The Optimal Resource Allocation in Stochastic Activity Networks via Continuous Time Markov Chains

Adam J. Rudolph and Salah E. Elmaghraby\(^{(a)}\)
Lenovo, Inc., Morrisville, NC, and
North Carolina State University, Raleigh, NC, USA.

Abstract

The problem we investigate deals with the optimal assignment of a resource to the activities of a stochastic project network. We seek to minimize the expected cost of the project, which we take as the sum of resource utilization costs and lateness cost, if the project is completed after a specified due date. Both costs are functions of the resource allocations to the activities with opposite responses to changes in allocation. We assume that the work content required by the activities follows an exponential distribution. An immediate result of this assumption is that the duration of the activities also follows an exponential distribution based on the degree of resource allocation. We construct a continuous time Markov chain (CTMC) model for the activity network and use the Phase-Type (PH-type) distribution to evaluate the project completion time.

Absence of an analytically tractable means of evaluating the sensitivity of the project cost to variation in the resource allocation to an individual activity led us to develop a derivative-like descent algorithm for the optimization of the expected cost of the project. We approximate the value of the derivative at a particular allocation by evaluating the differential cost of a slightly modified allocation. These quasi-derivatives lead to the selection of an activity to which we optimize the resource allocation. We use Fibonacci search over the interval of permissible allocations to the activity to determine the minimum expected cost. This iterative process of activity selection followed by changing the resource allocation is repeated until the expected cost is not significantly decreased. Finally, through extensive experimentation with a variety of projects of different structure and size, we show that this algorithm yields promising results in terms of both computation time and accuracy.

\(^{(a)}\) Corresponding Author, Department of Ind. and Syst. Eng. and the Graduate Program in Oper. Res., NCSU, Raleigh NC 27695-7906. <elmaghra@eos.ncsu.edu>. 
LIST OF ACRONYMS

AoA: activity-on-arc
AoN: activity-on-node
CI: complexity index
CTMC: continuous time Markov chain
CPM: critical path method
DP: dynamic programming
DTCTP: discrete time-cost trade-off problem
GAN: generalized activity network
GED: generalized Erlang distribution
i.o.u.: interval of uncertainty
LTCTP: linear time cost trade-off problem
NBUE: new better than used in expectation
pdf: probability density function
PERT: program evaluation and review technique
PH-D: Phase-Type distribution
RCPSP: resource-constrained project scheduling problem
r.v.: random variable
SPN: state pace network
TCTP: time-cost trade-off problem
udc: uniformly directed cutset
1 INTRODUCTION AND LITERATURE REVIEW

We are concerned with the issue of optimal resource allocation to the activities of a project from the vantage point of stochastically varying work content of the activities, rather than their duration (which is the standard optic in the literature on the subject). Our analysis proceeds under the following assumptions: (i) the work content of the activity is exponentially distributed, (ii) there is a single critical resource of abundant availability which is continuously divisible, (iii) the intensity of the resource allocation to each activity is bounded from below and from above, (iv) for each activity the relationship between the intensity of resource allocation, activity duration, and the work content is hyperbolic, (v) the cost of resource usage is quadratic in its allocation for the duration of the activity, and finally (vi) the total project cost is composed of two components: the cost of resource usage and the cost of tardiness beyond a specified project completion time. The objective is to determine the resource allocation to the activities that minimizes the total project cost. More will be said about these assumptions in the appropriate sections of the paper, and all the concepts mentioned shall be made precise in our specification of the mathematical model of the problem in § 2. For the moment we would like to motivate the study and review past contributions that relate to it.

Uncertainty is a fact of life in project planning and control. Unfortunately, though accepted by managers, much of the literature pertaining to project resource allocation/scheduling issues ignores this fact, with disastrous consequences as evidenced by the large percentage of projects in real life that are completed late and over-budget (often very late and well over budget, both by a factor of 400% or more!). Keeping this in mind, we seek to develop a model and create an algorithm that will find the optimal way to assign resources to the various stochastic activities of a project to minimize the stated economic objective function.
1.1 LITERATURE REVIEW

This review is categorized into two parts. First we briefly survey the literature within the realm of project scheduling and activity networks in both deterministic and stochastic settings. Second, we discuss several recent research projects that are more closely related to our own. To render this paper self-contained, we summarize the relevant concepts in continuous time Markov chains, Markov PERT Networks, the Phase-type distributions derived from them and the advantages afforded by their application in the Appendix.

1.1.1 PROJECT SCHEDULING AND PERT NETWORKS

Classically, the CPM (“Critical Path Method” [18]) and PERT (“Program Evaluation and Review Technique” [20]) models have been the industry standard for some time mainly due to their simplicity. (The issue of their utility in the actual conduct of projects is an open question that has not been empirically determined, to the best of our knowledge.) In a real sense, however, neither model takes resource constraints into account as they opt to deal more with precedence relationships and time considerations. More recently, much effort has been spent researching the deterministic Resource Constrained Project Scheduling Problem (RCPSP) which we briefly look into before discussing the stochastic case.

The objective of the majority of the RCPSP studies has been to schedule the activities in such a manner as to meet resource availability and precedence constraints in order to optimize an objective function typically related to the project completion time. The RCPSP can be subdivided into four categories: unimodal or multimodal activities and renewable or nonrenewable resources. The unimodal case involves a fixed resource allocation and corresponding duration for each activity, whereas the multimodal case allows activities to be processed in a number of different “modes” requiring different resource levels and completion times. A renewable resource is available throughout the duration of the activity, perhaps in varying capacity, while
a non-renewable resource has a fixed capacity at the start of the activities, which is depleted through usage, never to be replenished. Chapters 7 through 9 of the book by Demeulemeester and Herroelen [5] provide an excellent review of RCPSP problems and related optimization techniques.

An important segment of the studies of the multimodal RCPSP is concerned with the “Time-Cost Trade-off problem” (TCTP), which is closely related to the problem at hand. At its core, TCTP assumes that the duration of an activity is a function of the resources allocated to it for its duration. Further, both the TCTP and our study rely on the concept of “work content”. Normally, greater resource allocation will incur a greater expense; thus we have a trade-off between the time to complete an activity and its cost (a function of the assigned resource). When the task duration is a linear function of the allocated resource we have the “linear time cost trade-off problem” (LTCTP) which was treated by Fulkerson [14] with the objective of minimizing the project cost subject to completion before a due date. The nonlinear case of the TCTP was discussed by Elmaghraby & Salem [7],[8] and Elmaghraby [11], [10]. Deckro et al [4] deviate from the traditional approach to the problem, which would involve piecewise linearization of the nonlinear functions, and suggest a quadratic program and its solution using commercially available software. The authors also investigate some extensions to the model including a goal-programming formulation.

If the resource allocation to an activity is limited to distinct discrete values we have the “Discrete Time-Cost Trade-off problem (DTCTP)”. Hindelang and Muth [17] developed a dynamic programming (DP) algorithm to solve this problem in pseudo-polynomial time when the corresponding activity-on-arc (AoA) project graph is series/parallel. In the case where the graph is not series/parallel, the problem was first treated by Elmaghraby & Pulat [6], and later by De et al [3] who provide an update to the DP algorithm of Hindelang and Muth, and show that the DTCTP is, in general, strongly NP-Hard (by reduction from 3SAT). In a more recent contribution van Houcke [30] utilizes the electromagnetism algorithm meta-heuristic to resolve the
When elements of the project are not known deterministically, analysis of the project network becomes much more difficult. The dearth of literature on the topic is perhaps good evidence of this fact. Even so, several studies have been performed both in the single and multiple mode fields. Golenko-Ginzburg and Gonik [15] investigate the RCPSP when faced with stochastic activity times. They use Beta, Normal, and Uniform distributions for activity times and determine a heuristically achieved best solution to the problem of minimizing the project’s expected completion time. They schedule start times for tasks and proceed heuristically. The authors resolve the problem of deciding on the “entering variables” (the activities to be initiated at a particular state of the project) by first using simulation to determine the probability that the activity is on the critical path and then solve a 0-1 Knapsack Problem. Stork [27] takes an in-depth look at the RCPSP in the stochastic case. He investigates the implications of AND/OR constraints to the problem. He also briefly discusses the Linear Time-Cost Trade-off problem and shows it is NP-hard when AND/OR constraints are present. He further investigates branch-and-bound, forbidden set, and priority based procedures for optimization, and their relationship to the “robustness” of a solution. Valls et al. [29] study the RCPSP with some activities facing stochastic interruptions and processing times. They deal with the concept of stochastic programming versus robust optimization and the burdens to computation imposed by multiple stages stochastic programming. Optimizing the weighted tardiness of either each activity or the total project, the authors propose and validate a meta-heuristic solution that blends Tabu Search and Scatter Search techniques. Wollmer [33] studies a version of the linear time cost trade-off problem (LTCTP) with an additional stochastic parameter. He deals with the problem where activity durations have a component that is a linear function of the allocation plus a random element; a model that approaches ours. The author first seeks to find the minimum required investment subject to an expected project completion time and then solves the “dual”
problem of finding the minimum expected completion time subject to a budgetary constraint. The author utilizes a cutting plane technique for his solutions. Gutjahr et al. [16] consider scheduling problems similar to our own; however, in their approach activities can have one of two possible distributions depending on whether or not a project is crashed. In this vein, our problem may be considered one in which projects can be "crashed" to (continuously) varying degrees (i.e., depending on the allocation of resource to tasks). Gutjahr et al. are also under the discrete time assumptions but, depending on the degree of discretization, this assumption can be overcome. Further, they define an “HS-Branch-and-Bound” algorithm that solves deterministic sub-problems using heuristic methods the “H” in the title is for “heuristic” and the “S” is for “stochastic”), and they have had success using this approach as opposed to Total Enumeration, or pure Stochastic Branch-and-Bound. They also suggest using different heuristics in different applications.

1.1.2 CLOSELY RELATED RECENT STUDIES

A few studies have been conducted recently that are in a similar vein to the research reported upon here. However, they differ from our study in either the objective, or the optimization methodology, or the constraining set. Tereso, Araújo, and Elmaghraby [28] investigate optimal resource allocation under stochastic conditions. They optimize the same objective with which we are faced and relax the assumption that work content follows an exponential distribution. They solve the problem using a dynamic programming (DP) algorithm that is extremely taxing computationally. Their approach optimizes the allocation of the resources along a longest (in the sense of the number of arcs) single path through the network from start to finish while holding the allocations to other activities fixed. This necessitates the eventual enumeration over the “fixed” allocations in order to secure the “unconditioned” optimum, which gives rise to the computational complexity of the approach. In their conclusions they suggest treating only exponential distributions which gives rise to the research pre-
sent in this paper. Elmaghraby & Morgan [12] derive a fast method for optimizing resource allocation in a single stage stochastic activity network. The justification for optimizing a single stage is that a project manager’s strategy should change once the state of the project changes as evidenced by the completion of an activity. They use Sample Path Optimization combined with Geometric Programming to solve the single stage problem. Repeatedly applying the single stage procedure they derive a methodology for solving the multiple stage project. The procedure could be applied to any distribution of work content provided it is amenable to random sampling. The approach also allows for absolute bounds on the resources to be applied; see also Morgan [22] for a more extensive report on the approach.

