International Journal of Industrial Engineering, 15(1), 53-61, 2008.

Scheduling Maintenance Activities During Planned Outages At Nuclear Power Plants

Alan R. McKendall, Jr.[1], James S. Noble[2], and Cerry M. Klein[2]

[1]Department of Industrial & Management Systems Engineering
West Virginia University
325A Mineral Resources Building
PO Box 6070
Morgantown, WV 26506, USA
Corresponding author's e-mail: {armckendall@mail.wvu.edu}

[2]Department of Industrial & Manufacturing Systems Engineering
University of Missouri-Columbia
E3437 Engineering Building East
Columbia, MO 65211, USA

In order to maintain high production rates, electric power as well as manufacturing plants shut down so that maintenance activities can be performed on machines/equipment. These planned shut downs (outages) usually occur at least once a year and are more frequent for plants with older machines/equipment. The major costs associated with outages are cost of materials, labor cost, and loss of production cost. Due to high cost incurred from loss of production, the objective is to schedule maintenance activities such that outage duration is minimized. Since the resources required to perform maintenance activities are very limited, the problem of scheduling the maintenance activities is defined as a resource constrained project scheduling problem (RCPSP). In this paper, a solution technique, which consists of a simulated annealing heuristic, is presented for the RCPSP.

Significance: This paper presents a unique resource constrained project scheduling problem encountered during maintenance outages at nuclear power plants. Furthermore, the proposed heuristic performs well for large real-world problems.

Keywords: Resource Constrained Project Scheduling Problem, Simulated Annealing, Meta-heuristic, Outage Planning, Electric Power Plants

(*Received 19 May 2006; Accepted in revised form 7 March 2007*)

1. INTRODUCTION

This research was motivated by the challenge to schedule outage activities at a nuclear power plant such that outage duration is minimized. During planned outages, maintenance activities such as laydown, preventative maintenance, testing and inspecting activities are scheduled. In other words, start and finish times are obtained for each activity such that outage duration is minimized and constraints on resources, space, and precedence relationships between activities are satisfied. This problem is defined as a resource constrained project scheduling problem (RCPSP). Resources such as cranes (jib, pedestal, and polar), work crews (laborers, operators, engineers, etc.), toolboxes, and materials are needed to perform maintenance activities. Also, space is needed to store, stage, and move materials, as well as to perform activities. However, the resources and space required to perform the maintenance activities are limited. In addition, certain activities have to be completed before other activities can start. That is, precedence constraints exist between certain activities. Other important issues considered when scheduling maintenance activities are safety and environmental issues, reducing personnel radiation exposure, and allowing for unanticipated activities. Although minimizing cost is important, minimizing outage duration is either equally or more important, since most of the outage costs depend on outage duration. For example, it was estimated that the replacement of power from either a nuclear or fossil fuel plant can cost over $1,000,000 per day for a 1000-Mwe (megawatts of electric power) plant in addition to cost for outages extending past their due dates. In other words, roughly $1,000,000 per day is used only to purchase and supply power to the power plant customers during planned outages. As a result, when minimizing outage duration, minimizing cost is achieved simultaneously.

During outage planning, over 3,000 maintenance activities may need to be scheduled. As a result, project planning software is used to schedule the maintenance activities such that outage duration is minimized. Project planning software schedules maintenance activities subject to precedence relationships between activities but does not consider the limitations of the resources. More specifically, there are two major drawbacks to using project planning software. First, leveling the resources is an arduous task when scheduling more than 100 activities. For example, project planning software may schedule four activities to start at a specific time period, which may require a total of 8 skilled workers to perform these activities. However, only 5 workers may be available. Therefore, project managers have to check the schedule for these inconsistencies and either increase the number of workers available (i.e., increase labor cost) or increase the duration of the activities which may increase outage duration. Second, project planning software considers workers or workgroups with respect to managing the project budget since labor costs are identifiable; however, it does not consider other limited resources (e.g., equipment, toolboxes, and workspaces) where costs are not easily quantifiable. Since these critical resources are not considered during the scheduling process, maintenance duration is usually underestimated and outages usually extend pass their due dates.

