*Proceedings of the 2005 Winter Simulation Conference*
*M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joins, eds.*

# RESOURCE ALLOCATION AND PLANNING FOR PROGRAM MANAGEMENT

Kabeh Vaziri
Linda K. Nozick
Mark A. Turnquist

School of Civil and Environmental Engineering
Hollister Hall
Cornell University
Ithaca, NY 14853, U.S.A.

## ABSTRACT

Much of the project scheduling literature treats task durations as deterministic. In reality, however, task durations are subject to considerable uncertainty and that uncertainty can be influenced by the resources assigned. The purpose of this paper is to provide the means for program managers (who may have responsibility for multiple projects) to optimally allocate resources from common resource pools to individual tasks on several competing projects. Instead of the traditional use of *schedules*, we develop *control policies* in the form of planned resource allocation to tasks that capture the uncertainty associated with task durations and the impact of resource allocation on those durations. We develop a solution procedure for the model and illustrate the ideas in an example.

## 1 INTRODUCTION

Initial planning has a great deal to do with the potential success of any project effort. Whether the project involves delivery of a service, a new piece of software or hardware, renovating a building, etc., a key question is how to allocate resources to tasks efficiently so that the projects can finish on time and on budget. For some projects, if there are delays in their completion beyond their due date, a financial penalty is incurred. Under such circumstances careful planning of resource use over time is crucial.

Task durations are often subject to considerable uncertainty. This uncertainty increases when the project is a unique attempt for which there is little past experience. It is very intuitive that if the amount of resources allocated to a single task increases, the mean and variance of its duration will decrease. For example, in construction projects multiples of the ideal crew size can be used to shorten the duration of tasks.

A program manager must balance the needs of multiple projects. Resources must be allocated among the projects so that they are completed on schedule. In addition to optimal allocation of the currently available resources, the project manager must evaluate potential changes in resource availability that are likely to be valuable. For instance, the program manager might conclude that the availability of some resources are more than what is really needed while there is another resource that is scarce and therefore drives the duration of the project. This model can be used to develop the analysis to identify opportunities to change resource availabilities.

The focus of this paper is the development of a model and a solution procedure for resource allocation across projects when there are significant uncertainties associated with the task durations and the resource needs of individual tasks. This paper makes two significant contributions to the literature on project management. First, it creates a mathematical model that captures the uncertainty associated with task durations and resource requirements. Second, it develops a solution procedure that allocates available resources to tasks. The procedure is sensitive to the relationship between the resources allocated to a task and the probability distribution for the duration of the task. We use simulation to capture the risk associated with the resource assignments to tasks.

The next section presents a review of the related literature. In section three the problem is defined and in section four a mixed-integer non-linear mathematical model is developed to support the definition. Section five presents a solution procedure for the problem that combines optimization and simulation to estimate optimum control policies for planned resource use. In section six, an illustrative example is discussed to show how this technique can be applied. The last section highlights the conclusions.

## 2 PRIOR RELATED WORK

This research draws on literature in three major areas: resource-constrained project scheduling, stochastic PERT analysis, and combined simulation-optimization methods.

The key question in this paper is how to allocate the available resources across competing projects under uncertainty. Burt (1977) was one of the first to consider how allocation of resources to tasks might affect the probability distributions for task duration. He developed a model based on the uniform and symmetric triangular distributions for task duration. In his model, the allocation of additional resources shifts the right end point of the distribution to the left. His model provided a simple mechanism for looking at resource allocations to tasks, where the effects of additional resources have a specific effect on both the mean and variance of task duration. His procedure was limited to looking at parallel sequences of serial tasks, and to allocating a single, non-renewable resource (e.g., overall budget), but it was an important beginning in examining the general problem.

More recently, Gerchak (2000) studied a related problem where allocating more of a single limited resource to an activity can reduce the variability of its duration without affecting its mean duration. His objective was to construct analytic results for allocating a single resource (e.g., budget) to two activities in a sequence so as to minimize the variance in the sum of their durations.

Our solution method is related to algorithms for the resource-constrained project scheduling problem (RCPSP). The RCPSP is a generalization of the static job shop problem and therefore is NP-hard (Blazewicz et al., 1983). Solving this problem has been a theoretical challenge for researchers. Brucker et al.(1999), Hartmann (1999), Hartmann and Kolisch(2000), and Herroelen et al.(1998) give thorough reviews on how RCPSP has been attacked by many different researchers.

Because this problem is NP-hard, exact methods are very expensive and are generally not practical for even moderately sized problems. Therefore substantial research focuses on developing heuristics to solve the RCPSP. Beside exact methods, there are two other main categories of procedures to attack the RCPSP: priority rules (Kolisch, 1996) and meta-heuristic approaches (Bouleimen and Lecocq 2003, Hartmann 1998).