Elmaghraby & Ramachandra [13] study the problem of optimal resource allocation in a Markov PERT Network with respect to an economic objective that is similar to ours. Their objective function is comprised of the expected cost of the resource usage and the cost of expected tardiness. Though the cost of expected tardiness is a major simplifying assumption when compared to the expected cost of tardiness, the authors demonstrate through extensive Monte Carlo experimentation that this approach to solution delivers an excellent approximation to the optimal solution. They first describe a “DP Policy Iteration-like” approach to solution and then show a simulation-cum-optimization approach to the problem. They use Monte Carlo and Latin Hypercube Sampling approaches and show the latter to have advantages with respect to variance minimization. The authors further admit that any aggregate bounds on resource allocation would need to be applied in the search technique. (For a more extensive discussion of the various approaches mentioned, see Ramachandra [24].)

Finally, Azaron et al. [1] use control theory to study resource allocation in Markov PERT Networks. The authors explain that the resource allocation costs and project tardiness costs are in conflict and therefore decide to model the problem as a multi-objective stochastic program. Ultimately the authors describe Goal Attainment and
Goal Programming methodologies to solve the problem and provide computational results for a few examples. The authors state that their solution allows for activity times to be taken from any distribution as a generalized Erlang distribution. They arrive at the conclusion that an optimal solution to the control problem cannot be found and so they resort to the discretization of time, modeling the problem with difference (rather than differential) equations and use non-linear programming to solve the problem.

2 STATEMENT OF THE PROBLEM

A project is represented as a directed acyclic graph $G = (N, A)$ where $N$ is the set of nodes representing the “events” of the project and $A$ is the set of arcs representing the activities. This is commonly referred to as the activity-on-arc (AoA) representation of the project. Precedence among the activities is represented by the sequence of arcs along any chain in the graph – an activity cannot be initiated until all its predecessors are completed (strict precedence relation as distinguished from generalized precedence; see [9]).

We are given for each activity $a$ its work content, $W_a$, representing the amount of work (effort, energy) to be done in the activity. We assume that $W_a$ is known in probabilistic terms. It is important to note that we consider the source of uncertainty in $W_a$ to be internal to the activity; “external” uncertainty, such as that due to labor absenteeism, machine failure or inclement weather, is considered to be beyond the control of the manager and is in addition to the internal uncertainty embedded in $W_a$. It is not considered here, but more will be said about it in § 6. The work content is measured in some convenient unit depending on the activity and the resource, such as “worker-hours” or “computer-seconds”, etc. Consequently, only when the resource allocation level is specified can we estimate the time to complete the activity. Thus we have a relationship among three entities: work content $W$, resource allocation level
$x$, and activity duration $Y$. If a simple hyperbolic relationship is assumed then the activity duration is equal to the work content divided by the resource level, as shown in eq. (2). We are also given a due date for the project, $T_s$, which is specified by the project owner and is known with certainty. We let $x_a$ represent the allocation of the resource to activity $a$; it is the decision variable. We seek to find the best vector of allocations $\{x_a\}_{a \in A}$ that optimize the objective function stated in (5) below.

We consider only one resource, assumed to be of abundant availability, although in real life application the particular resource under scrutiny may vary from activity to activity within the project; e.g., one activity may require electricians as a resource while another may require a specialized tool maker; but no activity requires more than one (critical) resource at the same time. This could be accommodated in the proposed model by replicating the activity arc in parallel with the same source and terminal nodes for each resource required by the activity.

Costs incurred in the execution of the project are two: resource costs and lateness costs. We assume a constant marginal cost for the resource, $c_R$, and also assume that this cost is constant across all activities. (The latter assumption is made for ease of computation; it can be easily relaxed in the algorithm detailed in the next section.) We are also given a marginal cost of lateness for the project, $c_L$, per unit time of delay beyond a specified due date $T_s$. In our application, we normalize these costs by dividing through by $c_R$. Again, this operation serves to simplify the computations; in any application more realistic costs would be known and could be used (such as piece-wise linear and convex costs). If we let $\Upsilon_n$ denote the time of realization of the last node in the project graph, which also connotes project completion, a random variable (r.v.), the cost of tardiness is defined as $c_L \cdot \max\{0, \Upsilon_n - T_s\}$.

We seek to find an allocation vector, $X$, that minimizes the expected value of the sum of the these two costs as shown in (5) below.
3 THE MODEL

The first major assumption we make is that, for all activities, the activity’s work content \( W_a \) is exponentially distributed with parameter \( \lambda_a \),

\[
W_a \sim \exp (\lambda_a), \forall a \in A.
\] (1)

As we show below, this assumption will greatly decrease our computational burden because of the memory-less property of the exponential distribution.

Further, we assume that the resource allocation, \( x_a \), is bounded from below by \( l_a \) and from above by \( u_a \)

\[
0 < l_a \leq x_a \leq u_a < \infty, \forall a \in A.
\]

Note that this assumption does not conflict with the earlier assumption of unlimited resource availability – it only constrains the resource allocation to any individual activity. We assume the total resource availability is abundant enough to accommodate all activities running in parallel at a given time.

Given our assumptions of exponential work content and bounded resource allocation, we let \( Y_a \) denote the duration of activity \( a \), given as

\[
Y_a = \frac{W_a}{x_a}.
\] (2)

As \( W_a \) is exponentially distributed and \( x_a \) is a decision variable, it can be easily demonstrated that \( Y_a \) also follows an exponential distribution with parameter \( x_a \lambda_a \).

Note that this definition necessitates our bounding of \( x_a \) away from 0; without such bound on \( x_a \) the duration of the activity may not necessarily take on meaningful values. Justification for the assumption of exponential distribution is discussed in more detail in § 6.

Finally, we assume that the cost of resource allocation to activity \( a \) is quadratic in the allocation over the duration of the activity. This reflects the added cost of employing additional resources. It also could be considered representative of the law
of diminishing marginal returns. It has the added advantage of resulting in a linear function of the allocation $x_a$, which simplifies the calculations considerably. Thus the cost of resource allocation to activity $a$ is defined as:

$$C_{R,a} = c_R \cdot x_a^2 \cdot Y_a = c_R \cdot x_a \cdot W_a,$$

the second equality is by (2).

Therefore, the expected cost due to the resource allocation to all the project activities, denoted by $C_R$, is given by (recall that the marginal resource usage cost has been normalized to 1),

$$\mathbb{E}[C_R] = \sum_{a \in A} \left( \frac{x_a}{\lambda_a} \right).$$

(3)

In order to properly evaluate the portion of the project cost due to tardiness we must first establish the distribution of the project completion time. To do so we establish the “state space” of the project following Kulkarni and Adlakha [19], which leads us to a continuous time Markov chain (CTMC) model of the project’s progress and the interpretation of the desired distribution as a Phase-Type Distributions (PH-D’s). For the sake of self-containment, the basic concepts of these two domains are summarized in the Appendix.

## 3.1 THE LATENESS COST FUNCTION

The PH-D can be used to formally state the cost function over which we seek to optimize. As of the moment, we have in hand the expected cost of resource allocation, given by (3) above; it remains to derive the portion pertaining to the expected cost of tardiness.

First, as previously stated, the cost of tardiness is defined by $C_L(\Upsilon) = c_L \cdot \max \{0, \Upsilon - T_s\}$. With the distribution of $\Upsilon$ following the PH-D (see the Appendix)
we must use integration to calculate the expected cost of tardiness:

$$
\mathcal{E} [C_L] = \int_0^\infty C_L(t) \cdot f(t) \, dt \\
= \int_0^\infty C_L(t) \cdot \alpha \cdot e^{T^t} \cdot T^0 \, dt \\
= \int_{t=T_s}^\infty c_L(t-T_s) \cdot \alpha \cdot e^{T^t} \cdot T^0 \, dt \quad (4)
$$

It is important here to remember that the entries of the matrix $T$, as well as $T^0$ are determined by the completion times of the activities in $A$. Under our assumptions these completion times are functions of the resource allocations to the activities in the project. Thus, the expected cost of tardiness is also a function of the vector $X$. In total, we seek to find the optimal resource allocation $X$ to minimize the expected project cost:

$$
\mathcal{E} [C(X)] = \sum_{a \in A} \left( \frac{x_a}{\lambda_a} \right) + c_L \int_{t=T_s}^\infty (t-T_s) \cdot \alpha \cdot e^{T^t} \cdot T^0 \, dt, \quad (5)
$$
in which the value $c_R$ has been dropped out of the equation as it has been set equal to 1 and $c_L$ has been normalized with respect to it, subject to the bounds on resource allocation as well as the precedence constraints expressed as

$$
\Upsilon_j - \Upsilon_i \geq Y_{(i,j)}, \quad \forall (i, j) \in A. \quad (6)
$$

4 THE PROCEDURE

From the given project network (the activity network (AN)) we establish the state pace network (SPN) $\mathcal{S}$ which is independent of any assumption concerning probability distributions. $\mathcal{S}$ is the set of all possible combinations of active and dormant activities of the project that can occur according to the precedence constraints, extended by the empty set, $\emptyset$, (see Appendix). The assumption of exponentially distributed work content leads to the construction of a CTMC model of the progress of the project. For any given resource allocation vector $X$ one can easily construct the rate matrix
$Q$ of transfer from state $s_p$ to state $s_q$ with entry $x_{(i,j)}\lambda_{(i,j)}$ in cell $(s_p, s_q)$ in which activity $(i, j)$ is identified as the activity that completes processing first and causes the transition from from state $s_p$ to state $s_q$ in the state space.

These concepts are best understood relative to an example. Consider the project represented by the graph in Fig. 1 with $\Lambda = (\lambda_1, \lambda_2, \lambda_3) = (0.2, 0.1, 0.07)$. The state space $S$ contains 6 states, which are enumerated in Table 1 and represented graphically in Fig. 2. Assume an initial allocation of $X^{(0)} = (1, 1, 1)$. The rate matrix $Q(X, \Lambda)$ of the CTMC is immediately available in (7) with the specified $\lambda$-parameters.