Exact methods for solving the RCPSP are dynamic programming (Carruthers and Battersby, 1966) and branch and bound (e.g., Demeulemeester and Herroelen 1992 and 1997). Since only small-size problems can be solve optimally in reasonable computation time, heuristics are presented for the RCPSP. The heuristic procedures for solving the RCPSP are truncated branch and bound procedures, disjunctive arc based heuristics, local search techniques, and priority rule based scheduling methods. In this paper, a priority rule based scheduling method and a local search technique is used to solve the RCPSP. Nevertheless, the first heuristic approach to solve the RCPSP used priority rules. Kelly (1963) presented two single-pass approaches (each activity is only scheduled once) called the serial and parallel methods, which generates feasible schedules. Most of the algorithms proposed in the literature and used in practice are parallel algorithms because they seem to work better than the serial ones (Alvarez-Valdes and Tamarit, 1989, p. 115). Wiest (1967), Ulusoy and Ozdamar (1989), and Boctor (1990) presented multi-pass versions of the parallel scheduling method. In multi-pass approaches, several priority rules are used to generate several feasible schedules where the "best" one is selected. Alvarez-Valdes and Tamarit (1989, p. 116-120) presented a list of priority rules used to solve the RCPSP. In this paper, the maintenance activities are scheduled using a multi-pass parallel method with five priority rules: late start time (LST), late finish time (LFT), most total successors (MTS), shortest processing time (SPT), and longest processing time (LPT). The first three priority rules are considered because they are known to perform well on the RCPSP. However, the last two priority rules are considered since the activity durations are already available. However, they usually do not perform well for the RCPSP. In this paper, an approach, similar to the two-stage approach presented in Lee and Kim (1996), is presented for the RCPSP. In the first phase, the authors represented the solution as a permutation of numbers in which the elements are ordered based on the priority of the activities. In the second phase, feasible neighborhood solutions are obtained using simulated annealing. From the permutation of numbers (i.e., the initial solution or the solutions generated using simulated annealing), schedules are easily generated using the parallel method. Cho and Kim (1997), Hartmann (1998), and Thomas and Salhi (1998) presented a simulated annealing heuristic, a genetic algorithm, and a tabu search heuristic, respectively, for the RCPSP.

In this paper, a unique RCPSP is defined and solved using a technique which uses a simulated annealing heuristic. In section 2, the problem description is presented. Then the proposed solution technique for the RCPSP is given in section 3 and its major components are discussed and illustrated on a small problem instance. Finally, section 4 provides conclusions and future research directions.

2. PROBLEM DESCRIPTION

As mentioned previously, this research was motivated by the challenge to schedule maintenance activities at a nuclear power plant such that outage duration is minimized. Knowledge of this problem was obtained through red security badge training, observing maintenance operations within the reactor containment building during an outage, and interviewing power plant personnel. More importantly, how maintenance activities are schedule as well as how limited resources are allocated to activities were also observed and analyzed. As a result, the polar crane, toolboxes, skilled workers, and space were defined as the most constraining resources. A brief description of how these constraining resources are utilized during outages is as follows. First, larger objects (e.g., materials, equipment, toolboxes) are move 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/from work/storage spaces. However, workers are used to move smaller objects into the building through the personnel hatch, and they are also used to move these objects to/from work/storage spaces. Furthermore, skilled workers (operators, engineers, etc.) are needed to operate equipment and to perform maintenance activities. In addition, toolboxes are needed to perform specific maintenance 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 only two main levels of work aggregation that are defined. These are activities and work requests or jobs. An activity is a job or a set of jobs needed to be performed during an outage. There are also anticipated and unanticipated activities. Unanticipated activities may develop as a result of performing other activities such as testing and inspecting activities. Some anticipated activities occur every outage while others do not. The anticipated activities that occur every outage are called core activities. Also, activities can be further classified as maintenance activities and storage (laydown) activities. Maintenance activities are activities, such as preventative maintenance, inspecting, and testing, performed on either system components or equipment. However, laydown activities are activities requiring the use of storage space to store resources such as toolboxes and equipment. During outage planning, over 3,000 activities were scheduled with over 5,000 work requests (jobs).

In the literature, the RCPSP is defined as the task of scheduling a set of activities with respect to an objective such that both precedence relationships between activities and constraints on resources are satisfied. In this paper, the consideration of space as a resource makes this RCPSP unique. Space is needed to transport materials, workers, equipment, and other resources within the highly space-constrained reactor containment building. Most importantly, space is needed for a period of time either to store materials, components/equipment (storage activities) or to perform maintenance activities. In Riley and Sanvido (1995), the authors define space needs as the physical, three-dimensional space required to accommodate a resource and is specified by shape and volume and governed by material dimensions, quantities, stacking ability, overlap, and shape. Since modeling space is extremely difficult and has not been fully addressed in the literature, the three dimensional data for space (length, width, and height) is transformed into one dimensional data by representing volumes of space in terms of regions (or grids). The regions of space (or grids) used as either a material or personnel path are not defined as limited resources, since it cannot be allocated to activities. However, space required to store materials and equipment as well as to perform maintenance activities are defined as limited resources. In addition, space on top of certain components can be allocated either to storage activities or to maintenance activities. For instance, after removing the donut supports on the CRDM ductwork, the supports may be placed on top of the containment coolers. Another example is when performing “head work”, stud cleaning may be performed on top of the pressurizer.

Each job may require a number of limited resources such as skilled workers and/or the polar crane. As mentioned previously, the scheduling of activities while satisfying resource constraints is difficult when using project scheduling software. During outage planning, managers put in tremendous amounts of effort to analyze the schedules produced from project scheduling software and to resolve resource conflicts. Some of these conflicts were observed during an outage. For example, there was a conflict between a laydown activity and a preventive maintenance (PM) activity. The laydown activity was moving toolboxes into the reactor containment building using the pedestal crane, and the PM activity was on the jib crane. The problem was that the scaffolding used to perform PM on the jib crane was blocking the space required by the pedestal crane to move boxes into the containment building. Hence, unstaging the scaffolding was required in order to continue the laydown activity, and then restaging the scaffolding later was required to complete PM on the jib crane. Therefore, several hours were lost which could have extended outage duration. As a result, the methods presented in this paper eliminate such conflicts by explicitly considering limited resources such as toolboxes and space when scheduling jobs. It allows outage planners to consider tradeoffs between outage time and outage cost by varying dynamic resource levels (e.g., space and the number of skilled workers) and fixing static resource levels (e.g., cranes and toolboxes), while scheduling jobs.