We focus on the use of priority rule scheduling and meta-heuristics. In the meta-heuristic approach, usually an activity list is created and then by changing the order of the tasks in the list a neighboring schedule is identified. Simulated annealing (SA), genetic algorithms (GA), tabu search and greedy search have all been tested by practitioners to solve the RCPSP. Hartmann's (1998) GA and Bouleimen and Lecocq's (2003) SA approach appear to be the best currently available methods.

Kolisch (1996) has explored priority rule scheduling in great detail. A priority rule schedule consists of two parts, a priority rule and a scheduling scheme. The scheduling scheme is constructed in a stage-wise fashion by building on partial schedules. The priority rule estimates, among the eligible tasks, which task should be scheduled at the current time point with the goal of obtaining the shortest project duration.

Golenko-Ginzburg and Gonik (1998) focus on developing a decision-making process to determine, as a project unfolds, which tasks to start when and what resources to assign. When a task finishes, a local optimization is performed to determine which task or set of tasks should be started next, with what resource assignments. They estimate the effectiveness of this decision-making process using simulation. This research is similar to ours in three respects. First both are focused on modeling project activities when the resources assigned affect the probability distribution of the activity duration. Second, both focus on a simulation-based philosophy to create a schedule and to evaluate that schedule. Finally, both incorporate information about what tasks are likely to be on the critical path when making resource assignments and placing activities on the schedule. With that said, there are some important differences. Perhaps, the most important difference is that Golenko-Ginzburg and Gonik (1998) focus on creating a decision-making process to be used as the project unfolds whereas this paper focuses on creating a plan. Golenko-Ginzburg and Gonik (1998) assume that the resources assigned to an activity have a linear impact on the random duration of an activity. This paper is focused on a declining marginal impact on the duration as additional resources are assigned. Golenko-Ginzburg and Gonik (1998) estimate what tasks may or may not be on the critical path prior to doing the resource assignment, whereas this paper bases its estimates on the probability that an activity is on the critical path once a resource assignment is performed.

This paper is built on a formulation proposed by Nozick et al. (2004). They introduce the notion of resource multipliers which determine the amount of a resource allocated to a task. Changing the resource multipliers affects the probability distribution for the duration of the task. In this paper, we have adopted simulation to capture the effect of a particular set of resource assignments to tasks and the resulting uncertainty of the task durations. We develop a solution procedure based on combined use of simulation and optimization to make the planned resource assignments.

## 3 PROBLEM DEFINITION

The program management problem can be stated as follows: There is a set of projects $L$ that compete for resources from common pools. Each of the projects $l \in L$ consists of a set $J_l$ of tasks. Due to technological requirements, there will be precedence relationships among the tasks. To represent this relationship, we use the activity on node (AON) structure for the project network. We assume that the tasks are numbered based on their precedence, i.e., if task *i* is an immediate predecessor of task *j* then $i < j$. $P_{j_l}$ is the set of immediate predecessors of task *j* from the $l^{th}$ project and task *j* cannot be started until all of its predecessors are completed. We consider additional dummy tasks as well: a single source node and a single sink node to identify the beginning and end of each project. Also, two additional dummy tasks will be added to the beginning and end of the collection of the projects to mark the successful start and completion of the entire set of projects. Without loss of generality, we can assume that these dummy nodes have zero duration and no resource requirements.

We focus on renewable resources in this analysis, but extension to include nonrenewable or doubly-constrained resources is straightforward. Assume there is a set *K* of renewable resources. Each resource $k \in K$ has some finite number of units $R_{kt}$ available in a small time slice indexed by *t*. For notational purposes it is convenient to index time by discrete periods, *t*, but we will allow task durations to vary continuously, so there is an implicit assumption that the periods *t* are very short relative to the duration of the project.

Each task *j* of project *l* has a default requirement, $r_{i_l k0}$, of resource *k* in order for its duration distribution to be described by a nominal distribution. The nominal mean duration of task $i_l$ is $\mu_{i_l 0}$ and nominal variance is $\sigma^2_{i_l 0}$.

Assume $M_{i_l k}$ is the resource multiplier of task *i* of project *l* for resource *k*. If $M_{i_l k} = 1$ for all resources, then task *i* has a nominal average duration of $\mu_{i_l 0}$, a nominal variance in duration equal to $\sigma^2_{i_l 0}$ and the resource requirement of task *i* for renewable resource *k* is $r_{i_l k0}$. However, when the resource multipliers for different resources change, the mean and variance of the duration of task $i_l$ and the need for resource type *k* will change accordingly. $\lambda_{i_l k}$ and $\gamma_{i_l k}$ are elasticities of the mean and the variance, respectively of the duration of task $i_l$ with regard to resource *k*. Equations (1), (2) and (3) represent the mathematical relationship among these parameters.

$$\mu_{i_l} = \mu_{i_l 0} \prod_{k \in K} M_{i_l k}^{\lambda_{i_l k}} \quad \forall \; i_l, \, l \tag{1}$$

$$\sigma^2_{i_l} = \sigma^2_{i_l 0} \prod_{k \in K} M_{i_l k}^{\gamma_{i_l k}} \quad \forall \; i_l, \, l \tag{2}$$

$$r_{i_l k} = r_{i_l k 0} M_{i_l k} \quad \forall i_l, l, k \tag{3}$$

If the resource multiplier of task $i_l$ with respect to resource *k*, $M_{i_l k}$, is changed by 1%, the mean and the variance of the duration for that task will change by $\lambda_{i_l k}$% and $\gamma_{i_l k}$% respectively. This is the same notion as elasticity in microeconomics.

To illustrate the effect of different resource allocations on the distribution of task duration consider the following example. Assume a project task requires only one resource and that its nominal duration is described by Normal distribution with a mean of 10 and variance of 9. The elasticities of the mean and variance are both assumed to be -0.5. Further assume that in the nominal case, the task only needs one unit of the resource when active.

Figure 1 shows how increasing the resource multipliers shifts the probability density function of task duration to the left (lower mean) and compresses it (lower variance).

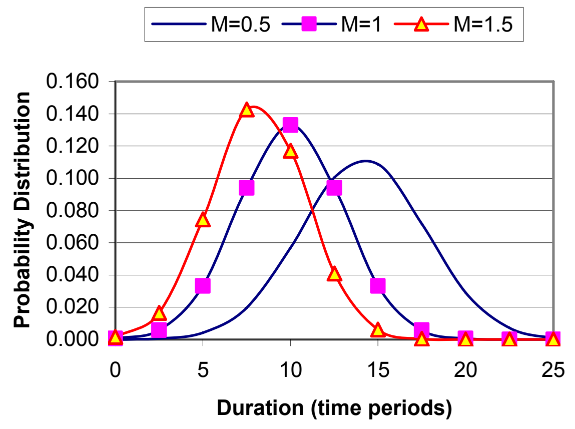

Figure 1: Effect of a Change in the Resource Multiplier on the Probability Distribution of Duration

Figure 2 illustrates that the rate of decrease in the mean duration declines as the resource multiplier increases. This illustrates diminishing marginal returns to increasing resources. In equations (1) and (2) there will be diminishing marginal returns as long as the elasticities are less than 1.0 in absolute value. Figure 2 shows the relationship between expected duration and the resource multiplier for two different values of the elasticity parameter.

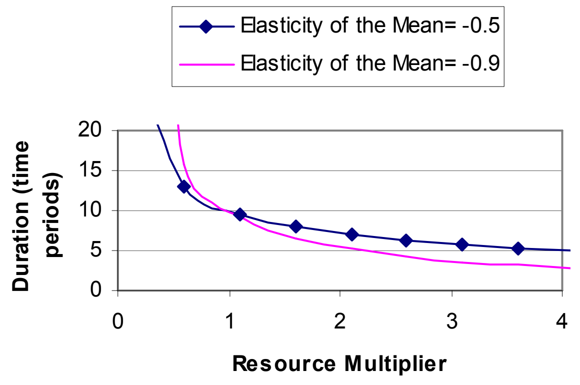

Figure 2: Effect of a Change in the Resource Multiplier on Expected Duration for Two Different Values of Elasticity

The most commonly used objective function in the project management literature is to minimize the makespan of the project subject to honoring precedence and resource requirements. Makespan is defined to be the time span between the start and end of the project. For the purpose of this analysis we focus on minimizing the expected makespan of the collection of the projects subject to precedence constraints and resource limitations. This analysis can be easily generalized to other objectives like minimizing weighted tardiness across the collection of projects.

## 4 MODEL FORMULATION

Equation (4) represents the precedence constraints. It states that a task can only start when all of its predecessors are complete. $P_{j_l}$ is the set of all immediate predecessors of task *j* in project *l*. The duration of task $i_l$ ($d_{i_l}$) is a random variable whose probability density function (pdf) depends on its nominal mean and variance and the resource multiplier for each of the resources. We will denote this pdf as $f_{i_l}(t \mid M_{i_l k},\ \mu_{i_l 0},\ \sigma^2_{i_l 0})$. $S_{i_l t}$ is a binary variable that equals one if task $i_l$ is scheduled to start in period *t* and zero otherwise. The entire project has an allowable time span of *T* time periods.

$$\sum_{\tau=1}^{T} \tau S_{i_l \tau} + d_{i_l} \le \sum_{\tau=1}^{T} \tau S_{j_l \tau} \quad \forall i_l \in P_{j_l}, \forall\ l \tag{4}$$

Assume $N_l$ is the completion task for project *l*. This task is a dummy task and therefore has a duration of zero with no resource requirements. Equation (5) represents the project completion constraint for each project.

$$\sum_{\tau=1}^{T} S_{N_l \tau} = 1 \quad \forall l \tag{5}$$

Equation (6) represents the resource requirement constraints. In these constraints, $r_{i_l k}$ is the amount of resource *k* allocated to task $i_l$ if the resource multiplier is $M_{i_l k}$. $R_{kt}$ represents the amount of resource *k* available at time *t*. If $f_{i_l}(t)$ is the probability density function for the duration of task $i_l$ (suppressing the parameters for notational simplification), then $F_{i_l}(t-\tau)$ is the cumulative probability that this task would complete by time *t*, assuming it started at time $\tau$. Therefore the inner summation in equation (6) represents the probability that task $i_l$ is active in period *t*.

$$\sum_{l \in L} \sum_{i_l} r_{i_l k} \sum_{\tau=1}^{t} S_{i_l \tau} \left[1 - F_{i_l}(t-\tau)\right] \le R_{kt} \quad \forall\ k,\ t \tag{6}$$

In many cases the resource multipliers can only be changed within a certain range. Equation (7) says that the resource multiplier of task $i_l$ for resource *k* should stay within its bounds.

$$M_{i_l k \min} \le M_{i_l k} \le M_{i_l k \max} \qquad \forall i_l, l, k \tag{7}$$

The objective is to minimize the weighted expected makespan of the collection of the projects, where the weights represent the relative importance of each project. If $N_l$ is the dummy completion task for the $l^{th}$ project and $w_l$ is the relative importance of this project ($w_l \ge 0$ and $\sum_{l \in L} w_l = 1$), then the objective function in (8) is minimized subject to equations (1) through (7).

$$\min \sum_{l \in L} \sum_{\tau=1}^{T} w_l \tau S_{N_l \tau} \tag{8}$$

This problem differs from the usual RCPSP because the task durations that appear in equation (4) are random variables and the resource multipliers can be adjusted to change the task duration distributions. However, simulation can be combined with heuristic optimization tools to yield a robust method for allocating the resources effectively.

In the next section we develop a simulation-optimization solution procedure that captures the uncertainty associated with the duration of the tasks when optimizing the resource multipliers.

## 5 SOLUTION PROCEDURE

### 5.1 Overview

The formulation that is presented in the previous section cannot be solved directly because the task durations are uncertain. However, a solution procedure can be constructed that iteratively updates the resource multipliers (an optimization step) and evaluates the implications of specific values for the multipliers (a simulation step). Simulated annealing is the optimization tool used to improve the resource allocation. Based on a given set of resource multipliers, Monte Carlo simulation is used to estimate the distribution of the duration for each project.

The following subsections describe the various elements of the solution procedure. We begin by discussing the generation of the initial solution. The third subsection focuses on the evaluation of a given resource allocation policy, using simulation. The fourth subsection describes how the simulated annealing (SA) algorithm moves to a new solution (set of resource multipliers) in the neighborhood of the existing solution. Subsection five describes how sequential sampling can be used to improve the efficiency of the simulation portion of the algorithm.

## 5.2 Initial Solution Generation

The initial solution for each run is chosen by random sampling. Initial resource multipliers are sampled independently from uniform distributions defined by the stated bounds (equation 7). It is important to note that if the elasticity of the mean and the elasticity of the variance of task $i_l$ for resource $k$ are both zero, then $M_{i_l k}$ should be set to one since there is no incentive for increasing the amount of resource allocated to $i_l$ . When the nominal requirement of task $i_l$ for resource $k$ is zero, no additional resources of that type should be assigned. These special circumstances are implemented by setting the bounds on the resource multipliers appropriately.

Once the initial resource multipliers have been identified, they can be evaluated. In the next subsection, we focus on how to estimate the completion time of the projects once the resource multipliers are known.

## 5.3 Evaluation of Resource Multipliers

Once the resource multipliers are known we use simulation to evaluate the performance resulting from that resource allocation policy. The performance measure considered to determine the quality of the solution is the expected makespan of the collection of the projects. Each replication of the simulation creates a feasible schedule using sampled task durations. The information collected from the simulation is used to update the resource multipliers and generate a neighboring solution to the current one.

Assume that the resource multipliers are known. The mean and the variance of duration for all of the tasks and their resource requirements can be calculated from equations (1) to (3). The scheduling is done using the *parallel scheduling scheme (PSS)*. It is a priority rule based method. A task can be put on the schedule if its precedence and resource requirements are met. To adapt the deterministic notion of the traditional PSS method and the available classic priority rules, the following is done. After the initial dummy task is put on the schedule, the decision set is identified. The decision set is the collection of tasks that are as yet unscheduled and are precedence and resource feasible. The priority value for each of the tasks is calculated and the task that has the best priority value is selected to be put on the schedule.

To evaluate the task priorities, it is assumed that the durations of the tasks already on the schedule are known. For as-yet-unscheduled tasks, the duration is stochastic. Hence, to evaluate priority rules like *latest finish time (LFT)* or *minimum slack (MSLK)*, an embedded simulation has been constructed within the parallel scheduling scheme to accommodate this uncertainty. The simulation allows estimation of the expectations for latest finish time for task *j*, $E[LFT_j]$, or slack relative to the current time $t_n$: $E[LST_j]$-$t_n$. $LST_j$ is the latest start time for task *j*. For *Greatest Rank Positional Weight (GRPW)* and *Most Total Successors (MTS)* there is no need to use embedded simulation because the priority rule definitions are static. Table 1 shows different rules that are used, where $\bar{S}_j$ is the set of successors for task *j*, and *v*(*j*) denotes the priority value computed for each task *j*.

The *PSS* was chosen over the *Serial Scheduling Scheme (SSS)* because the *SSS* considers tasks for placement on the schedule based on precedence constraints only. Hence the order in which tasks are placed on the schedule is not in the order of their start times. As the project, in reality, unfolds this cannot occur. Using the *SSS* would therefore bias our estimates of project duration.

We have also experimented with single pass biased random sampling using the *MTS* priority rule. This method is exactly like parallel scheduling with a deterministic priority rule for all steps except for the actual selection of the task in the decision set to put on the schedule. In the deterministic case, the task in the decision set that has the best value for the priority rule will be chosen to be scheduled. In biased random sampling, each task in the decision set has a chance to be selected, with a probability proportional to its value of the selected rule.

Table 1: Priority Rules Used in Conjunction with PSS

| Priority Rule | Extremum | *v(j)* |
|---|---|---|
| Most Total Successors (MTS) | Max | $\bar{S}_j$ |
| Greatest Rank Positional Weight (GRPW) | Max | $d_j + \sum_{i \in S_j} E[d_i]$ |
| Latest Finish Time (LFT) | Min | $E[LFT_j]$ |
| Minimum Slack (MSLK) | Min | $E[LST_j] - t_n$ |

When a task is selected for placement on the schedule, its duration is sampled from the appropriate probability distribution (given the current values of the resource multipliers), and this duration remains fixed for the remainder of the schedule construction in the current simulation trial. When all tasks have been scheduled, we record the makespan for the collection of projects.

This simulation of the *PSS* with uncertain task durations is replicated *n* times. From these *n* replications, we can estimate the mean makespan and its standard deviation. These estimates are used in the SA algorithm to determine whether to accept or reject the current set of resource multipliers. It is possible to reduce the computational burden of the comparison between sets of resource multipliers significantly by using sequential sampling. This is discussed briefly in Section 5.5.

## 5.4 Updating the Resource Multipliers

Based on the simulation evaluation of the current resource allocation policy, we want to make changes that are likely to reduce the expected makespan of the collection of projects. The key idea is to increase the resources for the tasks that are on the critical path most of the time by releasing resources from those tasks that are not on the critical path. Calculating the percentage of the time that task *i* has been a member of the critical path is quite straightforward.

In this discussion we consider the critical path for the entire collection of projects. This is motivated by the example in the following section. This can be easily extended to consider each individual project critical path if appropriate. Let $CP_i$ be the percent of the time that task *i* has been on the critical path. If task *i* has been on the critical path for at least one of the *n* replications of the scheduling algorithm, then we increase the resource multiplier of that task randomly between zero and $CP_i(1-u_k)(1-r_{i_l k}/R_k)$ where $u_k$ is the average utilization of resource *k* in the current solution and $r_{i_l k}/R_k$ is the proportion of resource *k* consumed by task $i_l$. If the availability of resource *k* varies over the planning horizon then we use the maximum resource availability across all the periods in which the task was active in any replicate.

If task *i* has never been part of the critical path in *n* replications, the resource multiplier of that task for all of the resources will be decreased by a random number between zero and $((E[LFT_i]-E[EST_i])/E[TD])(1-u_k)(1-r_{i_l k}/R_k)$ where EST and LFT represent the earliest start time and latest finish time of task *i*. *E[TD]* is the expected total duration for the projects. All changes in the resource multipliers occur subject to the availability of the resources during the range of time periods that the task is found to be active across all replicates. We also ensure that the new resource multipliers are consistent with the minimum and maximum given in equation (7).

## 5.5 The Overall Simulated Annealing Algorithm

The SA algorithm starts from an initial point (set of resource multipliers) as described in Section 5.2. Using these multipliers, the projects are scheduled using the *PSS-simulation* procedure described in Section 5.3. This provides an evaluation of that resource allocation policy. A neighboring solution is then generated using the update mechanism described in Section 5.4, and this new solution is evaluated by scheduling the projects. If the neighboring solution improves the performance measure, it will replace the current solution. If the performance measure is not improved, there is some probability that the new solution will be accepted anyway. SA stops when a maximum number of solution evaluations have been accomplished.

The overall procedure is summarized in the pseudocode shown in Figure 3.

```
[SA]
Initialization:
Input $r_{i_l k 0}, \mu_{i_l 0}, \sigma^2_{i_l 0}, \lambda_{i_l k}, \gamma_{i_l k}, \alpha$, MaxIteration ;
Generate random initial resource multipliers that agree with equation (7);
Calculate $r_{i_l k}, \mu_{i_l}, \sigma^2_{i_l} \;\; \forall i_l, k, l$ from equations (1) to (3);
denote this as the Current solution.
Evaluate this resource policy using PSS-simulation.
WHILE Iteration<=MaxIteration;
    BEGIN;
    Create a neighboring solution by updating resource multipliers; Denote this as the New solution.
    Evaluate New.
    Δ:= Expected Makespan of New Solution - Expected Makespan of Current Solution
    p:=exp(-Δ/T)
    Rank-sum test:

        IF New is better than Current, THEN accept New to replace Current;
        ELSE IF Current is equal to or better than New, THEN accept New with probability p and keep Current with probability 1-p
        ELSE IF samples are not enough to draw conclusion, sample more; GO TO Rank-sum test;

    Iteration:= Iteration+1;
    T:= αT;
    END;
Perform justification of final solution.
Stop
```

Figure 3. Pseudo Code for the Algorithm

The comparison between the current solution and the new neighboring solution can be made more efficient by using sequential sampling. The new solution is only sampled enough to conclude whether the new or current solutions is better. In the case that the new solution is better than the current, it is immediately accepted. If current solution is equal to or better than new solution, the new solution is still accepted with probability $p = exp(-\Delta/T)$. The parameter $\Delta$ is the difference between the expected total duration of the current solution and the new solution. The value of $T$ is the temperature for that iteration which is controlled by the user defined parameter $\alpha$. To compare the two solutions, we use the Wilcoxon rank-sum test because the underlying distributions of the new and the current solutions' samples are unknown and the sample size for the new and the current solution may differ.

Once we have identified the "best" resource multipliers, we attempt to improve our estimate of the probability distribution of the duration of the projects by using the notion of *justification* based on the forward/backward scheduling proposed by Li and Willis (1992). In this method, the finish times of the tasks in the forward pass, become the priority rules for a backward pass. Valls et al. (2004) showed that justification can reduce the makespan.

## 6 ILLUSTRATIVE EXAMPLE

To illustrate the concepts, consider the following example first described by Yamin and Harmelink (2001) and extended in Nozick et al. (2004). Three identical projects involve constructing a concrete bridge as illustrated in Figure 4. For each bridge there are 8 tasks. The precedence constraints between these tasks are given in Figure 5. The objective function is to minimize the total time required to complete all three bridges.

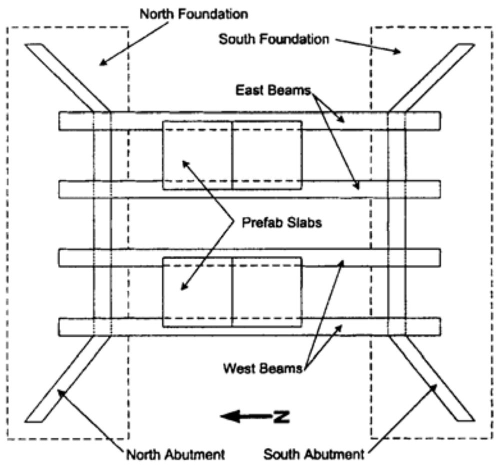

Figure 4: Concrete Bridge

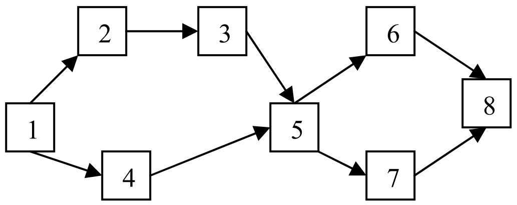

Figure 5: Network Diagram of the Concrete Bridge

Table 2 shows the description, the nominal mean and variance of duration and the nominal resource requirement of each of the tasks. We assume that the distributions for the task durations are Normal. There is only one resource, *crew,* of which 16 are available. We also assume that for all the tasks, the elasticity of the mean duration with respect to the resource *crew* is -0.8 and elasticity of the variance is -0.4. The resource multipliers can be between 0.2 and 2.

For the simulated annealing algorithm, the initial and final probabilities of accepting uphill moves were set to 95% and 1% respectively. The maximum number of iterations is set to 1000. Also if for 200 consecutive iterations no improvement occurs, the algorithm stops.

For sequential sampling, the minimum number of samples is set to 10. For Wilcoxon rank-sum test, it is necessary to have at least 8 observations to be able to use the normal approximation.

Five different priority rules are examined: latest finish time (LFT), minimum slack (MSLK), most total successors (MTS), greatest rank positional weight (GRPW) and random sampling biased towards most total successors (RS).

The code is written in C and the computations are performed on a 2.8 GHz Pentium 4.

Table 3 gives the best result found with 30 independent runs. Each run starts from a set of randomly generated independent initial solutions for the resource multipliers. The initial solutions are the same across algorithms to allow for fair comparison. The first column shows the algorithms that were used. The first five rows use different priority rules for the scheduling (GRPW, LFT, MTS and MSLK as described in Table 1). RSMTS represents biased random sampling based on MTS. The last row shows the results for the case where all of the resource multipliers are fixed at one and biased random sampling is used for scheduling. By fixing the resource multipliers, we obtain a benchmark solution. The second and third columns show the mean and standard deviation of the best solution found in 30 runs. The next two columns show the mean and standard deviation of the total resource use when these algorithms are employed. The last column shows the computational time is seconds.

Table 2: Task Descriptions and Nominal Parameters

| | | Mean Duration | Variance of Duration | Resource Required |
|---|---|---|---|---|
| Task ID | Description | (day) | (day$^2$) | (crew/day) |
| 1 | North Foundation | 14 | 5 | 3 |
| 2 | South Foundation | 14 | 4 | 3 |
| 3 | South Abutment | 16 | 4 | 3 |
| 4 | North Abutment | 16 | 6 | 3 |
| 5 | East Beams | 4 | 1 | 4 |
| 6 | West Beams | 4 | 1 | 2 |
| 7 | Prefab Slabs | 6 | 1 | 4 |
| 8 | Paving | 8 | 2 | 2 |

The results are sorted based on the expected total duration. It can be easily seen that all five algorithms present roughly the same quality of the solution if total duration of the collection of the projects is the only criterion considered. The total resources used (crew-days) in all of the optimized cases are higher than that of the nominal. In terms of the computational time, RSMTS, GRPW and MTS are substantially faster than LFT and MSLK because the priority rules of latest finish time and min slack are dynamic and must be calculated with simulation at each decision point. However, most total successors and greatest rank positional weight just depend on the structure of the network and need to be calculated only once.

Let's focus on the RSMTS solution. Figure 6 and Figure 7 show that for only a 3.4% increase in the expected total crew-days used, a decrease of 13.6% in expected total duration is accomplished. The probability distribution of the total crew use is shifted slightly to the right of the nominal whereas the probability distribution of the total duration is shifted to the left of the nominal.

This decrease in the total duration of the collection of bridge construction projects is accomplished through changes made in the duration of the individual tasks. Tasks 1, 2, 3, 5, 7, 8 are often on the critical path. shows that for tasks that are on the critical path most of the time, the duration has decreased significantly. An example of such task is task 1. However, for cases where the tasks are not on the critical path, e.g., task 4, the duration has increased. In fact the sum of the duration of tasks 2 and 3 have been set so that they are consistent with the duration of task 4. Figure 8 shows the changes in the expected durations of the tasks after optimization.

Table 3: Comparison of the Results

| | Expected Total Duration | Standard Deviation of Total Duration | Expected Total Resource Use | Standard Deviation of Total Resource Use | Computational Time |
|---|---|---|---|---|---|
| Algorithm | (days) | (days) | (crew-days) | (crew-days) | (seconds) |
| RSMTS | 59.69 | 2.64 | 757.85 | 26.97 | 6.90 |
| GRPW | 59.81 | 2.67 | 781.93 | 29.07 | 5.90 |
| LFT | 61.01 | 2.76 | 776.13 | 31.56 | 1118.60 |
| MTS | 61.15 | 3.03 | 793.42 | 34.00 | 6.30 |
| MSLK | 62.22 | 3.03 | 865.49 | 41.26 | 1088.30 |
| Nominal (RS) | 69.10 | 5.05 | 732.90 | 27.09 | NA |

Figure 6 shows that the left-shift of the probability distribution for project duration, is achieved through allocating more resources to the project.

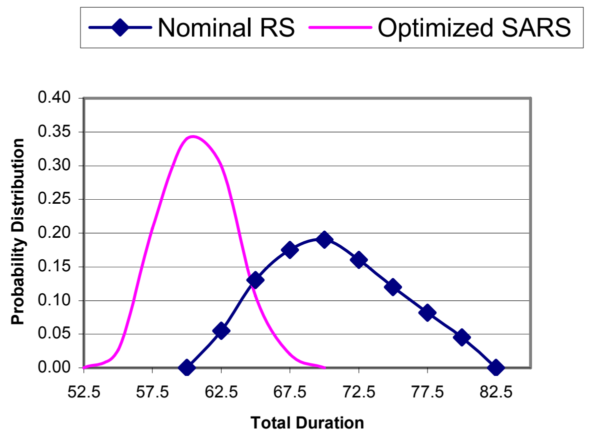

Figure 6: Comparison of the Distributions of Total Duration

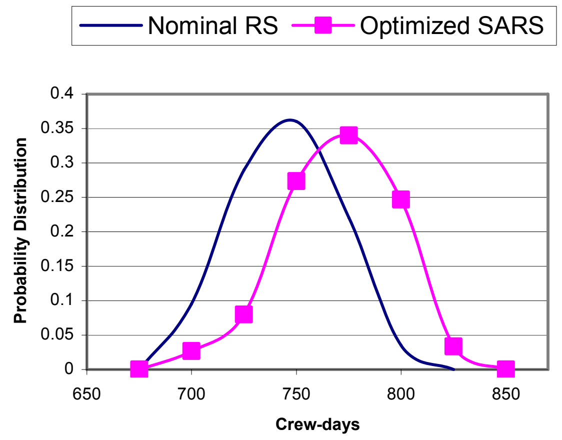

Figure 7: Comparison of the Total Resources Used

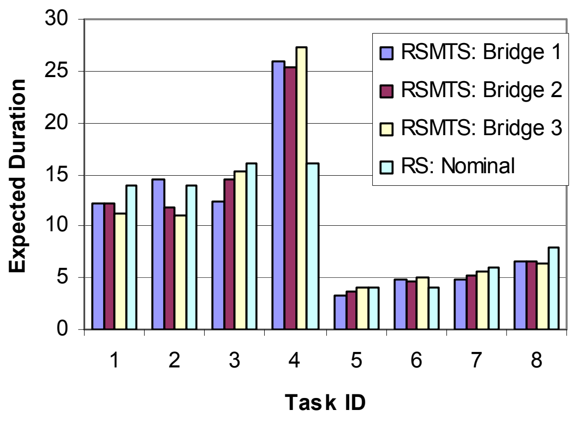

Figure 8: Expected Durations of the Tasks after Optimization

## 7 CONCLUSIONS

This paper has developed a mathematical model to optimize a plan for the assignment of resources to tasks when the assignment of resources to tasks effects the probability distribution for task duration. A solution procedure has been created based on combining simulation and optimization. Different priority rules in the PSS-simulation element of the solution approach produce very similar project durations in an example problem, but the computational times are much longer for the dynamic priority rules (LFT and MSLK). The computational results for an illustrative problem show that reallocating resources among tasks can be used effectively to reduce overall makespan on a set of projects (about 13% in this test case).

## ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0218837. This funding is gratefully acknowledged but it implies no endorsement of the findings.

## REFERENCES

Blazewicz, J., J.K. Lenstra, and A.H.G. Rinnooy Kan. 1983. Scheduling subject to resource constraints: Classification and complexity. *Discrete Applied Mathematics* 5: 11-24.

Bouleimen, K., and H. Lecocq. 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: 268-281.

Brucker, P., A. Drexel, R. Möhring, K. Neumann, E. Pesch. 1999. Resource-constrained project scheduling: Notation, classification, models and methods. *European Journal of Operational Research* 112: 3-41.

Burt, J. M. 1977. Planning and dynamic control of projects under uncertainty. *Management Science* 24: 249-258.

Gerchak, Y. 2000. On the Allocation of Uncertainty-Reduction Effort to Minimize Total Variability. *IIE Transactions,* 32: 403-407.

Golenko-Ginzburg, D., and A. Gonik. 1998. A heuristic for network project scheduling with random activity durations depending on the resource allocation. *Int. J. Production Economics* 55: 149-162.

Hartmann, S. 1998. A competitive genetic algorithm for the resource-constrained project scheduling. *Naval Research Logistics* 456: 733-750.

Hartmann, S. 1999. *Project scheduling under limited resources, models, methods, and applications.* Berlin, Germany: Springer-Verlag.

Hartmann, S. and R. Kolisch. 2000. Experimental evaluation of state-of-the-art heuristics for resource-constrained project scheduling problem. *European Journal of Operational Research* 127: 394-407.

Herroelen, W., B. De Reyck, and E. Demeulemeester. 1998. Resource-constrained project scheduling: A survey of recent developments. *Computers & Operations Research* 25 (4): 279-302.

Kolisch, R. 1996. Efficient priority rules for the resource-constrained project scheduling problem. *Journal of Operations Management* 14: 179-192.

Li, K.Y. and R.J. Willis. 1992. An Iterative Scheduling Technique for Resource-Constrained Project Scheduling. *European Journal of Operational Research*, 56: 370-379.

Nozick, L. K., M. A. Turnquist, and N. Xu. 2004. Managing portfolios of projects under uncertainty. *Annals of Operational Research* 132: 243-256.

Valls, V., F. Ballestin, and S. Quintanilla. 2005. Justification and RCPSP: A technique that pays. *European Journal of Operational Research* 165 (2): 375-386.

Yamin, R. A., and D. J. Harmelink. 2001. Comparison of linear scheduling model (LSM) and critical path method (CPM). *Journal of Construction Engineering and Management* 127: 374-381.

## AUTHOR BIOGRAPHIES

**Kabeh Vaziri** is a doctoral candidate in Transportation Engineering and Logistics in Cornell University. Her research interests include simulation modeling and analysis, network planning and scheduling theory. Her e-mail address is <kv37@cornell.edu>

**Linda K. Nozick** is a professor of civil and environmental engineering at Cornell University. She specializes in systems engineering and applications to transportation, supply chain logistics, infrastructure, and project planning. She was a Presidential Early Career Award winner for her work on uncertainty modeling in transportation systems. Her e-mail address is <lkn3@cornell.edu>

**Mark A. Turnquist** is a professor of civil and environmental engineering at Cornell University. He specializes in large-scale network optimization models for use in transportation, logistics, manufacturing, and infrastructure systems, with a particular emphasis on models that include uncertainty. He is also active in network modeling related to project planning under uncertainty. His e-mail address is <mat14@cornell.edu>