### Table 1. State Space of the CTMC of Project 1.

<table>
<thead>
<tr>
<th>Activity 1</th>
<th>Activity 2</th>
<th>Activity 3</th>
<th>State</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Active</td>
<td>Idle</td>
<td>Active</td>
</tr>
<tr>
<td>1</td>
<td>Idle</td>
<td>Active</td>
<td>(1a,3a)</td>
</tr>
<tr>
<td>2</td>
<td>Active</td>
<td>Idle</td>
<td>Dormant</td>
</tr>
<tr>
<td>3</td>
<td>Idle</td>
<td>Active</td>
<td>Dormant</td>
</tr>
<tr>
<td>4</td>
<td>Idle</td>
<td>Dormant</td>
<td>Active</td>
</tr>
<tr>
<td>5</td>
<td>Idle</td>
<td>Idle</td>
<td>(\emptyset, \emptyset)</td>
</tr>
</tbody>
</table>

$$Q(X, \Lambda) = \begin{bmatrix} -0.27 & 0.20 & 0.07 & 0 & 0 & 0 \\ 0 & -0.17 & 0 & 0.07 & 0 & 0 \\ 0 & 0 & -0.20 & 0.20 & 0 & 0 \\ 0 & 0 & 0 & -0.10 & 0 & 0.10 \\ 0 & 0 & 0 & 0 & -0.07 & 0.07 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$ (7)

Figure 1 here.
Project 1 activity network.

Figure 2 here.
The state space of the AN of Fig. 1.
With this resource allocation the expected lateness is 13.514 and the lateness cost is $c_L \times 13.514$. The total cost is (recall that the resource usage cost $c_R$ was normalized to 1.)

$$1 \times \left( \frac{1}{0.2} + \frac{1}{0.10} + \frac{1}{0.07} \right) + c_L \times 13.51402.$$  

Is the allocation $X^{(0)}$ optimal? Ideally one should take the partial derivative of the criterion function (5) with respect to the components of the decision vector $X$, equate the partial derivatives to 0, and solve for the optimum. (We are assured of achieving the minimum because of the convexity of the criterion function in each component of $X$, see [13]) Such course of action is not feasible because the lateness cost is an implicit function of the resource allocation $X$, and the manner of its variation cannot be determined analytically, to the best of our knowledge.

In order to resolve this seeming impasse we focus our attention on a “modified derivative descent” algorithm. We first compute an approximation to the partial derivative of the expected cost with respect to the allocation to each of the activities, select the activity that has the greatest negative derivative and then modify the allocation to it. The question of “how much should the modification be?” is resolved by appeal to the Fibonacci search technique to find the minimum expected cost allocation to the chosen activity while holding all other allocations constant. The algorithm iterates using these two steps until a specified stopping criterion is met.

In order to execute any of these steps, we first must be able to compute the expected cost at a particular allocation, a problem which is addressed next.

### 4.1 COST COMPUTATION

As previously described, the expected cost of the project is given in (5) and repeated here for convenience,

$$\mathbb{E} [C(X)] = \sum_{a \in A} \left( \frac{x_a}{\lambda_a} \right) + c_L \int_{t=T_s}^{\infty} (t - T_s) \cdot \alpha \cdot e^{Tt} \cdot T^0 dt.$$
It contains two terms: The first relates exclusively to the level of resource allocation. and the second relates to the tardiness of the project with respect to the given due date \( T_s \). Since the work content distribution parameter, \( \lambda_a \), is given for each activity, the expected cost of resource allocation can be computed directly for any vector \( X \). The second term cannot be computed directly due to the complexity of the integrand and the need to evaluate \( e^{T_t} \) (see the Appendix). We must settle upon a summation instead of integration to approximate the expected cost of tardiness. Applying a summation requires that we assume time takes on discrete values. Azaron et al. [1] arrive at the same conclusion through the use of control theory. They take the approach of dividing the “time interval” – that is, the interval of time to project completion – into \( K \) portions of length \( \Delta t \) each and solving a system of difference equations. However, we choose to depart from this approach since the “time interval” of the project is not known \( a \) priori; it is a function of the allocation \( X \). For example, a trivial project involving only a single activity will have a significantly shorter “time interval” when it is allocated 3 units of the resource than when it is allocated just 1 unit. Instead, we set the length of the time increment \( \Delta t \) and take the sum of tardiness costs until the probability that the project takes longer than \( q \cdot \Delta t \) time units is sufficiently small, say \( p_\Delta \). Since the expectation of cost involves computing the probability that the project completes within a certain time interval, this method of calculation requires no additional computational steps.

We therefore take a midpoint Riemann sum of the product \( (\varsigma \cdot \Delta t - T_s) \cdot \alpha \cdot e^{T_t} \cdot T_0 \), \( \varsigma = 1, 2, \ldots, q \), to approximate the value of the integral. For suitably small values of \( \Delta t \) this method provides an excellent approximation. The downside, however, is that while decreasing \( \Delta t \) (to increase the precision of the approximation) we are also increasing the computational burden. We settle this trade-off by letting \( \Delta t \) equal to a fraction of the standard deviation of the project completion time, provided it is greater than a lower bound \( l_\Delta \), as this helps to normalize the computational time for each instance in which the expected project cost is computed.
The computational burden required by the cost calculation stems from the calculation of the probability that a project completes in a specified window of time. Recall from (19) in the Appendix that $F(\cdot)$ requires $e^{T_t}$ to be computed. Since the $T$ matrix of transients in the CTMC is not changed during the calculation of the expected cost, much can be gained if we let $H = e^T$. Equation (19) then becomes

$$F(t) = 1 - \alpha \cdot H^t \cdot e, \text{ for } t \geq 0. \quad (8)$$

This computation requires computing $H^t$ rather than $e^{T_t}$.

The algorithm for computing the cost then proceeds as follows.

### 4.1.1 ALGORITHM FOR COMPUTING LATENESS COST

**Step 1 Initialization.** Let

- $t_n = T_s$, ($t_n$ denotes the time of realization of node $n$);
- $\Delta t = \max \{z \cdot \sigma_{t_n}, l_{\Delta t}\}$, ($z$ is a predefined fraction, typically $\in [0.02, 0.05]$),
- $C_P = 0$,
- $p_0 = F(T_s)$, the probability that the project completes no later than time $T_s$,
- $H_{mat} = H^t$, the matrix $H$ raised to the power $t$,
- $U = H^{\Delta t}$, the matrix $H$ raised to the power $\Delta t$.

**Step 2** Calculate the probability $p$ that $t_n$ falls in the interval $[t, t + \Delta t]$ beyond the due date $T_s$,

$$p = p_I - p_0. \quad (9)$$

where,

$$p_I = F(t + \Delta t) = 1 - \alpha \cdot H_{mat} \cdot U \cdot e \quad (10)$$

**Step 3** Let $C_P = C_P + c_L \cdot p \cdot ((2t + \Delta t)/2 - T_s)$. If $p_I$ exceeds $1 - p_s$ then go to Step 4. Else, let $p_0 = p_I$, let $t = t + \Delta t$, let $H_{mat} = H_{mat} \cdot U$, and repeat from Step 2.
Step 4 Add the expected cost of resource usage to the expected tardiness cost, $C_L$, using (3) and stop. $C_P$ now equals the expected total project cost.

In the algorithm, we use $U$ and $H_{mat}$ in order to eliminate the need to calculate the power of an exponential in each step. Instead, we calculate these quantities once (from knowledge of the $W_a$’s and the resource allocation $X$) and can then rely on matrix multiplication, which is significantly less burdensome computationally than matrix exponentiation.

With an effective way to calculate the total expected project cost in hand, we can now describe the optimizing procedure we employ.

4.2 SELECTING THE CANDIDATE ACTIVITY

In general, our algorithm optimizes the project cost by changing the allocation to one activity at-a-time. We now explain the procedure to select activities that are the best candidates for optimization. The best candidates in our procedure are those that could lead to the greatest decrease in the expected project cost.

We have observed before that the two terms of the project total cost (resource cost and lateness cost) behave diametrically opposite in their response to changing the resource allocations: The project resource cost increases linearly with an increase in the resource allocation to any activity (at the rate of $1/\lambda_a$). On the other hand, increased resource allocation to the activities tends to shorten their duration (recall that $Y = W/x$) and so the expected tardiness cost would decrease or stay the same. A decrease in the overall cost, therefore, could be obtained from either a decrease or an increase in resource allocation.

An important observation is that the expected project cost is a convex function with respect to the resource allocation to a single activity; a result of the two convex components of cost: the expected cost of tardiness decreases convexly due to the exponential distributions on which it is based; further, since the resource costs are linear (hence convex) in the resource allocation, the sum of the two costs is convex.
Thus, repeatedly optimizing allocations to single activities one-at-a-time will descend monotonically to reach the optimal solution.

Since no analytical expression is known for the partial derivative of the total cost function with respect to the allocation to any activity, \( x_a \), we must rely on approximations to proceed. Such an approximation would enable us to select the activities to which a change in allocation could cause the greatest change in expected cost. The approach harks back to the very definition of a (partial) derivative. If we compute the cost associated with a small change in allocation \( \delta \) to a particular activity, we can approximate the derivative of the cost function by first taking the difference between the resulting cost and the cost of the initial resource allocation and then dividing by the magnitude of \( \delta \). Since a decrease in cost can occur due to an increase or a decrease in allocation, and thanks to the convexity of the cost function, the best candidate allocations are those with the steepest derivatives causing a decrease in cost. Further, if \( \delta \) is constant across all activities, we can simply find the change in allocation reflecting the greatest decrease in cost, as we are not concerned with the actual value of the derivatives, only in their relative magnitudes.

Let \( X_{+a} = X + \delta \cdot e_a \) denote the allocation \( X \) with the value of the \( a \)th allocation increased by the amount \( \delta \), where \( e_a \) is a vector of dimension \( |A| \) with zeros everywhere except in position \( a \) where it is equal to 1. Similarly, define \( X_{-a} = X - \delta \cdot e_a \) to denote the allocation with the \( a \)th allocation decreased by \( \delta \). We calculate \( \mathcal{E} [C (X_{+a})] \) and \( \mathcal{E} [C (X_{-a})] \) for each activity. With \( \mathcal{E} [C (X)] \) in hand, we find \( \mathcal{E} [C (X_{+a})] - \mathcal{E} [C (X)] \) and \( \mathcal{E} [C (X_{-a})] - \mathcal{E} [C (X)] \) for each activity \( a \), and use this as a surrogate for the value of the derivatives of the cost function with respect to the allocation \( x_a \). It is important to check both an increase and decrease in allocation rather than assuming that an increase in cost in one direction implies a decrease in cost in the opposite; for certain network structures both increasing and decreasing the allocation to an activity can cause an increase in cost. We select the activity corresponding to the greatest decrease in cost as the candidate activity (in case of multiple candidates,
select any one).