The RCPSP in this paper can be defined as the problem of scheduling maintenance activities during outages with respect to minimizing outage duration such that precedence relationships between activities and resource constraints are satisfied. The problem assumptions are as follows.

1. Jobs can be performed in only one way (single-mode);
2. Once jobs start, they cannot be interrupted (non-preemption);
3. Each resource assigned to an activity is assigned to the activity for its duration;
4. The availability of space may vary due to high radiation levels;
5. All the toolboxes required to perform the maintenance activities are initially moved into the building using the pedestal crane, and the storage space required to store the toolboxes when not used to perform activities are available.
6. If unanticipated jobs are created, then reschedule all incomplete jobs;

Assumptions (2) and (3) can be relaxed by breaking up jobs into smaller tasks. For example, if 1 hour is required to perform a job which requires the polar crane during the first and last 15 minutes of its duration, the job can be broken-down into 3 tasks. The task durations for the first, second, and third tasks would be 15, 30, and 15 minutes, respectively. Therefore, the polar crane would be required to perform only tasks 1 and 3 which requires the resource for only 30 minutes instead of the entire 60 minutes, and the polar crane can be used to perform other jobs during the other additional 30-minute period. Also, splitting the jobs into multiple tasks allows for interrupting jobs after specific tasks have been performed. In Assumption (4), if a job is performed such that radiation levels increases which would either contaminate workers performing jobs in nearby workspaces or materials stored in nearby storage spaces, then those locations would not be available until they are safe from high radiation levels. Oftentimes, enough storage spaces are not available to store all of the toolboxes that are idle, since space is very limited. Therefore, the planning horizon can be split into multiple periods

such that assumption (5) holds. As a result, it is not necessary to use the RCPSP to schedule the storage activities, since enough storage space is always available. Although the RCPSP ensures that resource availabilities are not exceeded, the allocation of specific resources to specific activities is not considered. However, the problem of assigning maintenance activities to workspaces and idle toolboxes (storage activities) to storage spaces is defined as the dynamic space allocation problem (DSAP), which was presented by McKendall et al. (2005). In other words, after the maintenance activities are scheduled using the RCPSP, the assignment of maintenance activities to workspaces and the assignment of storage activities to storage spaces are determined using the DSAP.

The zero-one integer programming formulation for the RCPSP presented below is an adaptation of the formulations presented by Pritsker et al. (1969) and Patterson and Huber (1974).

$$\text{Minimize} \quad \sum_{t=EFT_n}^{LFT_n} t X_{nt} \qquad \ldots(1)$$

Subject to

$$\sum_{t=EFT_j}^{LFT_j} X_{jt} = 1 \quad j = 1, 2, \ldots, n \qquad \ldots(2)$$

$$\sum_{t=EFT_i}^{LFT_i} t X_{it} \le \sum_{t=EFT_j}^{LFT_j} (t - d_j) X_{jt} \quad j = 1, 2, \ldots, n,\ i \in P_j \qquad \ldots(3)$$

$$\sum_{j=1}^{n} \sum_{q=t}^{t+d_j-1} r_{jk} X_{jq} \le R_{kt} \quad k = 1, 2, \ldots, K,\ t = 1, 2, \ldots, T \qquad \ldots(4)$$

$$X_{jt} = 0 \text{ or } 1 \quad j = 1, 2, \ldots, n,\ t = 1, 2, \ldots, T \qquad \ldots(5)$$

where

P_j = set of all immediate predecessors of maintenance activity j;
EFT_j = earliest finish time of maintenance activity j as computed by a CPM (forward recursion);
LFT_j = latest finish time of maintenance activity j as computed by a CPM (backward recursion);
d_j = duration of maintenance activity j;
r_{jk} = per period demand for resource k by maintenance activity j;
R_{kt} = number of units available for resource k in period t;

$$X_{jt} = \begin{cases} 1 \ if\ activity\ j\ finishes\ at\ the\ end\ of\ period\ t\ (or\ beginning\ of\ period\ t+1) \\ 0 \ otherwise \end{cases}.$$

Objective function (1) minimizes the completion time of the last activity (activity n). In other words, it minimizes outage duration. If two or more maintenance activities finish last, a dummy successor activity is created for these activities with zero duration and zero resources required. In other words, the last activity, activity n, should be unique. Constraint set (2) ensures that each maintenance activity is completed in exactly one period. Constraint set (3) ensures that precedence relationships are satisfied, and constraint set (4) guarantees that the total number of resources used within each period does not exceed the amounts available. Last, the restriction on the decision variables are given in constraint set (5).

The mathematical formulation for the RCPSP defined in this paper will be used to solve a simple outage scheduling problem with 8 maintenance activities and 9 toolboxes. Each activity requires one unit of space, and 3 workspaces are available. Additional data for the problem instance are given in the first four columns of Table 1. For instance, activity 2, which requires toolboxes 1, 5, and 8, cannot start until activity 1 is completed, and it takes 2 periods to complete this activity. However, activity 1 does not require any resources (including space), and its duration is zero. This activity as well as activity 8 may be milestones (e.g., start or end of a project). From this data, the critical path method (CPM) can be used to determine the EFTs and LFTs required to solve the RCPSP model as well as the ESTs, and LSTs. Also, MTS can easily be determined. This information will be needed to solve the problem instance using the solution technique presented below and are given for that purpose.

Table 1. A Simple Outage Scheduling Problem with Additional Data.

Activity	Duration	Immediate Predecessors	Toolboxes	EST	EFT	LST	LFT	MTS
1	0	none	none	1	1	15	15	7

Activity	Duration	Immediate Predecessors	Toolboxes	EST	EFT	LST	LFT	MTS
2	2	1	1, 5, 8	1	3	17	19	3
3	2	1	1, 4, 6	1	3	17	19	3
4	8	1	2, 6, 9	1	9	15	23	2
5	4	2, 3	6, 7	3	7	22	26	1
6	7	2, 3	1, 2, 8	3	10	19	26	1
7	3	4	3, 5, 9	9	12	23	26	1
8	0	5, 6, 7	none	12	12	26	26	0

Using the RCPSP model and the data given above, as well as the CPLEX solver (version 6.6), the minimum makespan of 18 is obtained. More specifically, activities 2, 3, 4, 5, 6, and 7 are completed at the beginning of periods 3, 11, 9, 15, 18, and 12, respectively. As stated previously, activities 1 and 8 are milestones which represent the start and end of the project, respectively. Also, the optimal solution is given in Figure 1, which shows the start and finish times of each activity.

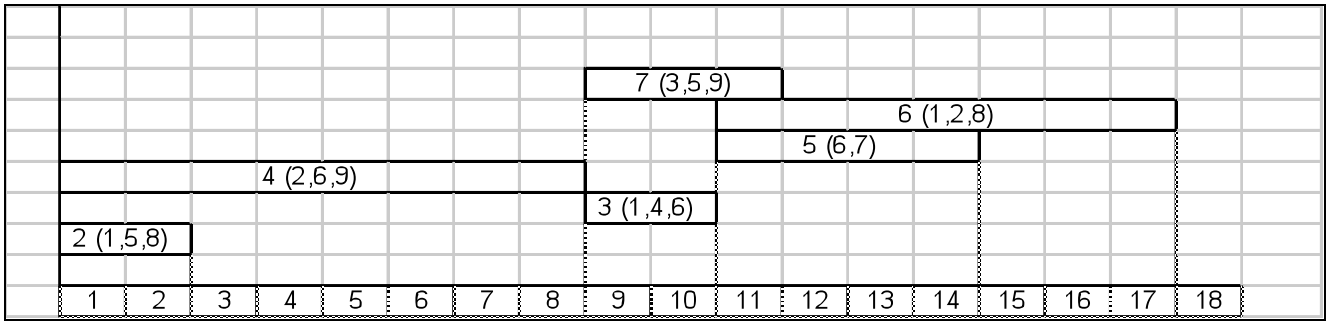

Figure 1. Optimal Solution of Simple Outage Scheduling Problem.

3. SOLUTION TECHNIQUE FOR THE RCPSP

The solution technique for the RCPSP presented in this paper is outlined as follows.
1) Generate an initial solution using a priority rule such as LST, LFT, MTS, SPT, or LPT.
2) Use a parallel scheduling method to generate a schedule for the initial solution and to obtain its makespan.
3) Use simulated annealing to improve the initial solution. The costs of the solutions generated using the simulated annealing heuristic are obtained using the parallel scheduling method.
4) Use a steepest descent heuristic to possibly improve the best solution generated by the simulated annealing heuristic.
As a result, the major components of the proposed solution technique for the RCPSP are the solution representation, the parallel scheduling method, as well as the simulated annealing and steepest descent heuristics. These components as well as other components embedded within the major components (e.g., random descent heuristic embedded within the simulated annealing heuristic) are discussed and are illustrated using the simple outage scheduling problem instance presented above.

For the proposed heuristic, the RCPSP solution can be represented as $\pi = \{\pi 1, \pi 2, \ldots, \pi n\}$, a permutation of activities, such that πi is the ith activity scheduled. An initial solution is obtained using one of the five priority rules. For example, consider the simple outage scheduling problem presented above in table 1. If the LST priority rule is used to obtain an initial solution for this problem and the latest start times are 15, 17, 17, 15, 22, 19, 23, and 26, for activities 1 – 8, respectively, then the activities are ranked according to their latest start times. If a tie exists, the activity with the smallest index is selected. For instance, since activities 1 and 4 have the minimum LST of 15, activity 1 is selected since it has the smallest index. As a result, $\pi = \{1, 4, 2, 3, 6, 5, 7, 8\}$ is the initial solution obtained using the LST priority rule. Since activities 1 and 8 are milestones, the initial solution may be defined as $\pi = \{4, 2, 3, 6, 5, 7\}$.

Once an initial solution is obtained, the following parallel scheduling method is used to generate a feasible schedule as well as determine the cost of the initial solution (i.e., makespan or completion time of the last activity).

Step 1: Obtain a solution π and initialize time (i.e., set t = 1).

Step 2: Using the solution π, determine the eligible activity set (EAS) which consists of the activities which are eligible to start (i.e., activities with all predecessor activities completed). Note, keep the activities in the same order as in π.

Step 3: Schedule the activities in the EAS in the order listed such that resources are available for the duration of the activities. When the activities are scheduled, update the level of resources available.

Step 4: If all the activities have been completed then stop. Else, set t = completion time of the next activity, update the level of resources available, and go to step 2.

To illustrate the parallel method, a feasible schedule will be generated for the solution $\pi = \{4, 2, 3, 6, 5, 7\}$ for the simple outage scheduling problem presented in table 1. First, set t = 1. In step 2, EAS = {4, 2, 3}, since their immediate predecessor is activity 1. In step 3, since activity 4 is listed first, activity 4 is scheduled which uses toolboxes 2, 6, and 9. Next, activity 2 is scheduled, since its toolboxes 1, 5, and 8 are available. In addition, the space constraints hold, since each activity requires only one unit of space, and three units of workspace are available. Because toolboxes 1 and 6 are not available and are required to perform activity 3, activity 3 cannot be scheduled at this time. Therefore, set t = 3 in step 4, since activity 2 finishes first, at the beginning of period 3. Also, release toolboxes 1, 5, and 8 as well as one unit of workspace, and go to step 2. In step 2, EAS = {3}. In step 3, because toolbox 6 is not available and is required to perform activity 3, activity 3 cannot be scheduled at this time. As a result, set t = 9 in step 4, since activity 4 finishes next at the beginning of period 9. Also, release toolboxes 2, 6, and 9 as well as one unit of workspace, and go to step 2. In step 2, add activity 7 to set EAS, since activity 7 had not been scheduled and its predecessor activity had been completed. That is, EAS = {3, 7}. Due to the availability of the resources (toolboxes and workspaces) required by activities 3 and 7, activities 3 and 7 are scheduled in step 3. In step 4, set t = 11, the completion time of the next activity, activity 3, and release its resources. Then go to step 2. Update EAS (i.e., set EAS = {6, 5}, since their predecessors had been completed) in step 2. Due to the availabilities of the resources required by activities 6 and 5, both activities are scheduled in step 3. Therefore, the current time t is updated to the completion time of the next activities, activity 7. In other words, set t = 12, release toolboxes 3, 5, and 9 as well as one unit of workspace, and go to step 2. In step 2, the set EAS is empty, since all activities have been scheduled. Therefore in step 4, set t = 15, completion time of activity 5, and release toolboxes 6 and 7 as well as one unit of workspace. Continue until all activities have been completed and all resources have been released. The heuristic terminates at t = 18. As a result, the cost of the solution is 18 (i.e., $TC(\pi) = 18$), and the schedule is obtained. This is the same schedule obtained using the RCPSP model and the CPLEX solver (see figure 1).

Although the initial solution obtained using the LST priority rule yields the optimal schedule using the parallel method, this is not usually the case, especially for large-size problems. Often times for large-size problems, one must iterate through many solutions to obtain good schedules which may or may not be optimal. Therefore, a random descent local search technique is developed to iterate from existing solutions to neighboring solutions. More specifically, when an initial solution π is generated and the cost of the solution is obtained ($TC(\pi)$), a solution in the neighborhood of π is generated, say π', randomly and is compared to π. If π' is better than π (i.e., $TC(\pi') < TC(\pi)$), then the neighboring solution becomes the current solution (i.e., set $\pi = \pi'$). Otherwise, π does not change. This process is repeated until a stopping criterion is met. Since the solution π represents a permutation of activities, a neighboring solution π' may be obtained by randomly exchanging the positions of two activities in the solution π. However, one must be concerned with exchanging the position of activities such that no activity appears before any of its predecessor activities. For example, $\pi = \{4, 2, 5, 3, 6, 7\}$ is not a feasible solution with respect to precedence constraints, since activity 5 appears before its predecessor activity 3. Therefore, the following technique is used to generate a precedence feasible solution π'.

Step 1: Given solution π. Randomly select an activity j in position u (i.e., $\pi u = j$) to be moved to a new position v.

Step 2: Determine the minimum (lj) and maximum (rj) positions j can be moved (i.e., $lj < v < rj$) such that

$$lj = \max\{k \mid \pi k \in Pj\} + 1$$

$$rj = \min\{k \mid \pi k \in Sj\} - 1$$

where Sj = set of successor activities of activity j and k represent position of predecessor or successor activities.

Step 3: Randomly select a new position v between [lj, rj], and move activity j to position v, and shift activities accordingly. The corresponding solution gives π'.

For instance, consider the solution $\pi = \{4, 2, 3, 6, 5, 7\}$ for the simple outage scheduling problem presented in table 1. First, randomly select an activity j, say j = 6. The predecessor activities are activities 2 and 3 (i.e., Pj = {2, 3}), and there are no successor activities, since activity 8 is a milestone. The position of the predecessor activities 2 and 3 are 2 and 3, respectively. Therefore, the maximum position is 3 (i.e., $\max\{k \mid \pi k \in Pj=6\} = 3$). Hence, the lowest position activity 6 can move to is position 4 (i.e., $lj=6 = 4$). Therefore, activity 6 is currently at its lowest position. On the other hand, the highest position activity 6 can move to is position 6 (i.e., $rj=6 = 6$), since activity 6 does not have a successor activity (or successor activity is activity 8 which is a milestone). In step 3, randomly select a new position v in the interval [4, 6], say 5. Then the neighboring solution $\pi' = \{4, 2, 3, 5, 6, 7\}$ is obtained. The total cost of the solution π' is 18 (i.e., $TC(\pi') = 18$) and was obtained from the parallel method. In addition, π' produced the same schedule as π. Once a neighboring solution π' and its cost (i.e., $TC(\pi')$) are obtained, the local search technique continues by comparing the solutions of π and π'. Since $TC(\pi') = TC(\pi) = 18$, π' is not better than π; therefore, π does not change, and this process is repeated until a stopping criterion is met.

For a steepest descent local search technique, all the solutions in the neighborhood of π are obtained and evaluated, and the best solution in the neighborhood (solution with lowest makespan) is defined as π'. If π' is better than π (i.e., $TC(\pi') < TC(\pi)$), then the neighboring solution becomes the current solution (i.e., set $\pi = \pi'$), and the process is repeated until π' becomes a non-improving solution. This occurs when the heuristic converges to the local optimum. When this happens, π does not change, and the heuristic is terminated at the local optimum, since no solution in the neighborhood is better than π. To generate the entire neighborhood of π, the following technique is used.

Step 1: Obtain a solution π, and set $u = 1$.

Step 2: Select activity j in position u (i.e., $\pi u = j$) to be moved to new positions v in the interval [lj, rj].

Step 3: Determine the minimum (lj) and maximum (rj) positions j can be moved such that

$$lj = \max\{k \mid \pi k \in Pj\} + 1$$
$$rj = \min\{k \mid \pi k \in Sj\} - 1$$

Step 4: For a solution in the neighborhood of π, put j in position lj and shift activities accordingly. Also, determine the makespan of the solution generated. For the next solution in the neighborhood, put j in position lj + 1 and generate its makespan, and continue generating solutions and their makespans until j is put in position rj.

Step 5: Set $u = u + 1$. If $u > n$ (number of last position), select π' (neighboring solution with lowest makespan) and return π' (best solution in neighborhood of π). Else, go to step 2 so that the other neighbors are generated.

The local search techniques (random and steepest descent heuristics) presented above accepts only improved solutions. Since only improved solutions are accepted during the search process, they may converge only to a local optimum of the initial solution. This may result in a low-quality solution, especially for large-size problems. Hence, these techniques do not guarantee the global optimum or even a "good" local optimum. However they are used within many meta-heuristics such as simulated annealing, tabu search, and hybrid ant systems which are local search techniques which accept non-improved solutions so that the global optimum may be obtained. Therefore, the local search techniques presented above are used within the solution technique presented below for the RCPSP. In other words, the simulated annealing (SA) heuristic presented below consists of a random descent local search technique. The best solution obtained from the SA heuristic may not be a local optimum; therefore, the steepest descent heuristic above is used to improve it to ensure that the final solution is a high-quality local optimum. This approach was used in Bouleimen and Lecocq (2003). More specifically, after the neighboring solution π' and its cost, $TC(\pi')$, are obtained in the SA heuristic using the random descent and parallel scheduling techniques presented above, the costs of the solutions of π and π' are compared. If the cost of the neighboring solution is less than the cost of the current solution (i.e., $TC(\pi') < TC(\pi)$), then the neighboring solution becomes the current solution (i.e., set $\pi = \pi'$). If the cost of the neighboring solution is worse than the cost of the current solution (i.e., $TC(\pi') > TC(\pi)$), then the neighboring solution becomes the current solution (i.e., set $\pi = \pi'$) with a probability PA. Otherwise, the current solution does not change. The SA heuristic continues to iterate between improved and non-improved solutions, while keep track of the best found solution. When the stopping criterion is reached, the steepest descent heuristic is used to improve the best found solution obtained from the SA heuristic, if the solution is not a local optimum. Details of the proposed heuristic and the components of the SA heuristic such as the probability of acceptance, PA, are discussed below. The simulated annealing (SA) heuristic is the most important component of the proposed technique, since it enables the technique to obtain high-quality solutions quickly. Simulated annealing (SA) is a meta-heuristic used to solve many combinatorial optimization problems. Kirkpatrick (1983) was the first to use SA to solve combinatorial optimization problems. McKendall et al. (2005) applied SA heuristics to the DSAP. Lee and Kim (1996), Cho and Kim (1997), and Bouleimen and Lecocq (2003) applied SA heuristics to the RCPSP. Since SA heuristics perform well for a number of related problems, it is applied to the RCPSP presented here. Next, the proposed heuristic is presented for the RCPSP.

Step 1: Generate a feasible solution π using one of the following priority rules: LST, LFT, MTS, SPT, or LPT.

Step 2: Calculate the total cost of the solution, $TC(\pi)$, using the parallel method, and set best solution equal π (i.e., $\pi best = \pi$). Also, set cost of best solution (i.e., $TC(\pi best) = TC(\pi)$).

Step 3: Initialize parameters and counters: Te = initial/current temperature, Imax = maximum number of temperature reductions, E0 = (number of neighboring solutions generated at initial temperature) initial epoch length (E0 = n where n = number of activities), E = current epoch length (set E = E0), α = cooling parameter (set $\alpha = 0.95$), β = parameter used to increase epoch length, Ii = inner loop (number of exchanges at current temperature) counter (Ii = 0) and Io = outer loop (number of temperature reductions) counter (Io = 0).

Step 4: For current solution π, generate a neighboring solution π' using the random descent strategy above.

Step 5: Evaluate the neighboring solution π' by calculating $\delta = TC(\pi') - TC(\pi)$.

Step 6: a) If $\delta < 0$, then accept the neighboring solution π' as the current solution (i.e., set $\pi = \pi'$) and update the best solution, $\pi best$, and cost, $TC(\pi best)$ if necessary.

b) If $\delta > 0$, generate a random number p between 0 and 1 and calculate $PA = \exp(-\delta/Te)$. If $PA > p$, accept the neighboring solution π' as the current solution (i.e., set $\pi = \pi'$). Otherwise, keep the current solution π.

c) If $\delta = 0$, keep the current solution π.

Step 7: Update inner loop counter Ii (i.e., set Ii = Ii + 1). If the inner loop counter Ii = E, then decrease current temperature by setting Te = αTe, update epoch length E = E(1 + β), reinitialize the inner loop counter (Ii = 0), and update the outer loop counter Io (i.e., set Io = Io + 1). Else, go to step 4.

Step 8: If the outer loop counter Io = Imax (i.e., total number of temperature reductions is Imax), improve the best solution, πbest, by running the steepest descent heuristic using πbest as the initial solution and return the solution. Otherwise, go to step 4.

The above heuristic is repeated for each of the five initial solutions generated by the five priority rules, and the best solution (schedule) generated is selected. First, the temperature Te is determined by using the formula PA = exp(-δ/Te) = 0.25 and δ = TC(π') – TC(π) = 0.10TC(π), where 0.10 and 0.25 were obtained experimentally. In other words, the initial temperature Te = -0.10TC(π)/ln(0.25) and the initial probability of accepting a non-improving solution π', which is 10% worse than π, is 25%. Then, Te is decreased by αTe at every temperature reduction. More specifically, after obtaining E neighboring solutions π', the temperature Te is reduced to αTe. At the initial temperature Te, the number of neighboring solutions generated is the number of activities (i.e., E0 = n). Otherwise, the number of neighboring solutions generated is E(1 + β), as in the SA heuristic presented by Bouleimen and Lecocq (2003). The process is continued until Imax temperature reductions are performed (i.e., Io = Imax). Since the last temperature, Tlast should be small and positive, set Tlast = 10-10. Also, since Tlast = αImaxTe, Imax = [ln(TLast/Te)]/ln(α) can be obtained where Te = initial temperature. However, α, β, and E0 were obtained experimentally. Initially, the probability of accepting non-improved neighboring solutions (PA) is relatively large. However, as the neighboring solution worsens (δ increases), the probability of acceptance decreases. Also, it decreases, as the temperature Te decreases. To ensure that the best solution obtained from the solution technique is a local optimum, a steepest descent heuristic is implemented on the best found solution as in the SA heuristic presented by Bouleimen and Lecocq (2003).

4. CONCLUSIONS

During planned outages at nuclear power plants, maintenance activities are scheduled with respect to precedence relationships and resource constraints. This problem is defined as a RCPSP. Although a mathematical model was presented for this problem, solving large-size RCPSPs optimally is computationally intractable. Therefore in this paper, a solution technique is developed to obtain high-quality solutions quickly for the RCPSP presented in this paper. The proposed technique uses a parallel scheduling method, a simulated annealing heuristic, and a steepest descent local search technique. More importantly, the proposed solution technique performs well for large real-world problems. The following recommendations are given for future research:

1. Consider relaxing assumption (1). That is, modify the proposed technique to consider performing jobs using multiple modes.
2. Consider relaxing assumption (5). In other words, consider modifying the proposed technique for the RCPSP where laydown activities are scheduled, since storage space may be a constraining resource.
3. Develop hybrid techniques (e.g., tabu search and simulated annealing) for the RCPSP presented in this paper.

