	**63**	64	69	21.48	−3.17
		P56	101	103	99	6.53	**98**	**98**	99	21.01	−1.02
	15	P57	84	82	80	15.78	**76**	**76**	80	32.27	−5.26
		P58	119	121	114	14.80	**112**	**112**	114	30.97	−1.79
		P59	108	107	115	13.83	**103**	108	115	32.39	−3.88
		P60	169	167	166	11.36	**162**	**162**	166	29.43	−2.47
		P61	98	96	**84**	16.99	89	90	**84**	32.28	0
		P62	140	146	**137**	13.19	138	138	**137**	31.02	0
		P63	137	133	134	16.14	**127**	**127**	134	30.54	−4.72
		P64	199	204	194	15.13	**190**	**190**	194	28.73	−2.11
	20	P65	139	129	122	17.10	121	**120**	122	43.16	−1.67
		P66	180	172	176	24.29	**167**	**167**	176	47.27	−2.99
		P67	166	170	172	15.61	**164**	165	172	45.35	−1.22
		P68	246	250	245	12.92	**244**	246	245	37.51	−0.41
		P69	154	**133**	139	19.36	135	138	139	36.41	1.48
		P70	205	200	202	25.87	**191**	**191**	202	60.30	−4.71
		P71	160	163	153	21.58	**142**	**142**	153	29.65	−7.75
		P72	232	233	227	17.33	227	**221**	**221**	31.49	−2.71
32	10	P73	88	88	86	19.44	**85**	**85**	86	33.66	−1.18
		P74	112	117	116	18.31	**108**	110	116	33.47	−3.70
		P75	121	120	120	17.57	118	**117**	120	30.26	−2.56
		P76	166	**160**	167	18.05	161	**160**	167	24.79	0
		P77	85	83	**79**	14.26	81	81	**79**	23.88	0
		P78	123	121	120	19.41	112	**111**	120	28.41	−8.11
		P79	126	124	119	16.28	**116**	118	119	33.38	−2.59
		P80	190	190	**182**	18.17	187	**182**	**182**	32.11	0
	15	P81	157	154	147	35.22	**140**	**140**	**140**	36.89	−5.00
		P82	208	205	199	41.27	194	194	**183**	36.36	−8.74
		P83	215	215	209	42.06	**200**	202	205	42.18	−4.50
		P84	295	308	310	28.82	289	288	**287**	35.87	−2.79
		P85	151	160	160	34.95	**144**	146	148	44.22	−4.86
		P86	229	227	213	50.01	**209**	210	**209**	54.37	−1.91
		P87	220	216	210	29.43	210	**205**	211	39.45	−2.44
		P88	307	328	311	27.64	**302**	310	310	39.92	−1.66

Table 4 (*continued*)

Problem size			SA I results				SA II results				(%) Deviation
N	*T*	Pb #	60%	70%	90%	Time	60%	70%	90%	Time	
	20	P89	215	228	204	57.71	**195**	**195**	**195**	66.09	−4.62
		P90	286	293	293	66.69	279	**278**	279	83.18	−2.88
		P91	311	311	311	54.61	**293**	**293**	**293**	69.47	−6.14
		P92	405	400	400	39.85	399	399	**395**	65.95	−1.27
		P93	239	238	217	71.32	217	215	**211**	81.67	−2.84
		P94	340	326	319	69.78	302	**298**	301	60.97	−7.05
		P95	357	348	348	57.67	342	**332**	**332**	62.45	−4.82
		P96	487	488	488	47.99	**485**	**485**	**485**	57.11	−0.41

techniques such as construction algorithms (used to construct good initial solutions), tabu search, as well as hybrid techniques (e.g., simulated annealing with tabu search) are areas for further consideration.

Acknowledgements

We would like to thank the referees for their valuable comments, which greatly improved the paper. This research was funded in part by AmerenUE/St. Louis, Missouri and NASA West Virginia EPSCoR Program.

References

[1] Askin RG, Standridge CR. Modeling and analysis of manufacturing systems. New York: Wiley, 1993. p. 204–53.
[2] Zouein PP, Tommelein ID. Dynamic layout planning using a hybrid movement solution method. Journal of Construction Engineering and Management 1999;125(6):400–8.
[3] Tommelein ID, Zouein PP. Interactive dynamic layout planning. Journal of Construction Engineering and Management 1993;119(2):266–87.
[4] Lin KL, Haas CT. An interactive planning environment for critical operations. Journal of Construction Engineering and Management 1996;122(3):212–22.
[5] Kirpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.
[6] Wilhelm MR, Ward TL. Solving quadratic assignment problems by 'simulated annealing'. IIE Transactions 1987;19:107–19.
[7] Heragu SS, Alfa AS. Experimental analysis of simulated annealing based algorithms for the layout problem. European Journal of Operational Research 1992;57:190–202.
[8] Chen WH, Srivastava B. Simulated annealing procedures for forming machine cells in group technology. European Journal of Operational Research 1994;75:100–11.
[9] Adil GK, Rajamani D, Strong D. Assignment allocation and simulated annealing algorithms for cell formation. IIE Transactions 1997;29:53–67.