**4.2.1 IMPROVING THE COMPUTATIONAL EFFICIENCY**

As with computing the cost function, approximating its derivative can be computationally burdensome, considering that we must calculate $E\left[ C(X+a) \right]$ and $E\left[ C(X-a) \right]$ for each activity at each iteration. We let $Q(X+a)$ represent the matrix $Q$ with the allocation to activity $a$ increased by the value $\delta$. Note that changing the allocation to a single activity may result in changing the values of several entries in the original $Q$ matrix, as well as changing the corresponding elements along the main diagonal. Changing the allocation to activity $a$ by $\delta$ implies that we must change the values in the $Q$ matrix by $\delta \lambda_a$. This change can be viewed as matrix addition. These facts are best illustrated with an example from the project in Fig. 1, where we change the allocation to activity 1 ($\lambda_1 = 0.20$) by adding $\delta = 0.1$:

$$Q(X+1) = \begin{bmatrix}
-0.27 - 0.02 & 0.20 + 0.02 & 0.07 & 0 & 0 & 0 \\
0 & -0.17 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.20 - 0.02 & 0.20 + 0.02 & 0 & 0 \\
0 & 0 & 0 & -0.10 & 0 & 0.10 \\
0 & 0 & 0 & 0 & -0.07 & 0.07 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$= Q + \begin{bmatrix}
-0.02 & 0.02 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -0.02 & 0.02 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

or, symbolically,

$$Q(X+1) = Q + \Phi_{+1}$$
where $\Phi_{+1}$ represents the matrix corresponding to the increase in allocation to activity 1. Note that the matrix $\Phi_{+a}$ does not depend on the current allocation to the other activities.

Recall from elementary matrix algebra that $e^{A+B} = e^A \cdot e^B$ only when $A$ and $B$ are commutative: $A \cdot B = B \cdot A$. Unfortunately, in our case $Q \cdot \Phi_{\pm a} \neq \Phi_{\pm a} \cdot Q$. However, the sparse nature of the $\Phi_{\pm a}$ matrix implies that these two terms are approximately equal. Further, $e^{\Phi_{\pm a}}$ is nearly equal to the identity matrix $I$. With these facts in mind, one may use $e^{Q \cdot \Phi_{\pm a}} \approx e^Q \cdot e^{\Phi_{\pm a}}$, or rather the transient matrices $e^{T+\phi_{\pm a}} \approx e^T \cdot e^{\phi_{\pm a}}$, where $\phi_{\pm a}$ now denotes the transient portion of $\Phi_{\pm a}$, for use as an approximation in the calculations for the density function of the completion time distribution. This approximation provides a good indicator of the derivative of the cost function with respect to direction (as in whether increasing the allocation leads to an increase or a decrease in cost) as well as the magnitude of the change in cost. We validate this approximation in the next section by comparing the expected costs returned by the algorithm when using the approximation and the exact determination of the cost derivatives.

The key advantage provided by this approximation is that it eliminates the need to calculate the matrix exponential for the evaluation of the derivative at each iteration. Since $\Phi_{\pm a}$ does not depend on the current allocation, the $e^{\phi_{\pm a}}$ matrices can be computed once at the beginning of the algorithm and stored for use throughout the optimization process. Since $e^T$ is needed for calculating the cost at each iteration, this matrix is in hand as well. Thus, $e^T \cdot e^{\phi_{\pm a}}$ is computed and passed to the cost calculation algorithm as the matrix $H$.

### 4.3 Optimizing the New Allocation to an Activity

With a candidate activity in hand, we now seek to optimize the allocation of the resource to that activity while leaving the other allocations unchanged. As previously
stated, the expected project cost is a convex function with respect to the allocation to any single activity. Any convex optimization procedure could therefore be applied here. However, due to the difficulty involved in finding exact analytical expressions for the partial derivatives of the cost function, we opt to use Fibonacci search as our method of determining the optimal allocation to the selected activity. Note that this “optimal” allocation is, in a sense, locally optimal as it depends upon the allocation to the other activities.

At its core, Fibonacci search finds the optimal point in a range of feasible values by repeatedly shrinking the range, stopping when the range is sufficiently small to suggest a single optimal value. Fibonacci search makes use of the so-called “golden ratio,” which Wilde [32] shows to be computationally the “most efficient” method (in the sense of the number of function evaluations) of finding the optimal value of a unimodal variable when dealing with one-dimensional search.

Given a lower bound, \(l\), and an upper bound, \(u\), on the range of possible values of a variable, two new points are calculated within the range: \(m_l\) and \(m_u\) with \(m_l < m_u\). We define \(r\) as the inverse of the “golden ratio” and use \(r\) to give values for \(m_l\) and \(m_u\):

\[
r = \frac{2}{1 + \sqrt{5}} \approx 0.618, \tag{11}
\]

\[
m_l = l \times r + u \times (1 - r), \tag{12}
\]

\[
m_u = l \times (1 - r) + u \times r. \tag{13}
\]

For our purposes, these points represent different resource allocations. If activity \(a\) were selected as the candidate activity with a current allocation \(x_a\) and its cost decreases with increased allocation then the interval to be searched (the so-called “interval of uncertainty” (i.o.u.)) is \([x_a, u_a]\) while keeping the allocations to all other activities unchanged. The values \(x_a, m_{la}\) and \(m_{ua}\) are illustrated in Fig. 3. If the
cost of the activity decreases with decreased allocation then the i.o.u. is \([l_a, x_a]\). For example, suppose the current allocation in the project given in Fig. 1 is \(X = (2, 2, 2)\) with lower and upper bounds equal to 1 and 3; respectively, for all allocations. If activity 1 is selected as the candidate activity and its cost decreases with increased allocation we let \(l = X\) (the current allocation) and \(u = (3, 2, 2)\). On the other hand, if the cost decreases with decreased allocation, we let \(l = (1, 2, 2)\) and \(u = X\). Values for \(m_l\) and \(m_u\) are then defined by (12) and (13); respectively.

The interval of uncertainty (i.o.u.) \([x_a, u_a]\) and the points \(m_{l_a}\) and \(m_{u_a}\).

With these four values in hand \((l, u, m_l\) and \(m_u)\), we recursively redefine the bounds on the range to hone our search to the minimum expected cost allocation. Optimization proceeds by first calculating the expected costs of allocations \(m_l\) and \(m_u\). Convexity of the expected cost function allows us to conclude that the minimum falls between the greater of the two values and the opposite absolute bound; i.e., if \(\mathcal{E}[C(m_l)] < \mathcal{E}[C(m_u)]\) then the optimal value must lie between \(l\) and \(m_u\), otherwise \(\mathcal{E}[C(m_l)] > \mathcal{E}[C(m_u)]\) and the optimal value must lie between \(m_l\) and \(u\). If the two expected costs are equal, either bound can be used. Using these values we redefine the i.o.u. as either \([l, m_u]\) or \([m_l, u]\) depending on whether \(\mathcal{E}[C(m_l)] < \mathcal{E}[C(m_u)]\) or \(\mathcal{E}[C(m_l)] > \mathcal{E}[C(m_u)]\); respectively. The new i.o.u. shall always contain one of the two added points, \(m_l\) or \(m_u\) at the correct ratio \(r\) from one extremity of the i.o.u. One new point is inserted and its expected cost is calculated in each iteration, and the shrinking of the i.o.u. (at approximately 61.8%) continues until the i.o.u. is “small enough”; i.e., when \(u - l < \varepsilon_{lu}\) for some small \(\varepsilon_{lu}\). When this is reached, any allocation within the interval \([l, u]\) could be taken as the optimal solution; however, we elect to use the midpoint \((l + u)/2\). These concepts are illustrated numerically in § 4.5 below.
4.4 THE STOPPING CRITERION

The optimizing algorithm repeats the derivative approximation and Fibonacci search steps, finding a resource allocation with a lower expected cost in each iteration. The algorithm stops (in the sense of declaring the current allocation as the optimum) when the expected cost in a given iteration improves the expected cost by an amount smaller than a given “small enough” value $\varepsilon_v$. If $X^{(k)}$ represents the best resource allocation after iteration $k$, the algorithm terminates when $E[C(X^{(k+1)})] - E[C(X^{(k)})] < \varepsilon_v$.

Another potential stopping criterion would be when the approximated derivatives are all sufficiently close to 0. This criterion, however, would require a more exact calculation of derivatives, as currently we are concerned only with identifying the derivative causing the greatest decrease in cost. Further, this criterion requires all derivatives to be less than the predefined $\varepsilon_d$. Assume, without loss of generality, that the last iteration increased the allocation to activity $a$ and caused a decrease in expected cost less than $\varepsilon_v$. This implies that $E[C(X_{k+1})] - E[C(X_{k})]$ (the approximated derivative in the previous iteration) had the minimum value across all activities. If the difference in allocation after the Fibonacci search procedure is less than $\delta$, for a sufficiently small $\delta$, the current allocation is effectively unchanged, therefore, further iteration will not improve the solution. Further, if the new allocation to activity $a$, after the Fibonacci search procedure, differs from the previous allocation by more than $\delta$, then $E[C(X_{k+1})] - E[C(X_{k+1})] < E[C(X_{k+1})] - E[C(X_{k})]$, implying that all derivatives are less than $\varepsilon_d$. In other words, if we move in the direction of the steepest derivative and end with an effectively different allocation yet the expected cost decreased by less than $\varepsilon_v$, then $E[C(X_{k})]$ is less than $E[C(X_{k+1})]$ from the previous iteration. This implies that all the approximate derivatives at the previous allocation were less than $\varepsilon_v$, since the steepest derivative did not cause a change of more than $\varepsilon_v$. Convexity of the cost function implies that this will hold true at the new allocation as well.
4.5 AN ILLUSTRATIVE EXAMPLE

Consider the project represented by the graph in Fig. 1. We assume an initial allocation of $X^{(0)} = (1, 1, 1)$ with lower and upper bounds on allocation defined at 1 and 3; respectively, for all activities. In our calculations we let $\delta = 0.05$, $c_L = 3$, $T_s = 8$ and $\varepsilon_v = 0.005$. (Recall that $c_R = 1$. ) First we compute the expected cost of the initial allocation which turned out to be $\mathcal{E}[C(X^{(0)})] = 69.8278$. Next, we approximate the derivative of the expected cost with respect to the allocation to each activity. And since the resource allocation to all the activities is at their respective lower bounds, we need only investigate increasing the allocation.

Table 2: Example Derivative Approximation.