5. REFERENCES

1. Alvarez-Valdes, A. and Tamarit, J.M. (1989). Heuristic Algorithms for Resource Constrained Project Scheduling: A Review and an Empirical Analysis, in Advances in Project Scheduling, (eds) Slowinski, R. and Weglarz, J. (Amsterdam: Elsevier), 113-134.
2. Boctor, F.F. (1990). Some Efficient Multi-heuristic Procedures for Resource-constrained Project Scheduling. European Journal of Operational Research, 49:3-13.
3. Bouleimen, K. and Lecocq, H. (2003). A New Efficient Simulated Annealing Algorithm for the Resource-constrained Project Scheduling Problem and its Multiple Mode Version. European Journal of Operational Research, 149(2): 268-281.
4. Carruthers, J.A. and Battersby, A. (1966). Advances in Critical Path Methods. Operational Research Quarterly, 17: 359-380.
5. Cho, J.-H. and Kim, Y.-D. (1997). Simulated Annealing Algorithm for Resource Constrained Project Scheduling Problems. Journal of Operational Research Society, 48(7): 736-744.
6. Demeulemeester, E. and Herroelen, W. (1992). A Branch-and-bound Procedure for the Multiple Resource-constrained Project Scheduling Problem. Management Science, 38(12): 1803-1818.
7. Demeulemeester, E.L. and Herroelen, W.S. (1997). Branch-and-bound Procedure for the Generalized Resource-constrained Project Scheduling Problem. Operations Research, 45(2): 201:212.
8. Hartmann, S. (1998). A Compsitive Genetic Algorithm for Resource-constrained Project Scheduling. Naval Research Logistics, 45: 733-750.

9. Kelly, J.E. (1963). The Critical Path Method: Resources Planning and Scheduling, in Industrial Scheduling, (eds) Muth, J.F. and Thompson, G.L. (Englewood Cliffs, NJ: Prentice-Hall), 347-365.
10. Kirpatrick S., Gelatt, C.D., Vecchi, M.P. (1983). Optimization by Simulated Annealing. Science, 220(4598): 671-680.
11. Lee, J.K. and Kim, Y.D. (1996). Search Heuristics for Resource Constrained Project Scheduling. Journal of the Operational Research Society, 47: 678-689.
12. McKendall Jr., A.R., Noble, J.S., and Klein, C.M. (2005). Simulated Annealing Heuristics for Managing Resources during Planned Outages at Electric Power Plants. Computers & Operations Research, 32(1): 107-125.
13. Patterson, J.H. and Huber, W.D. (1974). A Horizon-varying, Zero-one Approach to Project Scheduling. Management Science, 20(6): 990-998.
14. Pritsker, A.A.B, Watters, L.J., and Wolfe, P.M. (1969). Multiproject Scheduling with Limited Resources: A Zero-one Programming Approach. Management Science, 16(1): 93-108.
15. Rahn, F.J., Adamantiades, A.G., Kenton, J.E., and Braun, C. (1984). A Guide to Nuclear Power Technology: A Resource for Decision Making. John Wiley & Sons, Toronto, Canada.
16. Riley, D.R. and Sanvido, V.E. (1995). Patterns of Construction-space use in Multistory Buildings. Journal of Construction Engineering Management, 121(4): 464-473.
17. Thomas, P.R. and Salhi, S. (1998). A Tabu Search Approach for the Resource-constrained Project Scheduling Problem. Journal of Heuristics, 4: 123-139.
18. Ulusoy, G. and Ozdamar, L. (1989). Heuristic Performance and Network/Resource Characteristics in Resource Constrained Project Scheduling. Journal of the Operational Research Society, 40: 1145-1152.
19. Wiest, J.D. (1967). A Heuristic Model for Scheduling Large Projects with Limited Resources. Management Science, 13(6): B359-B377.

BIOGRAPHICAL SKETCH

Alan McKendall is an Associate Professor in the Department of Industrial and Management Systems Engineering at West Virginia University. He received his Ph.D. in Industrial Engineering, MS in Industrial Engineering, and MS in Applied Mathematics from the University of Missouri in Columbia. He received his BS degree in Mathematics from Southern University in New Orleans. His main research interest is in developing efficient algorithms in the areas of Logistics and Scheduling. He has published in journals such as Computers & Operations Research, Information Sciences, International Journal of Industrial Engineering, International Journal of Operational Research, and International Journal of Production Research. He is currently on the Editorial Board of the Open Operational Research Journal.

James S. Noble is an Associate Professor in the Department of Industrial and Manufacturing Systems Engineering at the University of Missouri - Columbia. He received his Ph.D. in Industrial Engineering from Purdue University, MSIE from Purdue University, and BSIE from the University of Oklahoma. His research interests include integrated material flow systems, logistics systems, production planning and control, and production economics. He is currently an Associate Editor for Engineering Design and Automation and on the Editorial Board of International Journal of Industrial and Systems Engineering. He is a licensed professional engineer in Missouri and a senior member of IIE and ASME.

Cerry Klein is a Professor in the Department of Industrial and Manufacturing Systems Engineering at the University of Missouri – Columbia. He received his Ph.D. in Industrial Engineering from Purdue University where he also received his MS in Mathematics. His current research interests lie in logistics, entrepreneurship, service science, data mining, public policy, health care, and optimization. He has authored and co-authored over 170 technical publications. He is currently an area editor for the International Journal of Operations and Quantitative Management. He is a senior member of Institute for Operations Research and Management Science, the Mathematical Programming Society, the Society for Industrial and Applied Mathematics, and the American Society for Engineering Education.