<table>
<thead>
<tr>
<th>Activity</th>
<th>$X_a$</th>
<th>$\mathcal{E}[C(X_{\pm a})]$</th>
<th>$\mathcal{E}[C(X^{(0)})] - \mathcal{E}[C(X_{\pm a})]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1, increase</td>
<td>(1.05, 1, 1)</td>
<td>69.3272</td>
<td>0.5006</td>
</tr>
<tr>
<td>2, increase</td>
<td>(1, 1.05, 1)</td>
<td>69.5934</td>
<td>0.2344</td>
</tr>
<tr>
<td>3, increase</td>
<td>(1, 1, 1.05)</td>
<td>69.1142</td>
<td>0.7136 $\leftarrow$ $\text{max}$</td>
</tr>
</tbody>
</table>

Since activity 3 yields the approximate derivative of greatest decrease in cost as its allocation is decreased, it becomes the candidate activity. Thus, we take the bounds $l = (1, 1, 1)$ and $u = (1, 1, 3)$ and we begin the Fibonacci search procedure with $\varepsilon_d = 0.01$. Since $u_3 - x_3 = 2$, we know the Fibonacci search procedure will take 12 iterations, as $2 \times r^{12} < 0.01$. Table 3 details the optimization steps.

Table 3: Fibonacci Search Procedure Example.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>$l_3$</th>
<th>$u_3$</th>
<th>$m_{l_3}$</th>
<th>$C(m_{l_3})$</th>
<th>$m_{u_3}$</th>
<th>$C(m_{u_3})$</th>
<th>i.o.u.</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>3</td>
<td>1.7639</td>
<td>69.325</td>
<td>2.2361</td>
<td>73.7746</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2.2361</td>
<td>1.4721</td>
<td>67.7198</td>
<td>1.7639</td>
<td>69.325</td>
<td>1.2361</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1.7639</td>
<td>1.2918</td>
<td>67.5562</td>
<td>1.4721</td>
<td>67.7198</td>
<td>0.7639</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1.4721</td>
<td>1.1803</td>
<td>67.9483</td>
<td>1.2918</td>
<td>67.5562</td>
<td>0.4721</td>
</tr>
<tr>
<td>4</td>
<td>1.1803</td>
<td>1.4721</td>
<td>1.2918</td>
<td>67.5562</td>
<td>1.3607</td>
<td>67.5179</td>
<td>0.2918</td>
</tr>
<tr>
<td>5</td>
<td>1.2918</td>
<td>1.4721</td>
<td>1.3607</td>
<td>67.5179</td>
<td>1.4033</td>
<td>67.5592</td>
<td>0.1803</td>
</tr>
<tr>
<td>6</td>
<td>1.2918</td>
<td>1.4033</td>
<td>1.3344</td>
<td>67.5189</td>
<td>1.3607</td>
<td>67.5179</td>
<td>0.1115</td>
</tr>
<tr>
<td>7</td>
<td>1.3344</td>
<td>1.4033</td>
<td>1.3607</td>
<td>67.5179</td>
<td>1.3769</td>
<td>67.5318</td>
<td>0.0689</td>
</tr>
<tr>
<td>8</td>
<td>1.3344</td>
<td>1.3769</td>
<td>1.3506</td>
<td>67.5126</td>
<td>1.3607</td>
<td>67.5179</td>
<td>0.0426</td>
</tr>
<tr>
<td>9</td>
<td>1.3344</td>
<td>1.3607</td>
<td>1.3444</td>
<td>67.5107</td>
<td>1.3506</td>
<td>67.5126</td>
<td>0.0263</td>
</tr>
<tr>
<td>10</td>
<td>1.3344</td>
<td>1.3506</td>
<td>1.3406</td>
<td>67.51</td>
<td>1.3444</td>
<td>67.5107</td>
<td>0.0163</td>
</tr>
<tr>
<td>11</td>
<td>1.3344</td>
<td>1.3444</td>
<td>1.3382</td>
<td>67.5188</td>
<td>1.3406</td>
<td>67.51</td>
<td>0.0101</td>
</tr>
<tr>
<td>12</td>
<td>1.3382</td>
<td>1.3444</td>
<td>1.3406</td>
<td>67.51</td>
<td>1.3420</td>
<td>67.5102</td>
<td>0.0062</td>
</tr>
</tbody>
</table>
Thus the new allocation at the end of the first iteration is $X^{(1)} = (1, 1, 1.3413)$, where 1.3413 is the midpoint between $m_{l_3} = 1.3406$ and $m_{u_3} = 1.3420$ in the final step. The algorithm continues from this new allocation. After 5 total iterations the algorithm stops at an optimal solution of $X^{(5)} = (1.4284, 1.3150, 1.4171)$ at a cost of 64.3559. It is interesting to remark that Ramachandra [24] solved the same example problem, though his cost function accounted for lateness in terms of the cost of expected lateness, and he limited his resource allocations to increments of 0.25 between the lower and upper bounds. Ramachandra’s procedure resulted in a solution of $X^* = (1.5, 1.5, 1.5)$, which is different from our $X^{(5)}$, at an “expected cost” of 62.38. Taking into account the fundamental differences in the cost functions and resource allocations, these solutions are more or less equivalent (the difference is less than 3.1%).

5 COMPUTATIONAL RESULTS

The algorithm was tested on a set of 33 networks representing a variety of project network structures. Project networks were generated randomly using RanGen2, developed by Vanhoucke et al [31]. Details regarding RanGen2 and its use can be found in the Appendix. The RanGen2 generator allows the user to select networks by specifying values of six parameters. The values for these parameters were chosen to represent a wide variety of network structures and properties, such as the number of activities, the degree of parallelism, the length of the longest path in the graph, etc. Work content distribution parameters (the $\lambda_a$’s) were generated randomly in Matlab. These parameters were sampled from a uniform distribution between 0.1 and 2.0. The lower bound on resource allocation was taken as $l = 0.1$ and the upper bound was $u = 3.0$ units for all activities. Relative tardiness costs were fixed at $c_L = 3$ for all examples and the width of the final i.o.u. $\varepsilon_v = 0.005$.

Testing was conducted on a laptop computer, Dell Inspiron 1505, 2.4 GH with
dual processors, using the optimization algorithm initiated from 5 input allocations (i.e., for each network the optimization procedure was repeated 5 times, each with a different initial resource allocation). These allocations included the allocation of 1 resource unit, 2 resource units, and a randomly selected whole number of resource units between 1 and 3 to each activity. The optimization algorithm was then run to completion for each of the 5 initial allocations. Details regarding the test cases and the test results can be found in Table 4. Several additional networks were generated, however, we were unable to achieve the optimum allocation therein because the size of the state space in their CTMC representations were prohibitively large for the computer at our disposal.

Test Results with Indicators.

The indicators used by Vanhoucke et al. [31] can briefly be described as follows:

- \(I_1\) – The number of activities in the network.
- \(I_2\) – A series vs. parallel indicator. Low values for \(I_2\) indicate the graph has a high degree of parallelism and high values indicate that the graph follows a more series representation. \(I_2 = 1\) indicates that all activities occur in series.
- \(I_3\) – describes the spread of activities over the progressive levels of the graph.
- \(I_4\) – Indicator representing the presence of “short arcs” (arcs between nodes numbered almost successively).
- \(I_5\) – Indicator representing the presence of long arcs (arcs between nodes numbered apart from each other).

Further details on these indicators, including equations for their calculation are provided in Appendix A. Vanhoucke et al. [31] also specify an \(I_6\) as an indicator.
representing the “topological float” of the activities; however, after specifying only these five indicators we were often left with only one randomly generated network; therefore we did not use it.

These results show that this algorithm is relatively efficient as many of the networks were solved in a few seconds, on average. No network averaged longer than 37 minutes to solve. The computed optimal costs across all 5 input allocations differed by less than 5% in all of the test networks and less than 1% in 82% of the test networks. Note that decreasing $\Delta t$, thus making the integral portion of the cost function more accurate, could increase the precision of the returned solutions while increasing the computation time; however, the obtained level of accuracy was satisfactory for our purposes.

The last column of Table 4 measures the percentage difference between the minimum solution determined by our optimization algorithm and the solution determined using the exact evaluation of the matrix exponential in the derivative approximation step. As can be seen, the absolute difference from the exact optimum never exceeded 2.24% in all instances, and is less than 0.5% in 39 of our 43 experiments. This illustrates the validity of the approximation used in this step of the optimization procedure (described in more detail in § 4.2).

Interestingly enough, while the exact derivative procedure often arrives at a better solution than the approximation, equally often the approximation yields a lower optimal cost. Such a result is possible because when we deal with approximations to the derivatives and different initial allocations, activities may be selected in differing orders. Defining our stopping criterion as we have done allows iteration to conclude in a small range of allocations, thus the exact procedure may conclude in this range at a greater optimal cost than the approximation. This adds further weight to the contention that the exact procedure does not provide any real advantage in terms of accuracy over the approximation pursued in our algorithm.
5.0.1 NETWORK STRUCTURE CONSIDERATIONS

One important observation that can be seen immediately is the correlation between the number of states in the CTMC and the computation time required for optimization. As the number of states in the CTMC grows, the computational requirement of the algorithm increases exponentially as demonstrated in Fig. 5. This is due to the many matrix operations required by the algorithm, especially exponentiation. Though many of these exponentiations have been simplified or approximated throughout the algorithm, it is impossible to remove them completely and retain the spirit of the CTMC method of analysis. The remaining steps involving the value of $e^T$ are concentrated, primarily, in the Fibonacci search procedure, where the accuracy required for optimizing resource allocation to a single variable demands precise calculation of $e^T$ and it is not advantageous to make simplifying assumptions, as we did in the candidate activity selection step. Furthermore, the storage requirement becomes especially burdensome when the state space grows into the thousands and even millions. This eventuality is possible for projects with as few as 10 activities depending on the structure of the network (recall that the state space grows exponentially in the number of activities in parallel).

Solution Time vs. Number of States in CTMC.

A few observations can be made regarding the projects in our sample that shall help determine how many states will exist in the CTMC. First, independent parallel paths are multiplicative in producing states in the CTMC, implying the aforementioned exponential growth. Activities added in series are additive in producing states in the CTMC. These facts are further illustrated in the networks presented in Fig. 7. In Fig. 7(a), activities 1 and 3 are in series whereas in (b) these two activities are in parallel. The network of Fig. 7(b) yields twice the number of states in its CTMC than that of Fig. 7(a).
Another observation is that projects with a greater number of “progressive levels” (see the Appendix) tend to have fewer states in their CTMC representations than other projects with the same number of activities. A large number of progressive levels means that a long path exists within the network, leaving fewer activities to lie on parallel paths.

An additional observation is that often the complexity index (CI) of a project, defined by Bein, Kamburowski, and Stallmann [2], is less indicative of the difficulty of a project than in other project scheduling problem scenarios (such as in estimating the completion time). This stems from the fact that nonzero complexity indices are often derived from arcs extending between multiple, otherwise distinct, parallel paths. These “cross arcs” often dramatically reduce the number of states in the CTMC representation. Additionally, the optimizing algorithm could be considered robust in that it does not rely heavily on specific precedence constraints in its iteration, further reinforcing this idea.

Some of these concepts can best be made clearer with an example. Consider the networks in Figs. 6 through 9. Figure 6 provides an example of a network with 11 activities and CI = 3. This example has 44 states in its representative CTMC.

Figure 6 here.
Example Project 2.

Figure 7 provides two examples of networks obtained from the example in Fig. 6, both with 11 activities and equivalent CI’s. These graphs have 42 and 44 states in their CTMC’s; respectively. Note that the graph of Fig. 7(a) has an additional progressive level that is lacking in Fig. 7(b) and both graphs have CI = 2.

Figure 7 here.
Project 2 with Reduced Complexity.
Figure 8 provides another example with 11 activities obtained from the example in Fig. 6, again with a reduced complexity index. This graph has 28 states in its CTMC and $\text{CI} = 1$.

Figure 8 here.  
Project 2 With Further Complexity Reduction.

Figure 9 provides a network with 11 activities that is series/parallel i.e. $\text{CI} = 0$; it has 120 states in its corresponding CTMC. It provides an excellent example of how series/parallel graphs often have more states in their corresponding CTMC’s than graphs with greater CI.

Figure 9 here.  
Series/Parallel Network.

It is also important to note that the reductions in the CI obtained through the reductions between Figs. 6,7 and 8 lead to reductions in the number of states in the CTMC, however reducing the graph to a completely series/parallel graph produces more states than the original graph in Fig. 6.

These five project networks, represented graphically in Figs. 6-7 were tested using the optimization algorithm in the same manner as the randomly generated networks. Test results are given in Table 5. The project parameters are as follows:

$$\lambda_a = \{0.6885, 1.1724, 1.3330, 0.4007, 0.2630, 1.3469, 0.3725, 1.6764, 1.2735, 1.7496, 1.5418\},$$

$$T_s = 15,$$

$$c_L = 3.$$ 

Table 5 (Fig. 10) here.  
Example Network Test Results.
These tests further verify the statements made above regarding the number of states in the CTMC being indicative of the solution time. Further, this information confirms that a nonzero CI can dramatically reduce the number of states in the CTMC. Here, the percentage deviation from the minimum value is less than 1% regardless of the input allocation. Further, the percentage deviation of the solutions obtained using the exact computation of the matrix exponential in the candidate activity selection step from the minimum obtained through the use of our algorithm is less than 1% in each of these five cases as well.

6 LIMITATIONS, CONCLUSIONS AND DIRECTIONS FOR FUTURE RESEARCH

The objective of this research was to develop and test an algorithm to solve the problem of optimal resource allocation in AN’s with exponential work content distributions. Previous papers have been concerned with optimizing either multiple objectives, as in Azaron et al. [1], or an economic objective evaluating the cost of resources and the cost of expected tardiness, as in Ramachandra [24]. Our research instead dealt with the expected cost of a project as the sum of resource costs and the expected cost of tardiness.

In the narrative above the remark was made that in specifying the stochastic nature of the work content $W$ we are dealing only with the “internal” uncertainty of the activity, and a word of clarification is in order. Traditionally, researchers have been concerned with uncertainty in the activity duration due to “external” chance events (such as the weather). We prefer to make a clear distinction between such random occurrences, which indeed impact the duration of the activity, and the uncertainty that is inherent in the very nature of the activity itself – independent of any factor that is “external” to the activity. Perhaps the best examples of such “internal” indeterminateness are research and development activities: the uncertainty in their work content is primarily independent of any factor external to the activities.
Manipulation of the resources allocated to such activities, which is the central concern of this paper, is then a meaningful undertaking, which is not the case relative to the “external” factors (little can be done about the weather!). To be sure, these external uncertainties can be added to the treatment presented here to secure a complete integration of the uncertain elements in the conduct of the project.

We now discuss the assumption that the work content follows an exponential distribution which, *prima facie*, may not be palatable to many. The assumption can be justified on three grounds. Firstly, the memory-less property of the exponential distribution aids greatly in computation and, as we have shown in § 4 and the Appendix, lends well to the use of continuous time Markov chain (CTMC) theory and PH-D’s for its analysis. Secondly, and perhaps more importantly, the exponential distribution is the limit of the class of distributions known as “new better than used in expectation” (NBUE) distributions, to which all proposed distributions of activity durations known to us belong. Lastly, using the exponential distribution and modeling the project as a CTMC can be considered the fundamental “building block” which serves other potential distributions of the work content. This is due to the fact that any continuous distribution with rational Laplace transform can be approximated to any desired degree of accuracy by the generalized Erlang distribution (GED) [23]. The GED is a PH-D in which each “phase” is exponentially distributed – and we are back to the CTMC model based on these exponential distributions.

Thus, conceptually, one possible extension to the research presented here is to relax the assumption that the work content follows an exponential distribution. This assumption predicates our ability to model a project as a CTMC. In such a model, however, several activities in series now replace the original single activity. Such a change would cause a dramatic increase in the state space of the CTMC and in the computing time. It should be mentioned that this fact was noted by Azaron et al [1] who used it in an illustrative example of 10 activities with one activity possessing a generalized Erlang distribution of only two stages. As they noted relative to the
CTMC model:

“The limitation of the proposed model is that the state space can grow exponentially with the network size. As the worst case example, for a complete transformed network with $n$ nodes and $n(n-1)/2$ activities, the size of the state space is given by Kulkarni and Adlakha [19] is

$$N(n) = U_n - U_{n-1},$$

where,

$$U_n = \sum_{k=0}^{n} 2^{k(n-k)}.$$

In practice, the number of activities in an activity network is generally much less than $n(n-1)/2$. ... It should also be noted that for large networks, any alternate method of producing reasonably accurate answers will be prohibitively expensive.”

This explains our caveat above concerning the “conceptual” extension of our CTMC model to non-exponential work content. It also explains why Azaron et al [1] limited their example to a single activity with a generalized Erlang distribution of only two stages.

The discussion of the directions for future research serves also to implicitly highlight the limitations of our work.

We have assumed a hyperbolic relationship between the parameters $W$, $x$ and $Y$. A direct extension to this research would be to consider more general functional relationships. For example, one might investigate the use of $Y = W/x^\alpha$ where $\alpha \in (0, 1)$ represents the potential decrease in the “efficiency” in resource utilization due to the interference, and $\alpha > 1$ represents synergy in the use of more of the resource. In our research the hyperbolic relationship ($Y = W/x$) meshed well with the quadratic assumption regarding resource cost by creating a linear function of $W$. This quadratic assumption could be relaxed as well by using a factor $\beta$, transforming eq. (??) into

$$C_R = c_R \cdot x^\beta_a \cdot Y_a = c_R \cdot x^{\beta-\alpha}_a \cdot W_a.$$

(14)
For simplicity we have assumed that the cost of tardiness is a linear function of the difference between the completion time $\Upsilon_n$, a r.v., and the due date $T_s$, but a more general form of tardiness cost could be incorporated. Considering other forms of the tardiness cost, such as quadratic or piecewise linear, yields a variety of possible research extensions and would offer an opportunity to test the sensitivity of the optimal solution to various cost functions.

Further, the due date itself could be taken as a decision variable as in much of the project scheduling research. Such an extension would increase the complexity of the problem greatly as a change in the due date can cause extensive changes in the allocation of resources to the activities.

Another possible extension to this research would be the application of this optimization approach to other stochastic problems that can be modeled using CTMC’s. Though the cost would be evaluated by a different function, the optimizing procedure could perform in the same way, provided the cost function remains convex. Thus the algorithm could be modified to optimize continuous time Markov Decision processes where the transition rates are continuous functions of bounded input parameters. Examples of such problems may be machinery maintenance modeling where failure times are random, automatic guided vehicle routing, and emergency services planning. These scenarios have already been modeled as discrete time Markov Chains. Without much further effort, continuous models could be developed which could be solved with this algorithm.

A few research extensions pertain to the resource requirements. One key limitation of our treatment is the single resource assumption. Many projects have activities requiring the simultaneous use of different types of resources. Such problems are often difficult to contain within the confines of the CTMC model, but there are certainly many advantages to be gained by their successful solution.

Additionally, we assume no aggregate limit on the resource availability. Adding an aggregate resource constraint would certainly be beneficial in many real life scenarios.
Yet another avenue of future research may be to consider Generalized Activity Networks (GAN’s). The research conducted in this paper deals with a predetermined set of activities that must be completed with specified “strict precedence” constraints. Often, a project has certain decision points at which the planner may have a choice of the activities that will be abandoned and the activities that shall be undertaken, often with different costs and/or rewards. In some scenarios this choice of activities is specified by a probability distribution, in others it follows a “wait and see” policy. Researchers would need to consider both the choice of activities as well as the resource allocation to the chosen activities in this subject area.
7 APPENDIX

7.1 CONTINUOUS TIME MARKOV CHAINS (CTMC)

When we assume independence of activity work content with respect to one another, combined with the assumption of exponential distribution, the analysis of project networks is greatly aided by the use of Continuous Time Markov Chain (CTMC) theory; see Ross [25]. The model was dubbed “Markov PERT Network” by Kulkarni and Adlakha [19] who developed recursive formulas for obtaining the moments of the completion time of any node in the network. We describe Kulkarni and Adlakha’s notation which we have used in our work.

Consider a graph $G = (N, A)$ representing the project in the AoA mode of representation with $n = |N|$. Without loss of generality, we assume that the project begins at time zero and will end at a randomly distributed later time, $\Upsilon_n$, once all the activities have been completed. At a particular time $t$ between 0 and $\Upsilon_n$ each activity in $A$ can be in one of three states:

- **Active**: An activity is “active” if it is in process at time $t$.
- **Dormant**: An activity is dormant if it is completed but at least one other activity ending at the same node is still active at time $t$.
- **Idle**: An activity is idle if it is neither active nor dormant. This includes all activities that are yet-to-be-initiated and those that are completed.

Clearly, a project is completed when it has no active or dormant activities. Since each activity is in one of these three states Kulkarni and Adlakha define the following sets for any time $t \geq 0$:\footnote{We have changed notation to conform to our notation in this paper.}

1. We have changed notation to conform to our notation in this paper.
\[ V(t) = \{ a \in A : a \text{ is active at time } t \} \]

\[ D(t) = \{ a \in A : a \text{ is dormant at time } t \}, \]

and specify the state of the project as given by:

\[ S(t) = (V(t), D(t)). \]

The set of idle activities is the complement of the set \( S(t) \) in \( A \).

We need also to define a “cutset” and “uniformly directed cutset”. Given a project graph \( G = (N, A) \) in the AoA representation, let \( B \) be a proper, non-empty subset of the set of nodes \( N \) containing the “start” node 1, and \( \overline{B} \) be its complement in \( N \) containing the “terminal” node \( n \). Further, define \( (B, \overline{B}) = \{(i, j) \in A : i \in B, j \in \overline{B}\} \) where \( i \) and \( j \) denote the start and end nodes of activity \((i, j)\). If every directed path in the network from node 1 to node \( n \) has exactly one arc in the set \((B, \overline{B})\), the set of edges \((B, \overline{B})\) is defined as an \((1-n)\) cut or cutset. Further, the cutset is a uniformly directed cutset (udc) if no arcs extend from \( \overline{B} \) into \( B \); symbolically, \((\overline{B}, B) = \emptyset\). \( S = \{S(t), t \geq 0\} \) is a representation of the state of the project which forms a continuous time Markov chain (CTMC) with a single absorbing state when mutual independence of the activities is assumed. This stems from the fact that \( S(t) \) forms a 2-partition of a udc for all times \( t \). Thus the state space \( S \) is the set of all admissible 2-partitions of udc’s in \( G \), and \( S(t) \in S \) for all \( t \in [0, \Upsilon_n] \). When \( t \geq \Upsilon_n \) the sets \( V(t) \) and \( D(t) \) are both empty, \( V(t) = \emptyset = D(t) \), and the state space \( \mathcal{S} = S \cup (\emptyset, \emptyset) \) forms the extended state space for the CTMC. Thus \( \mathcal{S} \) is the set of all possible combinations of active and dormant activities of the project that can occur according to the precedence constraints, extended by the empty set \((\emptyset, \emptyset)\).

A few observations made by Kulkarni and Adlakha are worthy of note due to their importance and relevance to our research.

- Transitions in the CTMC correspond to the completion of activities in the project, and their associated rates correspond to the parameters of the comple-
tion time distribution. This directly implies that, as far as the time of sojourn of the process is concerned, the transition rates within the CTMC are random variables with rates \( \{x_a \lambda_a\} \).

- The state \((\emptyset, \emptyset)\) corresponds to the completion of the project. With no activities left to initiate or complete, no transitions are possible from this state and so \((\emptyset, \emptyset)\) is the single absorbing state of the CTMC.

- Since no activity can be completed more than once, each state of the CTMC shall be visited at most once. Thus, the states of \(S\) are transient and the states of \(\overline{S}\) can be numbered so that the rate transition matrix is upper-triangular.

- The number of states in \(\{S(t) : t \geq 0\}\) is finite for projects with a finite number of activities, with \(|S| = \sigma\) and \(|\overline{S}| = \sigma + 1\). Therefore, the process \(\{S(t) : t \geq 0\}\) will be completed in finite time with probability 1.

- Finally, if the number of activities \(|A| = m\), then the number of states in the CTMC is bounded by:

\[
m \leq |\overline{S}| \leq 2^m,\]

with the lower bound occurring when all activities of the project are in series and the upper bound occurring when all activities are in parallel.

### 7.2 Phase Type Distributions (PH-D)

In order to establish the expected cost of tardiness, we must first describe the distribution of time until the project’s completion. Employing the CTMC model the project can be considered completed when the Markov process reaches its absorbing state. When an absorbing CTMC has a single absorbing state the distribution of time until absorption can be described by the phase type distribution (PH-D). The phase type distribution gets its name from the path taken by the project from initiation to
completion, which can be viewed as a series of stages, or “phases,” each having an exponential distribution.

The parameters of the PH-D are a probability vector \( (\alpha, \alpha_{\sigma+1}) \), and the CTMC’s rate generator matrix is given by

\[
Q = \begin{bmatrix}
T_{(\sigma \times \sigma)} & T^0_{(\sigma \times 1)} \\
0_{(1 \times \sigma)} & 0_{(1 \times 1)}
\end{bmatrix}
\]  

(16)

Here, the matrix \( T \) is representative of the transient states of the matrix, with \( T_{i,i} < 0 \) for \( 1 \leq i \leq \sigma \) and \( T_{i,j} \geq 0 \) for \( i \neq j \). Additionally,

\[
T^0 + T \cdot e = 0,
\]

(17)

in which \( e \) represents a column vector of dimension \( (1 \times \sigma) \) and \( \theta \) is a vector of zeros of equivalent dimension. In other words, the rows of the matrix \( Q \) sum to 0. An example of the \( Q \) matrix for the project in Fig. 1 was given in (7) above.

The vector \( \alpha \) is an initial probability vector, also called the “counting probability” of \( Q \). This vector represents the probability that the Markov process begins in a given state. As such, we have:

\[
\alpha \cdot e + \alpha_{\sigma+1} = 1
\]

(18)

For our purposes, we assume the project begins with no activities completed i.e., the project is in the first state with probability 1. Thus we have \( \alpha = [1, 0_{(1 \times \sigma)}] \).

In general, the PH-D, hereby denoted by \( F(\cdot) \), can be represented by \( (\alpha, T) \). A necessary and sufficient condition for the states of the CTMC represented by \( T \) to be transient is that the matrix \( T \) is nonsingular. As a result, \( T^k \to 0 \) as \( k \to \infty \). When given an initial probability vector \( (\alpha, \alpha_{\sigma+1}) \), the cumulative distribution function of the time to absorption in state \( \sigma + 1 \) is given by:

\[
F(t) = 1 + \alpha \cdot e^{Tt} \cdot e, \quad \text{for} \quad t \geq 0.
\]

(19)

The evaluation of \( e^{Tt} \) has been studied in depth. Traditionally, it is given by the Taylor series:

\[
e^{Tt} = \sum_{i=0}^{\infty} \frac{1}{i!} (Tt)^i.
\]

(20)
However, it can be calculated in a myriad of other ways as demonstrated by Moler and van Loan [21].

A few observations about the PH-D type are now necessary for a complete understanding of our research:

- In general, the function $F(\cdot)$ has a jump of height $\alpha_{\sigma+1}$ at $t = 0$. This is equivalent to the probability that the process begins in the absorbing state. For our research, this is irrelevant because we assume the project starts with no activities completed. In practice, this assumption could be relaxed in the interest of managerial flexibility i.e., after a certain amount of time, the manager could re-optimize his or her resource allocation according to the progress of the project at that time.

- The probability density function (pdf) $f(\cdot)$ is given by:

$$f(t) = \frac{\partial}{\partial t} F(t) = \alpha \cdot e^{Tt} \cdot T^0 \quad (21)$$

- The expected value and variance of the PH-D, respectively, are given by; respectively,

$$-\alpha \cdot T \cdot e \quad \text{and} \quad 2\alpha \cdot T^2 \cdot e, \quad (22)$$

and, in general, the non-central moments $\phi_i$ of $F(\cdot)$ are given by

$$\phi_i = (-1)^{-i} \cdot i! \cdot (\alpha T^i \cdot e), \quad \text{for} \ i \geq 0. \quad (23)$$

7.3 THE GENERATION OF THE TEST NETWORKS

All of our test networks were generated using RanGen2, developed by Vanhoucke et al [31]. The RanGen2 generator operates by generating a network in the activity-on-node (AoN) mode of representation with the pre-specified number of nodes (activities) and enough precedence relationships (arcs) to ensure that the value of the series/parallel indicator is at least as great as the user specified input value. Arcs
are then removed one by one until the input value for the series/parallel indicator is reached. Since the AoA mode of representation often necessitates the creation of “dummy activities” (to properly identify all precedence constraints) with no work content, the AoN mode is a convenient alternative. The AoN mode of representation, however, often includes dummy start and finish activities to give the project a definite starting and finishing points. As an example, the AoN representation of the project from Fig. 1 is represented in Fig. 11 with the additional start and terminal nodes, labeled s and t in the figure; respectively.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{Fig11.png}
\caption{Representation of Project 1 as AoN, with added start and terminal (dummy) nodes.}
\end{figure}

In each generation run, RanGen2 creates multiple networks based on the set of arcs removed in the last step of generation. Several other parameters are calculated for each network. RanGen2 users can then randomly select networks from this set based on the other parameters set at values of their choosing.

We limit ourselves to the explanation of the indicators \( \{I_h\}_{h=1}^{5} \) used in describing the networks (see Tables 4 and 5). To render the formal definitions of these indicators meaningful we need to define a few symbols that are used in the specification of these indicators. Note that the symbols may not carry the same meaning as used in the main text and should be understood within the context of RanGen2 only.

The “progressive level” of node \( j \), \( PL(j) \) is the maximum number of arcs to node \( j \) from the start node 1; calculated recursively from the expression \( PL(j) = \max_{i \prec j} \{ PL(i) + 1 \} \), in which the symbol “\( \prec \)” indicates “immediate predecessor”. Now we define \( m \) as the maximum progressive level in the network, \( m = \max_{j \in N} \{ PL(j) \} \). Thus \( m \) defines the number of links in the longest path in the network. At progressive level \( l \), the “width” of the network is denoted by \( \omega_l \), and is the number of nodes (activities) at that progressive level. Define the average width as \( \overline{\omega} = n/m \) (recall
that \( n \) is the number of activities. Next, we define the “length” \( \rho(i,j) \) of arc \((i,j)\) joining nodes (activities) \( i \) and \( j \), where \( i \prec j \), as the difference between the progressive levels of the end node \( j \) and the start node \( i \), \( \rho(i,j) = PL(j) - PL(i) \).

Clearly, the maximum length of any arc is \( m - 1 \). Let \( n' \) be the number of arcs in the network with length equal to \( l \); i.e., \( n' = \# \{(i,j) \in A | PL(j) - (i) = l \} \). Finally, let \( D \) be the maximum possible number of short \((l = 1)\) arcs in the network; then

\[
D = \sum_{l=1}^{m-1} \omega_l \cdot \omega_{l+1}.
\]

With these preliminary definitions in hand we are ready to define the “indicators” \( I_1 \) through \( I_5 \) used in Tables 4 and 5.

- \( I_1 = n \), the number of activities.

- \( I_2 \): series/parallel indicator,

\[
I_2 = \begin{cases} 
1, & \text{if } n = 1 \\
\frac{m-1}{n-1}, & \text{if } n > 1 
\end{cases}
\]  

This series/parallel indicator measures how close the graph holds to a series or a parallel representation, not whether or not the graph is series/parallel.

- \( I_3 \): measures the distribution of the activities over the progressive levels based on the width of each progressive level,

\[
I_3 = \begin{cases} 
0, & \text{if } m = 1 \text{ or } n \\
\frac{\sum_{h=1}^{m} |\omega_h - \omega|}{2(\omega-1)(\omega-1)}, & \text{otherwise}.
\end{cases}
\]

- \( I_4 \): the short arcs indicator; it measures the presence of short arcs based on the progressive levels of the endpoints of each arc in the network,

\[
I_4 = \begin{cases} 
1, & \text{if } D = n-n_1 \\
\frac{n'-n+n_1}{D-(n-n_1)} & \text{if } D > n-n_1
\end{cases}
\]  

- \( I_5 \): the long arcs indicator, it measures the presence of long arcs \((l > 1)\) based on the difference between the progressive levels of the endpoints of each arc in
the network,

\[ I_5 = \begin{cases} 
1, & \text{if } |A| = n - \omega_1 \\
\frac{\sum_{l=2}^{m-1} n' \cdot (\frac{m-l-1}{n-1}) + n'_1 - n + \omega_1}{|A| - n + \omega_1} & \text{if } |A| > n - \omega_1
\end{cases} \]

Since the test networks are created in the AoN mode, in order to proceed with the optimization algorithm we must be able to enumerate the states in the CTMC; and since transitions in the CTMC occur whenever an activity is completed, the CTMC will have a state corresponding to each possible combination of complete and incomplete activities. We can consider active plus the subset of idle activities that have not begun processing as “incomplete”, and dormant plus the subset of idle activities that have completed processing as complete. This classification of states corresponds to the set of udc’s in the AoN mode of representation. As an example the states of the project represented in the AoN mode in Fig. 11 (and in AoA mode in Fig. 1) are described in Table 6.

**Table 6: State Space of the AoN Example Network.**

<table>
<thead>
<tr>
<th>State</th>
<th>Activity 1 (AoN)</th>
<th>Activity 2 (AoN)</th>
<th>Activity 3 (AoN)</th>
<th>udc (AoN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Incomplete</td>
<td>Incomplete</td>
<td>Incomplete</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>Complete</td>
<td>Incomplete</td>
<td>Incomplete</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>Incomplete</td>
<td>Complete</td>
<td>Complete</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>Complete</td>
<td>Incomplete</td>
<td>Complete</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>Complete</td>
<td>Complete</td>
<td>Incomplete</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>Complete</td>
<td>Complete</td>
<td>Complete</td>
<td>6</td>
</tr>
</tbody>
</table>

Note that these states correspond to the states we have enumerated in § 4. Thus, the states of the CTMC can be found directly from the AoN mode of representation by enumerating the udc’s of the AoN project graph. Output from the RanGen2 application is not in either of the traditional forms (of an adjacency matrix or an arc-node incidence matrix). Instead, RanGen2 creates a matrix with a row for each activity (including a dummy start and dummy finish activity). The first column of the output matrix is the activity number. Next, multiple columns are added representing any resource measures. For our purposes, however, we generated activity specific details, that is, the work content parameters, using Matlab. Finally, there is
a column listing the number of arcs emanating from the particular activity followed by the destination nodes of these arcs. As an example, had RanGen2 generated the project given in Fig. 1 without any resource measures, its output would be as shown in Table 7.

<table>
<thead>
<tr>
<th>Activity</th>
<th>Out-degree</th>
<th>Nodes</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>2</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This is the form of the “A” matrix given as input to the Matlab optimizing procedure. The complete Matlab code may be found in Rudolph [26].
References


Project 1 with three activities.

Figure 1:
The state space for the three activities project

Figure 2:
The insertion of the sample values $m_{la}$ and $m_{ua}$ in the $i.o.u.$ in case of increased allocation.
| Network | Indicators as per Vanhoucke et al | Number of States/CTMC | Min Sol'n Time, sec | Average Sol'n Time, sec | Range of Sol'n, $ | Maximum % Deviation Exact % Deviation From Minimum From Approx |
|---------|---------------------------------|-----------------------|---------------------|------------------------|----------------|------------------|-----------------------------|
| 1       | 8 0.25 0 0.16 1                  | 74                    | 15.049              | 18.878                 | 0.006          | 0.000987         | -9.9298E-05                |
| 2       | 8 0.25 0 0.83 1                  | 34                    | 2.350               | 3.239                  | 0.17           | 0.002232         | -0.003104115               |
| 3       | 8 0.25 1 0.16 1                  | 160                   | 111.359             | 165.440                | 0.004          | 0.000289         | -0.003296665               |
| 4       | 8 0.25 1 0.83 1                  | 130                   | 61.390              | 74.472                 | 0.016          | 0.0022571        | -0.002524656               |
| 5       | 8 0.25 0.33 1 1                 | 39                    | 2.440               | 2.628                  | 0.003          | 0.0000620        | -0.00544422                |
| 6       | 8 0.75 0.8 1 1                  | 11                    | 0.144               | 0.296                  | 0.015          | 0.001165         | -0.0004186                 |
| 7       | 8 0.75 0.8 0.5 1                | 12                    | 0.513               | 0.570                  | 0.434          | 0.002897         | -0.003032106               |
| 8       | 8 0.75 0.8 0.5 0.5             | 17                    | 0.151               | 0.354                  | 0.097          | 0.003333         | 0.00633469                 |
| 9       | 8 0.75 0.99 0 1                 | 25                    | 0.708               | 1.224                  | 0.075          | 0.001804         | 0.00195625                 |
| 10      | 8 0.75 0.99 0 0.25             | 23                    | 0.614               | 0.782                  | 2.575          | 0.024365         | 0.000917178                |
| 11      | 12 0.25 0 0.5 0.8              | 77                    | 24.228              | 29.261                 | 0.009          | 0.000625         | 0.00066837                 |
| 12      | 12 0.25 0 0.16 1               | 228                   | 681.933             | 1056.203               | 0.033          | 0.000799         | 0.028677817                |
| 13      | 12 0.25 0.33 0.21 0.66         | 246                   | 1123.007            | 1787.993               | 0.077          | 0.000999         | 0.022445409                |
| 14      | 12 0.75 0.75 1 1               | 16                    | 0.707               | 0.828                  | 0.059          | 0.005738         | -0.00696522                 |
| 15      | 12 0.75 0.75 0.5 1             | 33                    | 1.797               | 2.441                  | 0.012          | 0.001089         | 0.003382672                |
| 16      | 12 0.75 0.75 0.5 0.61          | 20                    | 0.846               | 1.062                  | 0.033          | 0.001005         | 0.002566588                |
| 17      | 12 0.75 0.99 0 1               | 80                    | 16.706              | 24.237                 | 0.024          | 0.001839         | 0.003422305                |
| 18      | 12 0.75 0.99 0 0.28            | 58                    | 9.773               | 13.206                 | 1.804          | 0.016880         | 0.003180397                |
| 19      | 16 0.25 0.22 1 1               | 93                    | 49.454              | 70.324                 | 0.013          | 0.000779         | 0.000181381                |
| 20      | 16 0.75 0.9 0 1                | 196                   | 491.106             | 918.612                | 1.450          | 0.015716         | 0.016436256                |
| 21      | 16 0.75 0.72 0 0              | 126                   | 30.913              | 105.635                | 0.032          | 0.001949         | -0.001014286               |
| 22      | 16 0.75 0.72 0 1               | 128                   | 113.700             | 173.073                | 0.306          | 0.007965         | -0.007040657               |
| 23      | 16 0.75 0.72 0 0.5            | 58                    | 5.387               | 14.399                 | 0.325          | 0.016273         | -0.00150206                 |
| 24      | 16 0.75 0.81 0.5 0.92         | 26                    | 1.369               | 2.020                  | 1.366          | 0.003350         | -0.00347482                 |
| 25      | 20 0.25 0.08 0.82 1            | 96                    | 94.483              | 137.139                | 0.030          | 0.000601         | 0.021854665                |
| 26      | 20 0.75 0.7 0 1                | 234                   | 1711.437            | 2199.084               | 0.376          | 0.00432755       | 0.006396162                |
| 27      | 20 0.75 0.71 0 0.71           | 77                    | 36.775              | 48.880                 | 0.047          | 0.003015         | 0.004071669                |
| 28      | 20 0.75 0.71 0.42 0.9         | 35                    | 3.802               | 6.165                  | 0.092          | 0.001536         | 0.007773735                |
| 29      | 20 0.75 0.78 1 1              | 28                    | 1.928               | 3.354                  | 0.281          | 0.013102         | 0.003236254                |
| 30      | 20 0.75 0.78 0.62 1           | 31                    | 2.702               | 4.973                  | 0.057          | 0.001242         | 0.007256605                |
| AKS I   | 4 NA NA NA NA                | 7                     | 0.119               | 0.273                  | 0.007          | 0.002311         | 0.00222053                 |
| AKS II  | 6 NA NA NA NA                | 17                    | 0.360               | 0.490                  | 0.034          | 0.003725         | -0.00046115                |
| AKS III | 10 NA NA NA NA               | 25                    | 1.669               | 1.869                  | 0.040          | 0.002888         | -0.00125985                 |

Figure 4:
Solution Time vs. Number of States in CTMC

Figure 5:
Project 2 with 11 activities.

Figure 6:
Figure 7:
Project 2 with Further Complexity Reduction

Figure 8:
Series/Parallel Network

Figure 9:
<table>
<thead>
<tr>
<th>Network</th>
<th>I1</th>
<th>I2</th>
<th>I3</th>
<th>I4</th>
<th>I5</th>
<th>States CTMC</th>
<th>Min Sol'n</th>
<th>Average Sol'n</th>
<th>Range of Sol'n</th>
<th>Max % Deviation</th>
<th>% Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig 4</td>
<td>11</td>
<td>0.67</td>
<td>0.63</td>
<td>0.67</td>
<td>0.64</td>
<td>44</td>
<td>3.519</td>
<td>5.137</td>
<td>0.037</td>
<td>0.002283</td>
<td>0.00316431</td>
</tr>
<tr>
<td>Fig 5a</td>
<td>11</td>
<td>0.67</td>
<td>0.75</td>
<td>1.00</td>
<td>0.68</td>
<td>42</td>
<td>1.979</td>
<td>4.211</td>
<td>0.080</td>
<td>0.004914</td>
<td>0.00122164</td>
</tr>
<tr>
<td>Fig 5b</td>
<td>11</td>
<td>0.58</td>
<td>0.95</td>
<td>0.78</td>
<td>0.81</td>
<td>44</td>
<td>3.514</td>
<td>5.647</td>
<td>0.021</td>
<td>0.001321</td>
<td>0.00013087</td>
</tr>
<tr>
<td>Fig 6</td>
<td>11</td>
<td>0.75</td>
<td>0.78</td>
<td>1.00</td>
<td>0.76</td>
<td>28</td>
<td>0.730</td>
<td>1.633</td>
<td>0.061</td>
<td>0.003596</td>
<td>-0.00115808</td>
</tr>
<tr>
<td>Fig 7</td>
<td>11</td>
<td>0.50</td>
<td>0.83</td>
<td>0.20</td>
<td>0.60</td>
<td>120</td>
<td>69.177</td>
<td>104.086</td>
<td>0.026</td>
<td>0.001848</td>
<td>-0.00205271</td>
</tr>
</tbody>
</table>

Figure 10:
Project 1 in AoN presentation.

Figure 11: