Simulation-Optimization Framework for Stochastic Optimization of R&D Pipeline Management

Dharmashankar Subramanian
Honeywell Laboratories, Minneapolis, MN 55418

Joseph F. Pekny, Gintaras V. Reklaitis, and Gary E. Blau
School of Chemical Engineering, Purdue University, West Lafayette, IN 47907

The simulation-based optimization framework (Sim-Opt) uses a twin-loop computational architecture, which combines mathematical programming and discrete event simulation, to address this problem. This article extends our earlier work to present methods for integrating information from the inner loop (Sim-Opt time lines, reactive adjustment) and using it in the outer risk-control loop (Stochastic Optimization loop) to obtain statistically significant improvements in the solutions to the underlying stochastic optimization problem. Two classes of information can be obtained from the inner loop time lines: the first pertaining to portfolio selection and the second resource crowding associated with the chosen operation policy. Methods presented quantify the information on these two classes, and a three-step heuristic incorporates this information in the outer risk-control loop to drive the system toward improving solutions with respect to the mean net present value (NPV) of the portfolio and the probability of delivering a positive NPV. This method was used on a pharmaceutical product development case study, consisting of 11 projects, 154 activities, 14 resource types and a 20-year planning horizon with respect to patent expiration. Basic algorithm engineering efforts are also described to significantly improve the performance of formulation generation, the generation of a heuristic lower bound and the identification of cut families to effectively apply branch-and-cut methods.

Introduction

The research and development (R&D) pipeline management problem addresses the issues of a new-product-development pipeline, where several new-product-development projects compete for a limited pool of various resource types. Candidate projects are then subjected to a network of testing and development activities (see Subramanian et al. (2001) for a detailed discussion). Furthermore, there is a significant chance of failure associated with every activity. As soon as a product fails the requirements, all the remaining work on that product is halted and the investment in the previous testing tasks yields no returns.

The R&D pipeline problem is characterized by a sequence of combinatorial decisions that are made with respect to portfolio composition and resource allocation, both of which depend on the state of the system. Furthermore, the state of the system is subject to discrete changes upon the occurrence of events, and evolution in the face of uncertainty. Such sequential decision problems have the following key ingredients (Puterman, 1994):

- A set of decision epochs
- A set of system states (state space)
- A set of available actions (action space)
- A set of state and action dependent cost and reward
- A set of event-based state transitions (event space).

The R&D pipeline management problem can, thus, be viewed as the control problem of resource-constrained, performance-oriented, and stochastic discrete event dynamic systems. In this view it is desired to know the optimal actions

Correspondence concerning this article should be addressed to D. Subramanian.

that need to be taken with respect to portfolio composition and resource allocation corresponding to every state into which the system could transition via events, with the objective of optimizing the expected value of a performance measure that is accumulated and discounted over the problem horizon. The algorithmic map of states to actions is referred to as a *policy* (Cassandras, 1993). Guided by the above perspective, Subramanian et al. (2001) considered the R&D pipeline management problem, and presented a two-loop computational architecture called the simulation-based optimization framework (Sim-Opt) with a detailed discussion of the inner loop. This article extends the above work and presents methods for integrating information from the inner loop (Sim-Opt time lines (Subramanian et al., 2001)) and using it in the outer risk-control loop (Stochastic Optimization loop (Subramanian et al., 2001)) to obtain improvements in the solutions to the underlying stochastic optimization problem.

Background and Problem Description

A summary of the literature pertinent to the R&D pipeline management problem is given in Subramanian et al. (2001). The most notable contributions are those of Schmidt and Grossmann (1996), Honkomp (1998), Jain and Grossmann (1999), Blau et al. (2000) and Blau and Sinclair (2001). The problem description is as follows. A set of projects is given, where each candidate project (product) is described in terms of an Activity-on-Node (AoN) graph. The AoN graph corresponding to any project is a directed acyclic graph, where nodes represent activities (tasks) and arcs represent predecessor-successor relationships to capture technical precedence relationships between activities. Each project has a forecast of commercial worth and a desired due date, particularly in the pharmaceutical context, where a patent expiration deadline causes a steep decline in commercial value. Given for each activity within each project (each node within each AoN), are: (1) processing time (duration); (2) resource requirement of every applicable resource type; (3) probability of success to quantify the aspect of failure; (4) cost and reward information; and (5) due date information, after which activity loses value (especially important in the case of projects with patent expiration features).

Also given are the system capacities of the various resource types, and the discounting factor for financial calculations. The expected net present value (ENPV) is often chosen as a measure of performance. Uncertainty, in terms of appropriate probability distributions, enters into the problem for each activity in each project by way of: (1) activity duration distribution; (2) activity resource requirement distribution of every applicable resource type; (3) activity success probability distribution; (4) activity cost distribution and reward distribution.

A key question is: Given such a problem instance and a measurement of performance, what is the best set of projects to pursue, and, furthermore, what is the best way to assign resources to activities in the chosen projects, such that the chosen measure of performance is maximized?

Sim-Opt: A Computational Architecture

The details of the motivation and implementation of Sim-Opt can be found in Subramanian et. al (2001). This section provides a brief summary to motivate the following section on information integration from the inner loop of Sim-Opt time lines (Subramanian et al., 2001). A stochastic programming formulation as a chance-constrained formulation (Kall and Wallace, 1995) can be summarized as below. Obtain a portfolio and a task schedule over the planning horizon that:

Maximizes the *expectation* of the net present value (NPV) distribution of the pipeline system, subject to:

- Allocation Constraints
- probability $\left\{\begin{array}{l}\text{Satisfying Precedence Constraints}\\ \text{Satisfying Resource Constraints}\\ \text{Satisfying Demand Constraints}\end{array}\right\} \geq \alpha$
- probability $\{\text{NPV} \geq M\} \geq \beta$, for suitably chosen α, β, and M.

It should be noted that every constraint in the deterministic formulation (Subramanian et al., 2001) that depends on random parameters is now stipulated to jointly hold in a probabilistic sense, that is, with a joint probability of at least α. Precedence constraints and demand constraints involve random processing times, while resource constraints involve random resource requirements, random processing times, and estimates for the probabilities of success that are themselves random. An additional constraint that enters the stochastic optimization problem is the risk constraint. A practical form of the risk constraint is obtained by insisting that the NPV exceed a value M with a probability of at least β. This constraint enters the problem due to variability in the estimates of market returns and costs of activities. (Note that a value of $M = 0$ represents the positive side of the NPV distribution, that is, positive rewards.) All the essentials of the above stochastic optimization problem can be captured in a single mathematical program, which takes the form of a nonlinear, stochastic, integer program. It will involve constructing mathematical expectations and higher moments of the NPV distribution (objective function) and those of the probabilistic risk constraints and physical constraints. The resulting numerical complexity of multivariate numerical integration for the stochastic moments (Kall and Wallace, 1995) combined with the NP-Hardness of integer programming (Garey and Johnson, 1979) renders such a monolithic program arguably outside the set of effectively solvable problems.

Sim-Opt (see Figure 1) addresses this with a discrete event dynamic system view of the R&D pipeline by integrating

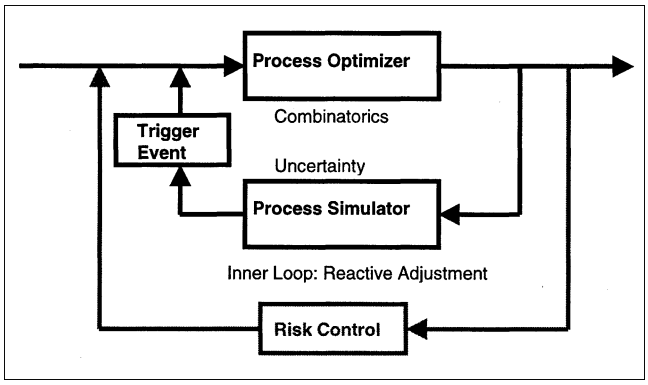

Figure 1. Sim-Opt.

combinatorial optimization and discrete event system simulation. The role of the inner loop of Sim-Opt is as follows. The optimizer establishes an initial state of the pipeline system by solving the resource-overbooked deterministic mixed integer linear program (MILP) formulation discussed in (Subramanian et al., 2001). The evolution of this initial state of the pipeline is tracked in a discrete event system simulation module, which forms the second element. The uncertainties associated with the task processing times, task resource requirements, task successes, task costs and rewards are modeled with appropriate probability distributions. Resource constraints are enforced in the system evolution at every point in time, as are the precedence relationships of the activities in the pipeline portfolio. As the system evolves in the constrained simulation mode to generate an artificial history, two kinds of possibilities are encountered. The first is that of a resource conflict that must be mediated among tasks that are simultaneously feasible within and across projects, and the second is that of the response to task failure that causes attrition of projects in the pipeline. When the simulation module encounters a need for either of the two types of control actions, it momentarily suspends itself and communicates the state of the system to the decision-making module. This solves a deterministic MILP that is appropriately modified to account for the current system state. The simulation is reprimed with the solution resulting from the optimizer, and it continues marching in time until its subsequent need for a control action. The simulation module marches in time, with an indeterminate number of departures to the optimizer, until a predetermined end of the planning horizon is reached, or a predetermined state like the exhaustion of all tasks in the pipeline is reached. One such realistic and *controlled* walk in time through the stochastic state space constitutes a *time line*.

The inner loop [Sim-Opt time lines (Subramanian et al., 2001)] of Sim-Opt consists of multiple time lines that are explored in a Monte-Carlo fashion to accumulate many unique combinations of realizations of uncertainty. The state-dependent control actions that are taken at different points in time within a simulated time line of the inner loop are referred to as "here-and-now" actions corresponding to the states at which they are taken. Due to the presence of uncertainties, it is important to note the following with respect to the scheduling solution that results from every deterministic MILP invocation from within the simulated time line. The notion of a *schedule* over a given horizon, as a deterministic, time-indexed assignment of resources to activities over their corresponding durations, loses its precision, as soon as its implementation commences in the Sim-module. It should also be noted that a deterministic time-indexed assignment of resources to activities implies a priority sequence for the scheduled activities, with the priority being defined with respect to the engagement of resources. The earlier an activity is scheduled to occur, the higher is its priority in engaging the corresponding resource types. This sequencing priority retains its meaning in the face of uncertainty, with ties broken suitably. The decision-making corresponding to assigning priorities to activities in the various projects at different points in time (and different corresponding states) with respect to engagement of respective resource types, defines the *policy* of operation. The objective of the optimization then is to construct a project portfolio (selection) and establish a policy for allocating limited resources to various activities of the projects in the chosen portfolio, so as to maximize the *mean* of the resulting probability distribution of the NPV of the portfolio. The NPV distribution of the portfolio is composed of negative and positive values, due to the presence of risk and reward in the system. It is also desired that the chosen portfolio and policy of operation provide an acceptable probability of achieving a positive NPV. In the inner loop of Sim-Opt, the deterministic (expected value based) optimizer decides the relative priorities of various activities that compete for resources at different points in time, and the simulation preserves these priorities.

The objective of the outer loop is to learn about the undesirable future effects of taking all "here-and-now" actions at various points in time (and corresponding to different states), with the solution from a deterministic optimizer. The outer loop's intent is to reflect this learning in the "here-and-now" action that is implemented at the present, along with the policy of operation that is chosen for the given system. This is so that a better solution is obtained with respect to both the *mean* of the resulting probability distribution of the NPV of the portfolio, and the probability of achieving a positive NPV. There are two classes of information that can be obtained from the inner loop time lines in the context of the new product development problem. The first class of information pertains to portfolio selection, and the second class of information pertains to resource crowding associated with the chosen policy of operation. The following section describes the ways in which valuable information can be integrated from such inner loop time lines, in the outer loop of Sim-Opt. The information can be used in the outer loop to drive the inner loop time lines towards improving solutions. The actual use of the above items of information will be illustrated in a pharmaceutical product development case study, consisting of 11 projects, 154 activities, 14 resource types and a 20 year planning horizon with respect to patent expiration.

Basic Algorithm and Software Engineering in Sim-Opt

This section discusses basic algorithm and software engineering issues that are pertinent to a process management framework, like Sim-Opt. It is evident from the problem description discussed earlier that the R&D pipeline management problem involves a data-intensive abstraction, namely a set of AoN graphs that describe the activity network of each candidate project. Furthermore, the hierarchical nature of data input is clear from the various levels at which data is required. The pipeline system is comprised of projects, which are in turn comprised of individual activities, which in turn are characterized in terms of connectivity and various complex data types for deterministic and stochastic information, as detailed earlier. A framework like Sim-Opt involves six steps from the perspective of implementation. These are a mixture of software engineering and algorithm engineering opportunities. They are:

(1) A *Data Model* that models the complex and hierarchical data needs in the form a structured input language.

(2) A parser that reads and interprets the above language in order to create an *in-memory* organization of the data modeled in Step 1, via suitable data structures.

(3) Formulation algorithms that act on the data structures of Step 2 in order to create mathematical programming and system simulation formulations at run-time (that is, inner loop of Sim-Opt).

(4) Solution algorithms that operate on the formulations created in Step 3. This pertains to every state-dependent MILP that is solved within any single time line.

(5) Effective means of tracking information from the inner loop time lines.

(6) Algorithms for utilizing the integrated information to lead to improving solutions for the underlying stochastic optimization problem (that is, outer loop of Sim-Opt).

The above steps represent a logical progression of data from input to output with suitable *interfaces* that operate between each of these steps to carry out the flow of data. The choice of implementation of each of these steps has significant implications for the complexity of the subsequent steps. For example, the choice of the data model and its representation in the form of a language affects the complexity of the parser that interprets the input. The data structures that act as the storehouse of all data in memory dictate the complexity of the problem formulation process, which in turn affects the solution process. Steps 1, 2, 3 and 5 predominantly represent opportunities for software engineering, that is, use of effective data structures, while Steps 4 and 6 represent opportunities for algorithm work in optimization. Issues pertaining to the data model and the language for representing input are outside the scope of this article (Varma et al., 2002), while those pertaining to the time complexity of formulation generation and solution are discussed in the following sections.

Every simulated time line in Sim-Opt requires several state-dependent MILP formulations to be generated at run-time, and solved to optimality. This makes Sim-Opt a computationally intensive framework, and, hence, requires time-complexity that is effective for both formulation generation and solution. This section describes the efforts undertaken to achieve significant improvements in the performance of formulation generation, and the generation of a heuristic lower bound along with the identification of cut families for effective application of branch-and-cut methods for formulation solution. The following discussion pertains to the MILP formulation that is discussed in Subramanian et al. (2001).

Formulation generation

First, a basic approach to formulation generation, which pays scant attention to data structures and time complexity, is described to understand the need for a judicious choice of data structures. Let n be the number of activities in the problem across all projects, H be the number of time periods in the uniform time discretization (UDM), and R be the number of resource types in the problem. In the basic approach, a binary variable (X_{it}) is generated for every activity (i), and corresponding to each time period (t) in the UDM horizon, by stepping over all the activities and the time periods. Each X_{it} variable holds its identity in terms of a string literal that represents the name of the activity (i) and the index of the corresponding time period (t). The resulting set of *all* such binary variables, with cardinality $O(nH)$, is stored in a single, linear list with no additional structure. The generation and storage of the set of "hold" (Subramanian et al., 2001) variables [$H_{ii't}$, cardinality $O(n^2H)$] and slack variables [S_{rt}, cardinality O(RH)] are also handled in a similar fashion, with a single linear list corresponding to each of these two variable sets. Furthermore, each $H_{ii't}$ variable holds its identity in terms of two string literals that represent the names of the activities (i and i') and the index of the corresponding time period (t) and each S_{rt} variable holds its identity in terms of a string literal that represents the name of the corresponding resource type (r) and time period (t). The generation time complexity in the basic implementation is $O(nH)$ for the X_{it} variables, $O(n^2H)$ for the $H_{i,i',t}$ variables, and $O(R^*H)$ for the S_{rt} variables. Also, all three variable sets are generated in the first step of formulation generation. With respect to constraints in the MILP formulation discussed in Subramanian et al. (2001), it should be noted that:

- An allocation constraint and a demand constraint are required for each activity.
- A material-balance-based precedence constraint is required for each immediate {predecessor, successor} pairs of activities at each time period.
- A renewable resource constraint is required for each resource type at each time period.

For the purpose of generation of each of these constraint families, there is a need to access the variables that participate in each constraint belonging to each family. For instance, to generate the material-balance-based precedence constraint corresponding to any given predecessor-successor pair of activities {i', i} at time period t, there is a need to access the variables, X_{it}, $H_{i,i',t}$, $H_{i,i',t-1}$ and $X_{i'(t-p_{i'})}$ given the names of activities {i', i} and the value of t. In this basic implementation, an $O(nH)$ search is required to access any desired X_{it} variable, given the name of activity i, and the value of time period t, from a single, simple list of size nH. Similarly, using the respective variable lists for the hold and slack variables, an $O(n^2H)$ search is required to access any desired $H_{i,i',t}$ variable, given the names of activities i, and i', and the value of time period t. Also, an $O(RH)$ search is required to access any desired S_{rt} variable, given the name of the resource type r, and the value of time period t. The above basic implementation effectively leads to $O(n^4H^2)$ performance for the generation of all material-balance-based precedence constraints, $O(n^2H)$ performance for the generation of allocation and demand constraints, and $O(RH^{2*}(\text{Max}\{p_i\})^*n^2 + R^2H^2)$ for generation of renewable resource constraints. Lastly, the generation of the objective function in the basic approach exhibits an $O(n^2H^2)$ complexity. Such a basic implementation can lead to unacceptable performances, particularly for a framework like Sim-Opt, which involves many time lines each requiring multiple state-dependent MILP formulations. The following section describes approaches to improving the performance of formulation generation by judiciously choosing data structures and performing variable domain reduction.

Improvements via variable reduction and data structures

The activity-on-node graph data structure corresponding to each of the candidate projects, which contains a unique name for each node, is used for variable domain reduction in the following manner. A variable domain reduction exercise is

carried out for the X_{it} variables, corresponding to each activity (node) i, in the UDM-based MILP formulation, with respect to the default range in the basic formulation, $[1,H]$, where H is the number of time periods (indexed from 1) in the UDM formulation and p_i is the processing duration of activity (node) i, in terms of the number of time periods. First, a standard critical path based graph algorithm is used to compute the Earliest Start Time Bucket Index ES_i, and the Latest Start Time Bucket Index LS_i, corresponding to each activity (node) i in every project (AoN graph). The recursive equations that give the earliest start and the latest start, along with the boundary conditions, are

$$ES_i = \max_{i'}\{ES_{i'} + p_{i'}\}, \qquad i' \in \text{Set of Immediate Predecessor Nodes of Node, } i \quad (1)$$

$$ES_i = 1, \text{ if } i \text{ has no predecessor} \quad (2)$$

$$LS_i = \min_{i'}\{LS_{i'}\} - p_i, \; i' \in \text{Set of Immediate Successor Nodes of Node, } i \quad (3)$$

$$LS_i = H - p_i + 1, \text{ if } i \text{ has no successor} \quad (4)$$

The minimal contiguous set of UDM time periods, representing the reduced variable domain, over which we define a start variable (X_{it}) for the corresponding activity (i) is identified as $[t_{i,\min},t_{i,\max}]$, where

$$t_{i,\min} = \min_{i'}\{ES_{i'} + p_{i'}\}, \; i' \in \text{Set of Immediate Predecessor Nodes of Node, } i \quad (5)$$

where $p_{i'}$ is the processing duration of activity i' in terms of number of time periods and

$$t_{i,\max} = LS_i \quad (6)$$

The choice of our lower terminus ($t_{i,\min}$ from Eq. 5) in the reduced variable domain of X_{it} is a subtle issue, particularly when one notes the following facts:

- It is true in any feasible scheduling solution that the variable X_{it} may take on a value of 1 only in the range, $[ES_i,LS_i]$.
- From Eqs. 1 and 5, our choice of $t_{i,\min}$ is always smaller than or equal to ES_i.

The above facts imply that our choice of the reduced variable domain $[t_{i,\min},t_{i,\max}]$ is larger than or (at best) equal to the underlying feasible range $[ES_i,LS_i]$ of X_{it}. This may incorrectly seem to indicate that we are not performing variable domain reduction to the fullest extent. However, this is not so, because we still need to define X_{it} over the potentially larger range $[t_{i,\min},t_{i,\max}]$ so that we retain all feasible scheduling solutions to all immediate predecessors of activity (node) i. This is due to the material-balance based hold variables $H_{ii't}$ that are defined alongside the corresponding X_{it} variable in the context of every activity i when paired with each of its immediate predecessors i'. These hold variables link every activity i to all its immediate predecessors i'. The above domain reduction from $[1,H]$ (of the basic formulation) to $[t_{i,\min},t_{i,\max}]$ is also valid for the definition of hold variables $H_{ii't}$.

Following the variable reduction, the generation for the X_{it} variables is carried out for each activity (i) and these variables are maintained in a balanced binary search tree (Cormen et al., 1990) that maps the name of each activity (node) i to a time-indexed array of the corresponding variables X_{it} (see Figure 2). This is so that we have an effective $O(\log n)$ performance guarantee for accessing any variable X_{it} in the corresponding time bucket range $[t_{i,\min},t_{i,\max}]$ given the name of the activity i, and time bucket index t [$O(\log n)$ to access the corresponding variable array, followed by $O(1)$ to access any given index. This is far superior to the $O(nH)$ required for the basic approach. Following this, the variable generation for the $H_{ii't}$ variables is carried out for each activity i over the time bucket range $[t_{i,\min},t_{i,\max}]$, and paired with each of its immediate predecessors i'. A *balanced binary search tree of balanced binary search trees* (Cormen et al., 1990) is maintained in this context, where the outer tree maps the name of each activity i to an inner tree, which, in turn, maps the name of each of the corresponding immediate predecessors i' to a time-indexed array of the corresponding variables $H_{ii't}$ (see Figure 3). This is so that we have an effective $O(\log n)$ performance guarantee for accessing any variable $H_{ii't}$ in the corresponding time bucket range $[t_{i,\min},t_{i,\max}]$ given the names of activity i, its immediate predecessor i' and time bucket index t [$O(\log n)$ to access the inner tree, followed by $O(\log n)$ to access the corresponding variable array, followed by $O(1)$ to access any given index]. This again is far superior to the $O(n^2H)$ required in the basic approach. Lastly, the generation of S_{rt} slack variables is postponed to the constraint

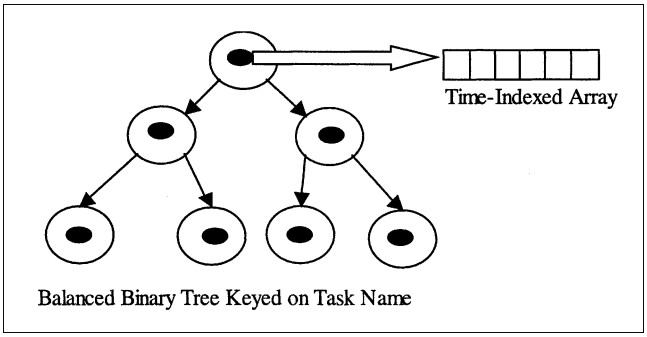

Figure 2. Data structure for X_{it} variables.

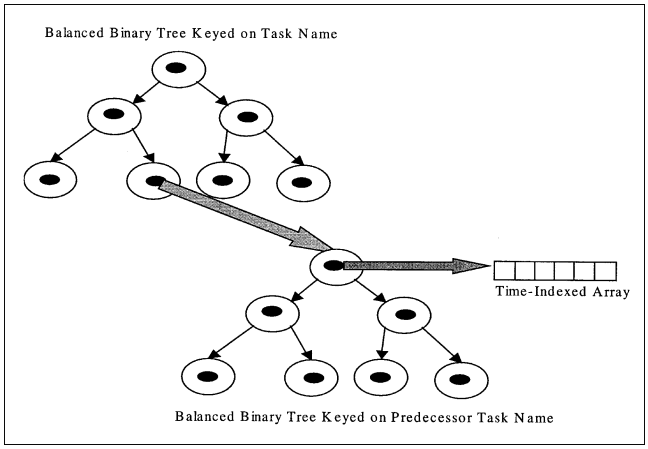

Figure 3. Data structure for the hold variables.

Table 1. Naïve vs. Improved Formulation Generation Complexity

Generation Type	Naïve Complexity	Improved Complexity
X_{it} Variables	$O(nH)$	$O(n(\max\{t_{i,\max}\} - \min\{t_{i,\min}\}))$
$H_{ii't}$ Variables	$O(n^2H)$	$O(n^2(\max\{t_{i,\max}\} - \min\{t_{i,\min}\}))$
Allocation/Demand Constraints	$O(n^2H)$	$O(n \log n)$
Material balance based precedence constraints	$O(n^4H^2)$	$O(n^2 \log n(\max\{t_{i,\max}\} - \min\{t_{i,\min}\}))$
Renewable resource constraints	$O(RH^2*(\text{Max}\{p_i\}*n^2 + R^2H^2$	$O(nR \log n(max\{t_{i,\max}\} - \min\{t_{i,\min}\})*\max\{p_i\})$
Objective Function	$O(n^2H^2)$	$O(n(\max\{t_{i,\max}\} - \min\{t_{i,\min}\}))$

generation phase as described below, so that we do not incur any time for accessing these variables [as opposed to the $O(RH)$ that was incurred in the basic approach].

The variable generation is followed by stepping over all the activities (nodes) in the AoN graphs corresponding to all candidate projects, to generate the objective function, allocation constraints, material balance based precedence constraints, and demand constraints, all of which benefit from the $O(\log n)$ time complexity guaranteed by the above data structures that contain all the X_{it} and $H_{ii't}$ variables. With respect to the generation of renewable resource constraints, it is noted that if an activity i with a processing duration of p_i number of time periods is scheduled to start in time bucket index t, it engages its respective resource types in all time buckets indexed in the range, $[t, t + p_i - 1]$. Generation of the renewable resource constraints for each resource type, within each time bucket in the UDM, requires efficient accumulation of appropriate X_{it} variables into the appropriate time buckets, along with their overbooked resource requirement coefficients of the corresponding resource type. This is done by stepping over all the activities (nodes), in the context of each resource type, and using the above data structures to efficiently access the required X_{it} variables, given the name of activity i and the appropriate time bucket index t. The generation of the resource slack variables (S_{rt}) is done at the time of the renewable resource constraint generation, for each resource type, within each applicable time bucket.

Table 1 shows a comparison of the formulation generation complexity of the naïve and the improved approaches. All the constructions shown in Table 1 represent significant improvements over the complexities exhibited by the basic approach. For example, the basic approach exhibits a quadratic complexity with respect to the number of time periods. To illustrate this improvement, Table 2 shows the basic and improved generation times for the overall formulation, the material balance based precedence constraints and the objective function for the problem instance, to be discussed later, against the number of time periods in the UDM formulation on a Pentium II 400 MHz Windows workstation. In practice, a reduction of close to 99.5% in the computational time requirement of formulation generation was observed on a Pentium III 800 MHz Windows workstation with the above data structures, over the basic implementation on problem sizes of interest to this article. An alternative data structure to store the X_{it} and $H_{ii't}$ variables could be a two-dimensional (2-D) array and a 3-D array, respectively, and this would exhibit $O(1)$ access to any desired variable. However, this would also involve an auxiliary associative array that maps the names (string literals) of activities to integer indices in the range [1, n], while also exhibiting a potentially undesirable space complexity, since both these arrays would be sparse (depending on variable reduction and the connectivity of the activity networks). Another alternative would be to use symbol table hashing (Cormen et al., 1990) with linear chaining for addressing collisions (when multiple symbols hash to the same hash value). The above discussion demonstrates the need for efficient time complexity behavior of the data structures that may be used for formulation generation in Sim-Opt.

Heuristic to generate a lower bound for the maximization MILP

Upon completing the MILP formulation generation, the generation of a lower bounding heuristic and the addition of cut families are undertaken to improve the time complexity of a formulation solution.

First, the Sim-module of Sim-Opt is used as follows to implement a heuristic that yields an integer feasible scheduling solution to generate a lower bound for the deterministic maximization MILP. The Sim-module is used to walk a *deterministic time line* that reflects the MILP formulation in terms of values for all the problem parameters. This time line involves no failure of activities, and mirrors the deterministic MILP through the use of overbooked resource requirement coefficients, and probability-weighted activity cost and reward coefficients from the deterministic MILP formulation. The policy of operation for this deterministic time line follows the use of a static resource constrained knapsack problem, as in Policy II (Subramanian et al., 2001), adapted in the following manner. At the very start of the time line, and upon the fin-

Table 2. Generation Times with Basic and Improved Implementations

	Overall Formulation (s)		Precedence Constraints (s)		Objective Function (s)	
Implementing Method of Formulation Generation	40 Time Periods	80 Time Periods	40 Time Periods	80 Time Periods	40 Time Periods	80 Time Periods
Basic	425.98	1,680.23	57.44	225.89	358.68	1,400.16
Improved	0.98	1.53	0.12	0.18	0.16	0.30

ish of any active task within the time line, a static knapsack problem is defined over the set of all activities that are technologically feasible and have not yet started. The objective function coefficient corresponding to any such activity i in the static knapsack problem is obtained as follows. The unfinished AoN sub-graph of the project corresponding to activity i is considered at the current system state, without resource constraints, and an as-soon-as-technologically-feasible schedule is computed for all activities that have not yet started in this project. The objective function coefficients corresponding to the activity starting times in the above schedule (for each activity in the schedule) are collected from the deterministic MILP formulation, and are added to produce the objective function coefficient for activity i in the knapsack formulation. The knapsack resource constraints, for every resource type, are constructed using the overbooked resource requirement coefficients of the corresponding activities from the deterministic MILP formulation. The righthand side coefficients for the resource constraints, for every resource type, are obtained by deducting the amounts engaged by partially completed (ongoing) activities, if any, at the corresponding state, from the system capacity of the corresponding resource type. This is so that ongoing activities are not upset upon resuming the deterministic time line in the Sim-module. The solution of the static knapsack is used to drive the time line ahead, and this is repeated until either the problem horizon is reached, or all activities have been scheduled within the problem horizon, which is when the deterministic time line concludes.

The schedule that emerges from the above deterministic time line is modified by the removal of activities corresponding to partially completed projects, if any, leaving the remaining schedule untouched. This modified schedule of pipeline activities is rated using the MILP formulation's objective function coefficients that correspond to the activity starting times in the schedule (for each activity in the schedule). A project-centric view of the modified schedule is taken and the contribution to the overall objective function from each individual project is computed. All the activities corresponding to projects that contribute negatively to the overall objective function are dropped from the schedule. The resulting modified schedule represents a heuristically generated integer feasible solution, and its objective function value is furnished as a lower bound for the deterministic MILP.

The above heuristic was effective in finding an integer feasible solution within a bound gap of 7% with respect to the root node linear programming relaxation of the very first MILP within a Sim-Opt time line, for the case study discussed in a later section of this article. This helps the node-pruning process of the branch-and-bound (or cut) algorithm used to solve the MILP. It led to a 70% decrease in the number of nodes processed (102 nodes vs. 331 nodes), and a 31% decrease in the number of iterations (18,818 vs. 27254) in the performance of the CPLEX solver on the largest MILP in the case study.

Relevant cut families for the MILP solution

It is observed that the allocation constraints, demand constraints, and the renewable resource constraints in the MILP formulation are knapsack-type constraints with non-negative coefficients on the lefthand and righthand sides. Knapsack cover cuts and generalized upper bound (GUB) cover cuts are two families of cuts that have been shown to work well with such knapsack-type constraints (Bixby et al., 2000). A brief discussion on the meaning of these cuts is given below.

A very detailed algorithmic treatment of how to identify cover cuts and GUB cover cuts in an efficient and effective fashion is found in Gu (1995), Gu et al. (1998, 1999). The resource constraints in the MILP formulation discussed in Subramanian et al. (2001) are the candidates for cover cuts, and the resource constraints along with the allocation and demand constraints lend themselves to a generation of GUB cover cuts. These cut families are readily available in ILOG CPLEX 7.1, and have been shown to work well with such knapsack-type constraints (Bixby et al., 2000). They have been used with aggressive cut-generation of both Knapsack cover cuts and GUB cover cuts in this study.

Without the use of these cuts or the lower bound, a conventional branch-and-bound strategy for the solution of the largest MILP (that is, the very first MILP in any time line) in the case study (to be discussed later) required upwards of 6 h (time required over and above that of formulation) on a Pentium III 800 MHz and was terminated before completion. The use of these cut families along with the heuristic lower bound reduced the cpu time requirement for the above MILP to approximately 3 min (time required over and above that of formulation), confirming the effectiveness of cuts due to the knapsack type constraint structure present in the MILP formulation.

Information Integration from the Time Lines

The time lines, which represent the various controlled futures along which the system can evolve, contain a wealth of information. This information can be used to identify improving solutions to the underlying stochastic optimization problem. The information that can be retrieved from a time line centers on the coexistence in time, of various feasible tasks, both within and across projects. This information, while being influenced by the nature of decisions that are exercised, is a function of uncertainty as well. Time lines reveal information about which constraints (on resource types) are binding at which periods in time, before we can investigate the worth of augmenting such resource types. They also contain information about the relative merits and demerits of inclusion into the portfolio for every individual project. Lastly, information can be obtained from the time lines about the effectiveness of the policy chosen for managing the portfolio, by examining and quantifying the resource crowding effect and delays that any given project can cause on the rest of the portfolio. The following sections describe two broad classes of information that can be obtained from the time lines. Methods to efficiently accumulate these two types of information and their use in obtaining improving solutions via the outer loop have also been presented.

Class One: information integration with respect to portfolio selection

In the first class of information, the portfolio is explicitly viewed as being comprised of individual projects. The perfor-

mance of every project that is introduced into the portfolio (at possibly different points in time, in the simulated time lines) by the deterministic optimizer is individually tracked across the inner loop time lines to assess its contribution to the portfolio NPV in terms of an individual NPV distribution. The corresponding project is also considered in isolation (without resource competition), and its unconstrained NPV distribution is calculated. It is important to note that the NPV distribution of any project in the portfolio is different from the NPV distribution of that same project considered in isolation, because of resource constraints and competition, and the accompanying delays in task executions of the project and the resulting shortened *patent* time window over which the project can earn commercial rewards. A comparison of these two distributions, in terms of the NPV means, reveals the impact of resource constraints on the project's performance, and presents a realistic assessment of the worth of the project in the portfolio. It also reveals projects that might be contributing negatively to the portfolio *mean* NPV. It should be recalled that the inner loop MILP makes portfolio decisions using a resource-overbooked deterministic formulation that has a limited representation of uncertainty and a fractional representation of task failure and pipeline attrition. The probabilistic (fractional) representation of failure enters into the objective function, which is expressed as the expected net present value, and also into the resource constraints by way of resource overbooking. While such fractional representations of failure yield plans and schedules that acknowledge failure in a broad decision-making sense, it should be noted that failure occurs in reality in a binary (whole, Fails/Succeeds) fashion, as opposed to something fractional. Such a realistic, binary representation to failure cannot be given in any *single* instance of a mathematical program, because of the chance-dependent nature of binary failure. These limitations can lead to portfolio selection decisions that appear attractive to the deterministic formulation at various "here-and-now" points in time, but which might contribute negatively when evaluated across the realistic time lines of Sim-Opt in the face of uncertainty.

Another piece of information that is tracked across the inner loop time lines is the probability of dropping a project at some point into the future, *after* having included it in the portfolio and *before* taking it to a conclusive finish. The deterministic MILP may introduce a project into the portfolio at a certain here-and-now decision-making point in time within a simulated time line, and incur development costs until some point in time into the future along the time line, when the corresponding *state*-defined MILP might discontinue the project. It might be unprofitable to continue with the project from the corresponding *state* of the system, under the current resource setting and patent horizon. Such probability information can be obtained from a frequency count across the inner loop time lines. The above probability information coupled with the corresponding project's individual performance in the portfolio can be used to prevent potential loss-making projects from being considered for portfolio inclusion in the "here-and-now" actions taken at various points in time.

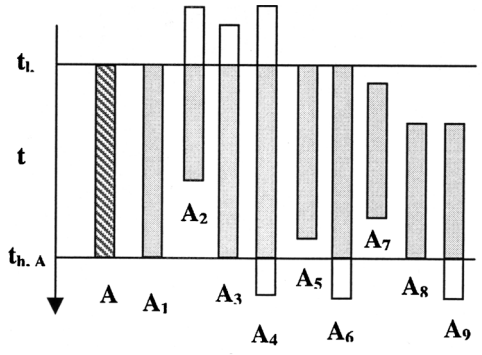

Figure 4. Coexisting tasks within a time line.

Class Two: information integration with respect to resource constraints

It was noted that, within the inner loop, the schedule resulting from the deterministic optimizer is used to assign priorities to activities in the various projects, at different points in time, with respect to engagement of respective resource types. In other words the deterministic optimizer defines the *policy* of operation, within the inner loop. Such a policy of operation may prove ineffective with respect to resource management in the face of uncertainty. This second class of information helps to systematically identify and quantify such ineffectiveness that may be introduced by the deterministic optimizer.

With respect to resource constraints within this class, there are two categories of information that can be integrated from the inner loop time lines of Sim-Opt. The first category is with respect to priority-based resource crowding (priorities being determined with the deterministic optimizer). Such crowding may be caused by higher priority activities that block out lower priority activities, while *waiting* on resources. The second category is with respect to resource crowding that may be caused by higher priority activities that block out lower priority activities, while actively executing and *engaging* resources. In the following discussion, let

- R = Set of renewable resource types r.
- $L([t_l,t_h]) = t_h - t_l$, the length of a time interval.
- $\rho_{r,A_i} \forall r \in R$, the resource needs of activity A_i of resource type r.

Figure 4 shows tasks that coexist in time, that is, tasks that have a nonzero, overlapping interval. Furthermore, in the following discussion, a task is said to be technologically feasible, if all its predecessors have successfully completed.

Category 1: Resource Crowding Caused by Activities that are waiting on Resources. Resource crowding can be caused by higher priority activities that are waiting for their resource requirements to be met. Technologically feasible lower priority activities that become resource feasible, while the higher priority activity is waiting as above, get queued in favor of the waiting higher priority activity. This is so that the higher priority activity may proceed as soon as possible. This aspect of priority-based resource crowding that is caused by higher priority activities on lower priority activities is the subject of investigation in this section.

Consider a higher priority activity (priority decided by the activity starting time resulting from the optimizer's schedule) that has become technologically feasible and is waiting for its

resource requirements to be satisfied. Let C_1 be the set of such activities A_i that can be fully characterized in terms of:

- The time interval, $[t_{sf_{A_i}}, t_{s_{A_i}}]$, where $t_{sf_{A_i}}$ denotes the point in time when activity A_i becomes technologically feasible, and $t_{s_{A_i}}$ denotes the point in time when activity A_i receives resources and starts executing.

Secondly, consider a technologically feasible, lower priority activity (priority decided by the activity starting time resulting from the optimizer's schedule) that becomes resource-feasible with respect to the free resources in the system, but does not get these free resources due to the presence of higher priority activities that are waiting for their resource needs to be fully satisfied. Let C_2 be the set of such activities A_j that can be fully characterized in terms of:

- The time interval, $[t_{rf_{A_j}}, t_{s_{A_j}}]$, where $t_{rf_{A_j}}$ denotes the point in time when the technologically feasible activity A_j becomes resource feasible, and $t_{s_{A_j}}$ denotes the point in time when activity A_j receives resources and starts executing.

Consider every pair, (A_i, A_j), that satisfies

- $A_i \in C_1$,
- $A_j \in C_2$,
- $L([t_{sf_{A_i}}, t_{s_{A_i}}] \cap [t_{rf_{A_j}}, t_{s_{A_j}}]) > 0$, and
- $t_{s_{A_i}} < t_{s_{A_j}}$.

These represent pairs of activities such that a higher priority activity A_i causes crowding for a lower priority activity A_j, while not even engaging resources. With respect to resource blocking, the pair (A_i, A_j) represents a {*blocker, blocked*} pair, where A_i can be thought of as a *blocker* and A_j can be thought of as a *blocked* activity. This crowding behavior results from relative priorities that are decided by the deterministic, expected value based, inner loop optimizer and represents ineffective resource utilization. For every such pair, the information that can be tracked across the inner loop time lines are:

- Mean duration of the above overlap, $L([t_{sf_{A_i}}, t_{s_{A_i}}] \cap [t_{rf_{A_j}}, t_{s_{A_j}}])$, averaged across the time lines: This overlap is a measure of the duration of delay caused by activity $A_i \in C_1$ on activity $A_j \in C_2$.
- Mean resource ratio of the blocked activity to the blocking activity, defined as $\gamma_{ij,r} = (\rho_{r,A_j})/(\rho_{r,A_i})$, $\forall r \in R$, if both $\rho_{r,A_j} > 0$ and $\rho_{r,A_i} > 0$: This is averaged across the time lines.
- Probability of occurrence of the pair (A_i, A_j) that satisfies the stated conditions: This is obtained as a frequency count from the inner loop time lines.

This information tells us about the tendency of any activity (that is, given a high priority by the deterministic optimizer) to cause crowding delays on other activities that could have gone ahead, if not for the prioritizing suggested by the deterministic optimizer. The information can be considered by grouping together all activities $A_j \in C_2$ that get paired with activity $A_i \in C_1$ to get the corresponding priority-based crowding tendency of any given activity $A_i \in C_1$. When considered in such a fashion, the smaller the ratio $\gamma_{ij,r}$ is with respect to unity, the higher the probability of occurrence of the pair (A_i, A_j). Also, the higher the mean duration of the overlap $L([t_{sf_{A_i}}, t_{s_{A_i}}] \cap [t_{rf_{A_j}}, t_{s_{A_j}}])$, the higher is the severity of the priority-based crowding caused by activity $A_i \in C_1$. The information can also be considered by grouping together activities $A_i \in C_1$ that get paired with activity $A_j \in C_2$ to get information about which activities cause priority-based crowding delays on any given activity $A_j \in C_2$. Information from Category 1 coupled with information integrated from the previous class with respect to portfolio selection can be used to infer a policy of operation that reflects both the above learning and the priorities decided by the deterministic optimizer. This will be illustrated in a pharmaceutical product development case study discussed later.

Category 2: Resource Crowding Caused by Activities that are engaging Resources. Consider a higher priority activity (priority decided by the activity starting time resulting from the optimizer's schedule) that is actively executing. Let C_3 be the set of such activities A_i that can be fully characterized in terms of:

- The time interval, $[t_{s_{A_i}}, t_{f_{A_i}}]$, where $t_{s_{A_i}}$ denotes the point in time when activity A_i receives resources and starts executing, and $t_{f_{A_i}}$ denotes the point in time when activity A_i finishes and disengages resources.

Secondly, consider an activity that becomes technologically feasible, but gets queued for resources due to insufficient availability. Let C_4 be the set of such activities A_j that can be fully characterized in terms of:

- The time interval, $[t_{sf_{A_j}}, t_{f_{A_j}}]$, where $t_{sf_{A_j}}$ denotes the point in time when activity A_j becomes technologically feasible, and $t_{s_{A_j}}$ denotes the point in time when activity A_j receives resources and starts executing.

Consider every pair, (A_i, A_j), that satisfies:

- $A_i \in C_3$,
- $A_j \in C_4$,
- $L([t_{s_{A_i}}, t_{f_{A_i}}] \cap [t_{sf_{A_j}}, t_{s_{A_j}}]) > 0$.

These represent pairs of activities such that an activity A_i causes crowding for a lower priority activity A_j, while actively executing and engaging resources. With respect to resource blocking, the pair (A_i, A_j) again represents a {*blocker, blocked*} pair, where A_i can be thought of as a *blocker* and A_j can be thought of a *blocked* activity. This crowding behavior results from relative priorities that are decided by the deterministic, expected value based inner loop optimizer. For every such pair, the information that is tracked across the inner loop time lines is the same as in Category 1, that is, the mean duration of the overlap (measure of delay caused by blocking on blocked), mean resource ratio of blocked to blocking activity, and the probability of the occurrence of the {blocking, blocked} pair.

This information tells us about the tendency of any activity to cause crowding delays on other activities that may have gone ahead, if not for the prioritizing suggested by the deterministic optimizer. It can be considered by grouping together all activities $A_j \in C_4$ that get paired with activity $A_i \in C_3$ to get the corresponding crowding tendency of any given activity $A_i \in C_3$. When considered along with the (unconstrained) characterization of every project in isolation (in terms of the project's unconstrained NPV distribution), this information may reveal if less promising projects (less *mean* NPV and less probability of positive NPV, when considered in isolation) have a tendency to get in the way of more promising projects, in the solutions produced by the deterministic optimizer. Such information can be used as a basis to prevent the inclusion of such projects that cause costly delays on more promising pro-

jects. The information can also be considered by grouping together activities $A_i \in C_3$ that get paired with activity $A_j \in C_4$ to get information about which activities cause crowding delays on any given activity $A_j \in C_4$. The use of information from Category 2 will be illustrated in a case study in the next section.

Pharmaceutical Case Study

The methods for information integration from the inner loop described in the last section are demonstrated on an industrially motivated pharmaceutical case study in this section. The three main stages in pharmaceutical new product development, as described in Blau et al. (2000), are Discovery, Development, and Commercial Launch. In the Discovery stage, literally thousands of molecules are applied to targets that are developed to simulate various disease groups. Once an active molecule (that is, a molecule that has a curative effect on the target) is discovered, various permutations of the structure of the molecule are tested to see if the activity can be enhanced. Testing for toxicological results in rats or mice follows, and, if no particular worrisome toxic endpoints are observed, the molecule becomes a candidate for human development. In the Development stage, significant costs are incurred on the candidate to observe its behavior in healthy volunteers, diseased patients, and, finally, in large-scale clinical studies conducted in concert with the Food and Drug Administration (FDA). Coincident with these studies, formulation and process development work are conducted to supply the drug for testing purposes, and to design and construct a commercial plant if the product is launched. If the drug is effective in the clinical studies, has no unacceptable side effects, and the FDA approves it, it moves to the Commercial Launch stage. Target markets are identified for a staged launch or "ramp-up" of the new compound. After a few years, a mature sales level is usually reached and maintained until patent coverage on the molecule expires, when competition from generics is realized. Sales are significantly diminished after expiration of the patent.

The case study comprises eleven projects with the Finish-to-Start precedence constraints within each project. The Activity-on-Node (AoN) graph for a representative project is shown in Figure 5, and this follows the simplified network flow diagram of the major activities involved in the development and commercialization of a new drug candidate, as presented in Blau et al. (2000). The activity descriptions corresponding to the nodes of the AoN graph are given in Table 3. Uncertainty with respect to task failure is considered for Phase I, Phase II, and Phase III activities. There are 14 resource types in the system, and the triangular distributions modeling the uncertainties in processing times, resource requirements, probabilities of success, and activity costs/rewards are given in Table 4 and Table 5 for one project named A. Data in an identical format for ten more projects named B through K can be found in Subramanian (2002) and Case Study (2002), and is not reproduced here due to space constraints. The capacity of the system with respect to each of the 14 resource types is given in Table 6. An annual discounting rate of 5% is used for discounted cash flow calculations, with compounding carried out on a weekly basis. The patent expiration feature is modeled as follows. In both the optimization and simulation formulations, successful projects earn recurring yearly revenues equal to their mature sales from the start of the Mature Sales activity until the expiration of the 20-year horizon, upon which time the project becomes worthless. ILOG Concert 1.1/CPLEX 7.1 (ILOG, 2002) is used for the optimization module and CSIM18 (Mesquite Software, 2002) is used for the simulation module.

It is evident from the above tables that decision-making under uncertainty in R&D pipeline management problems,

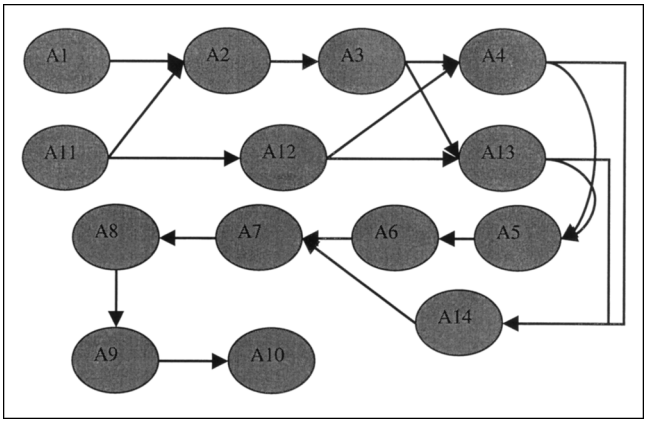

Figure 5. Case study activity-on-node graph.

Table 3. Case Study *AoN* Graph Activity Description

Activity	Activity Description
A1	First human dose preparation (FHD)
A2	Phase I
A3	Phase II
A4	Phase III
A5	FSA
A6	Prelaunch
A7	Ramp-up 1
A8	Ramp-up 2
A9	Ramp-up 3
A10	Mature sales
A11	Sample preparation
A12	Process development
A13	Design plant
A14	Build plant

Table 4. Project A: Resource Needs and Processing Time

		Resource Need Distribution, Units			Processing Time Distribution, Wks.		
Activity	Resource Type	Min.	Mode	Max.	Min.	Mode	Max.
A1	FHD	72	83	88	52	52	104
A2	Phase I	70	84	90	52	52	104
A3	Phase II	75	82	85	78	78	104
A4	Phase III	150	220	250	104	104	182
A5	FSA	18	20	22	52	52	104
A6	Prelaunch	45	52	55	26	26	26
A7	Ramp-up 1	8	12	15	52	52	52
A8	Ramp-up 2	18	22	25	52	52	52
A9	Ramp-up 3	35	42	45	52	52	52
A10	Mature sales	45	54	60	52	52	52
A11	Formulation	1.8	2	2.2	52	52	78
A12	Development	7	11	13	104	104	130
A13	Design plant	8	11	12	104	104	130
A14	Construction	45	52	55	104	104	104

Table 5. Project A: Probability, Cost, and Reward

	Probability of Success Distribution			Cost Distribution, MM Dollars			Reward Distribution, MM Dollars		
Activity	Min.	Mode	Max.	Min.	Mode	Max.	Min.	Mode	Max.
A1				72	83.2	88			
A2	0.9	0.925	0.97	70	84	90			
A3	0.3	0.325	0.37	75	82	85			
A4	0.87	0.895	0.94	150	220	250			
A5				18	20.8	22			
A6				45	52	55			
A7				8	12	15	418.75	450	487.5
A8				18	22	25	837.5	900	975
A9				35	42	45	1256.25	1,350	1,462.5
A10				45	54	60	1,675	1,800	1,950
A11				1.8	2.08	2.2			
A12				7	11.2	13			
A13				8	10.8	12			
A14				45	52	55			

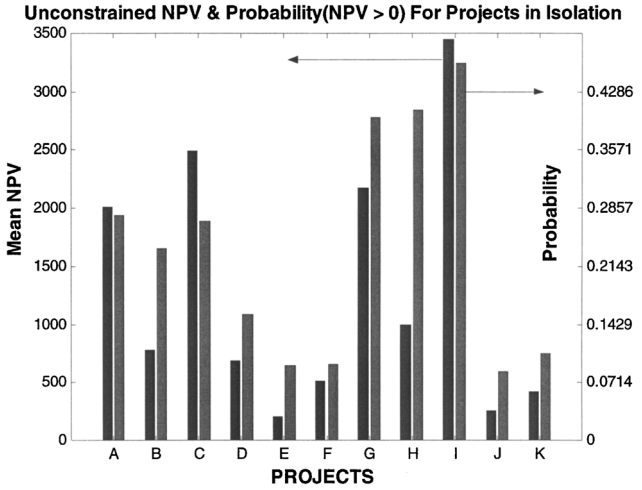

Figure 6. Unconstrained NPV and probability (NPV > 0) for projects in isolation.

such as the above instance, is a very data-intensive exercise. Such intensive data needs are modeled in an Object Oriented manner in a structured language that has been defined using the Extensible Markup Language (XML) format from the W3C Consortium (World Wide Web Consortium, 2002). The data corresponding to an instance of the R&D pipeline problem is first modeled in the XML based object oriented data model, which is then parsed and abstracted into a C++ class hierarchy comprising data structures that store all the model data and facilitate efficient construction of the state-dependent optimization and simulation formulations. A detailed discussion of the design and implementation of the XML language and the C++ class hierarchy is outside the scope of this article and can be found in Varma et al. (2002).

Information that is valuable for development pipeline management

This section discusses information useful for management of new-product development pipelines. In the context of the case study, there is a need to know which projects from the set {A, B, C, D, E, F, G, H, I, J, K} should be considered for inclusion into the portfolio, given the earning potential from the Mature Sales activity of the various projects. Given the probabilities of failure and the uncertainty present in various parameters, listed in the above tables, it is clear that this issue cannot be addressed by looking at the absolute mature sales driven reward potential alone. As a first step, the individual characterization of each of these projects is shown in Figure 6 in terms of the mean NPV and the probability of realizing a positive return on each of these projects, computed via network simulation. Since any portfolio will evolve under resource constraints within the patent interval, there is a need to address the important issue of resource allocation at various points in time. This decision-making pertaining to resource assignment to various competing project activities (that is, scheduling) at different points in time was noted as the *policy* of operation. It may often be the case that not all projects may be pursued under limited resources and yet be finished within the patent window to earn commercial returns. The questions of interest are:

- Which projects from the set {A, ..., K} should be removed from portfolio considerations?
- What should be the relative prioritization of the various project activities that are chosen in the portfolio?
- Should the policy of operation prioritize all activities of any given project, say Project X, over all other project activities, given that it may outperform all other projects with respect to both the performance measures noted in Figure 6? What effect will it have with respect to delaying activities in other projects and effectively reducing the time-window over which they may earn commercial returns? The resource requirements of Project X need to be considered vis-à-vis the other projects, for answering this question in a quantitative fashion.
- How should the relative priorities of the various projects change with time and outcome of activities (that is, with states of the pipeline)?

The above questions represent important tactical and strategic questions for the pharmaceutical pipeline management case study. Every project usually has a "project champion" in real-life pharmaceutical pipelines, and answering some of the above questions is replete with political and or-

Table 6. System Capacity

Resource Type	System Capacity
FHD	274 Units
Phase I	341 Units
Phase II	180 Units
Phase III	500 Units
FSA	97 Units
Prelaunch	548 Units
Ramp-up 1	25 Units
Ramp-up 2	50 Units
Ramp-up 3	100 Units
Mature sales	150 Units
Formulation	9 Units
Development	27 Units
Design plant	25 Units
Construction	120 Units

ganizational implications. Systematic integration of information from the Sim-Opt time lines helps answer these questions in a quantitative fashion, and, thus, may be used for objective conflict resolution. This is illustrated in the following sections.

Inner loop policy of operation

The inner loop of Sim-Opt uses the Policy I discussed in Subramanian et al. (2001). Policy I uses the deterministic MILP and the resulting scheduling solution to make both supervisory and regulatory decisions. The deterministic MILP uses a time discretization size of 26 weeks, and considers a horizon of 1,040 weeks, corresponding to the 20-year patent horizon. As noted earlier, the scheduling solution from the MILP decides the relative priorities of the scheduled tasks in the simulation. The earlier an activity is scheduled to occur, the higher is its priority in engaging the corresponding resource types. Tasks that have the same scheduled starting times are relatively prioritized in terms of the expected unconstrained NPV associated with their projects. The higher the expected NPV associated with an activity, the higher is its priority with respect to engaging resources. The expected NPV is based upon the critical path of the corresponding Activity-on-Node (AoN) subgraph that remains at the current state of the system, when considered without resource constraints. Resource starvation, occurring due to resource conflicts, is tolerated for a fixed duration of one time bucket beyond the scheduled starting times of such active tasks. If starvation continues beyond the above threshold of tolerance, it is treated as an event that needs decision-making and the current state of the system is communicated to the optimizer. The event of an unsuccessful finish of an active task is also treated as an event that needs decision-making and is handled similarly. The state-dependent MILP formulation reflects the current state of the system in terms of

- Removal of all tasks belonging to projects that have failed thus far
- Removal of tasks that have finished
- Addition of constraints to prevent the preempting of ongoing tasks, and
- Parameter updating for partially completed tasks in terms of processing times and resource requirements.

It should be noted that no constraints are added to insist on the continuation of ongoing projects, which may get suspended depending on the optimum of the corresponding state-dependent MILP.

Inner loop performance and a three-step heuristic based on information integration

The time lines of Sim-Opt yield the probability distribution of the NPV of the portfolio. The relative error criterion on the mean of the portfolio NPV distribution is used to determine the number of simulation time lines required before terminating the inner loop. The relative error in the estimate $\overline{X}(n)$ or random variable X, with true mean μ is taken to be $|\overline{X}(n)-\mu|/\overline{X}(n)$, where n is the number of observations. The specific objective of the following procedure (Law and Kelton, 2000) is to obtain an estimate of μ with a relative error of γ $(0<\gamma<1)$, and a confidence level of $100(1-\alpha)$ percent. The following steps are done starting with $n_o = 1{,}000$, and using the t-distribution based confidence-interval half-length given by

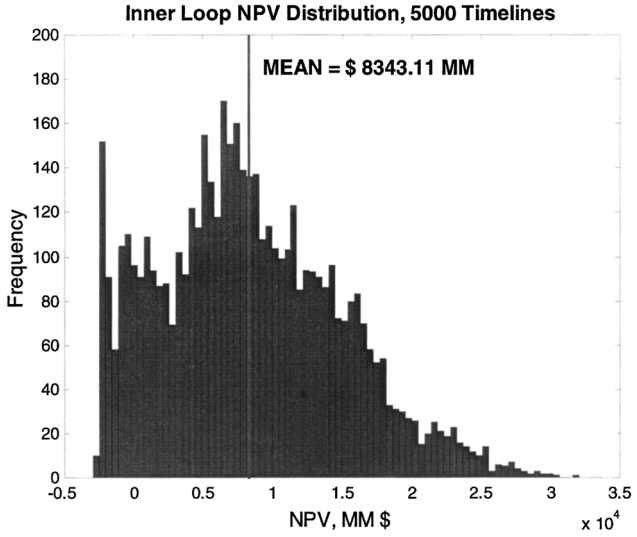

Figure 7. Inner Loop NPV distribution.

$$\Delta(n,\alpha) = t_{n-1,\alpha/2}\frac{s(n)}{\sqrt{n}} \tag{7}$$

where $t_{n-1,\alpha/2}$ is the upper $100\alpha/2$ percentage point of the t-distribution with $n-1$ degrees of freedom.

(1) Run n_o inner-loop Sim-Opt time lines, and set $n = n_o$

(2) Compute the mean of the portfolio NPV distribution, $\overline{NPV}(n)$, and $\Delta(n,\alpha)$ from NPV_1, NPV_2, ..., NPV_n.

(3) If $\Delta(n,\alpha)/|\overline{\mathrm{NPV}}(n)| \le \gamma/(1+\gamma)$, then use $\overline{\mathrm{NPV}}(n)$ as the point estimate for the true mean, and stop. Else replace n by $n+1{,}000$, make an additional 1,000 time lines and go to step 1.

Using $\gamma = 0.025$, the above procedure terminates with 5,000 time lines in the inner loop for the pharmaceutical case study. The portfolio NPV distribution resulting from the inner loop policy of operation is shown in Figure 7. The mean of the inner loop NPV distribution is 8343.11 MM $ with a maximum relative error of 0.025 and a confidence level of 95%. The $100(1-\alpha)$ percent, binomial proportion based, confidence interval half-length on the proportion of interest, which is that of realizing a positive NPV ($\mathrm{NPV} \ge 0$), is given by

$$\Delta_p(n,\alpha) = z_{\alpha/2}\sqrt{\frac{p(1-p)}{n}} \tag{8}$$

where $z_{\alpha/2}$ is the upper $\alpha/2$ percentage point of the standard normal distribution (Montgomery and Runger, 1999), and p is the proportion of time lines in the random sample of 5,000 time lines that deliver a positive NPV. The inner loop policy of operation leads to the probability of 0.886 for delivering a positive NPV, with a confidence interval half-length of 0.0088 at a 95% confidence level. The serial cpu time requirement for the 5,000 time lines on a Pentium III 800 MHz was ap-

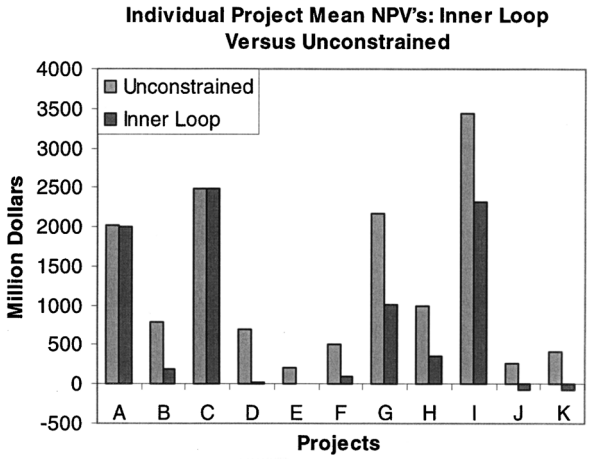

Figure 8. Individual project mean NPVs: inner loop performance and unconstrained performance.

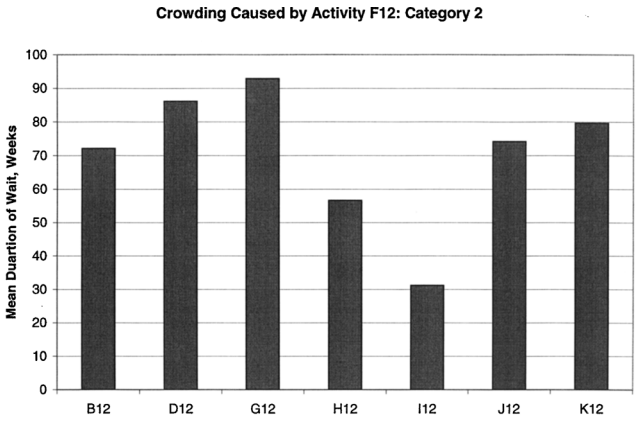

Figure 9. Category 2: crowding caused by Activity F12 from Project F: mean duration of wait.

proximately 100 hours. A heuristic is presented in the following section that attempts to integrate information from the inner loop time lines to lead to improving solutions for portfolio performance.

Heuristic Step 1. In Step 1 of the heuristic, a project-centric view of the inner loop portfolio performance is taken, and the contribution of every project to the portfolio mean is individually tracked across the 5,000 inner loop time lines. It should be noted that the contribution of each project in the resource-constrained portfolio is determined by the choice of the portfolio and *policy* of operation.

Figure 8 shows the comparison of the mean contribution of each project in the portfolio controlled by the inner loop policy of operation, against the mean unconstrained contribution of the same project considered in isolation (without resource competition). It reveals that projects J and K are contributing negatively to the portfolio *mean* NPV. The selection of these projects appears attractive to the deterministic formulation at various "here-and-now" points in time, but their negative impact on the portfolio mean performance is revealed when evaluated across the realistic inner loop time lines of Sim-Opt. Project E is never included into the portfolio by the deterministic optimizer, as it is found to be unprofitable at the given level of system capacity and its patent expiration deadline. Furthermore, Projects J and K exhibit a 2.4% and 9.2% chance of being discontinued *after* getting included into the portfolio and *before* being taken to a conclusive finish, while also contributing negatively to the portfolio mean

Step 1 excludes projects that negatively impact the portfolio mean performance, and projects that are never included (or get included with very small probabilities) into the portfolio across the inner loop time lines. As a result, Step 1 prevents the inclusion of projects E, J, and K.

Heuristic Step 2. Figure 8 also shows that Projects B, D, F, G, and H perform quite poorly when compared to their respective unconstrained performances. This is because of delays that are caused by resource constraints in task executions of these projects, and the resulting shortened *patent* time window over which the project can earn commercial rewards. In other words, this is a direct consequence of the policy of operation that is adopted in the system, and which might be ineffective. This leads into Heuristic Steps 2 and 3 that attempt to integrate information to quantify the effectiveness of deterministic scheduling and its implications for resource management in the face of uncertainty.

Figure 6 shows the unconstrained performance of each of the projects, in terms of the mean NPV and the probability of delivering a positive NPV. In the absence of Projects E, J and K that have been excluded by Step 1, the least promising project in the portfolio is Project F, when considered in isolation in terms of mean NPV and the probability of delivering a positive NPV. Step 2 investigates Category 2 {*blocker, blocked*} pairs in which activities from Project F appear as *blocker* activities. Figure 9 shows the mean duration of wait caused by activity F12 from Project F, on Process Development activities from Projects D, B, H, J, K, G, and I, by considering {*blocker, blocked*} pairs {F12, D12}, {F12, B12}, and so on, across the 5,000 inner loop time lines. This information needs to be coupled with the probability of the above pairs being realized in the system. Figure 10 shows the probability of pairs {F12, D12}, {F12, B12}, and so on, being realized at the corresponding mean duration of blocking shown in Figure 9. For instance, the Category 2 pair {F12, D12} is realized with a probability of 60.6% causing a mean blocking duration of 86 weeks, and a mean resource requirement ratio of 0.88 (resource needs of D12 over that of F12). Table 7 summarizes the corresponding information for Category 2 blocking caused by activities F1 and F3. It can be seen that Project F, the least promising project, causes blocking delays on a more promising project D, of a mean value of 86 weeks with a 60.6% chance in Process Development, a mean value of 54 weeks with a 81% chance in the First Human Dose Preparation (FHD) activity, and a mean value of 80 weeks with a 54% chance in Phase II. Project F also causes significant blocking delays on two other projects that are more promising (when evaluated in isolation), namely, projects B and H (Phase II activities) with greater than 30% and 20% probability, respectively. It can be noted from Figure 8 that Projects B, D and H suffer significantly in terms of their inner loop mean performance in the portfolio, when compared against their unconstrained potential. Heuristic Step 2 pre-

Table 7. Category 2: Crowding Caused by Activities F1 and F3 From Project F

F1 (Category 2 *Blocker*)				F3 (Category 2 *Blocker*)			
Blocked Activity	Mean Resource Ratio	Mean Duration of Wait (Wks.)	Probability of Causing the Wait	*Blocked* Activity	Mean Resource Ratio	Mean Duration of Wait (Wks.)	Probability of Causing the Wait
B1	0.86	52.8	0.25	B3	0.72	76.3	0.31
D1	0.96	54.8	0.81	D3	0.97	80.2	0.54
J1	1.60	53.1	0.31	G3	0.99	72.5	0.01
K1	1.60	57.3	0.46	H3	0.67	77.4	0.22
				I3	2.01	90.0	0.01
				J3	1.15	65.8	0.18
				K3	1.15	79.9	0.42

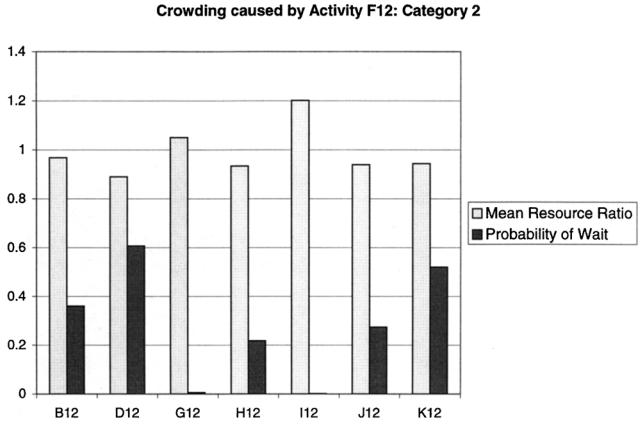

Figure 10. Category 2: Crowding caused by Activity F12 from Project F: probability of crowding and mean resource ratio.

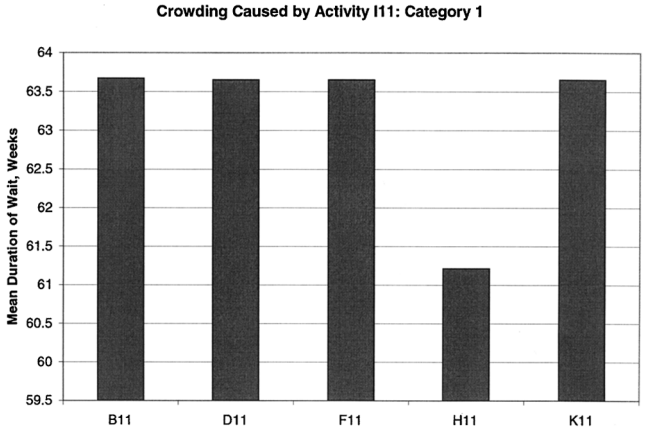

Figure 11. Category 1: crowding caused by Activity I11 from Project I: mean duration of wait.

vents from inclusion the least promising project (when evaluated in isolation) if it causes significant Category 2 blocking delays on more promising projects. As a result, Heuristic Step 2 prevents the inclusion of Project F. Steps 1 and 2, thus, reassess the decisions made by the deterministic optimizer with respect to the portfolio selection. This leads into Step 3 that investigates the need to reassess the scheduling decisions made by the deterministic optimizer with respect to resource management.

Heuristic Step 3. Step 3 looks into Category 1 resource crowding that can be caused in the face of uncertainty due to prioritizing decisions from the deterministic scheduler. This information is tracked in terms of finding {*blocker, blocked*} pairs that fit the criteria described in the section on Category 1 resource crowding. Such priority based delays caused on *blocked* activities can lead to significant degradation in the performance of their corresponding projects, as lost time is lost opportunity in the face of a patent horizon. Figure 11 shows the mean duration of such Category 1 delays caused by activity I11 from Project I, on the Sample Preparation activity in Projects B, D, and H (ignoring Project K and Project F that have been dropped by Steps 1 and 2). Figure 12 shows the probability of pairs {I11, B11}, {I11, D11} and {I11, H11} being realized as Category 1 pairs because of the inner loop policy of operation where priority is decided by the scheduling solution from the deterministic optimizer. Table 8 summarizes the same information in the context of activity I1

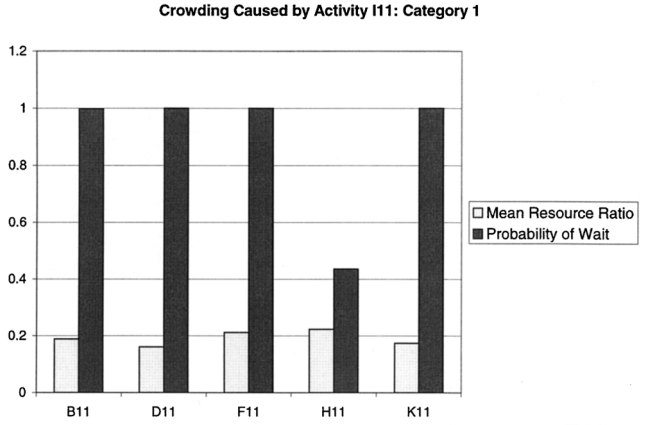

Figure 12. Category 1: crowding caused by Activity I11 from Project I: probability of crowding and mean resource ratio.

Table 8. Category 1: Crowding Caused by Activity I1 from Project I

Blocked Activity	Mean Resource Ratio	Mean Duration of Wait (Wks.)	Probability of Causing Wait
B1	0.35	75.3	0.99
D1	0.39	75.5	0.99
F1	0.40	75.5	1
G1	0.40	75.3	0.94
H1	0.38	75.3	0.99

that effectively blocks the FHD activity from Projects B, D, G and H. While it can be seen from Figure 6 that Project I is the most promising project in terms of unconstrained potential, the priority-based crowding delays that it causes on Projects B, D, G, and H with very high probability, contribute to the poor performance of these projects across the inner loop time lines, as shown in Figure 8, when compared against their unconstrained potential. This reveals the ineffectiveness of prioritizing activities with respect to resource engagement, based on the scheduling solution from the deterministic scheduler alone. Heuristic Step 3 attempts to correct this ineffectiveness by using the Sim-module to implement the following policy of operation. The Sim-module preserves the relative priorities decided by the deterministic optimizer until a Category 1 resource crowding pair arises. When such a blocking occurs, the Sim-module overrides the deterministic optimizer's solution by allowing the resource-feasible lower priority activity to go ahead and engage resources, thus overriding the higher priority *blocker* activity. While this is expected to worsen the performance of Project I, it is also expected to improve the performances of Projects B, D, G and H. It should be noted that this overriding decision made within the Sim-module would be reflected in the subsequent state-dependent MILP formulation invocations, via the states that the pipeline system would transition into by virtue of the above overriding decision. Heuristic 3 is, thus, one of several ways to reflect the integrated information with respect to Category 1 resource blocking into the MILP formulation that is used to take control actions in the pipeline system.

A summary of the three-step heuristic is as follows:

Heuristic Step 1 involves the following steps:

• Project-Centric Tracking of individual project contributions to Portfolio NPV

• Project-Centric Tracking of the probability of being discontinued before conclusive finish, after having been started

• Prevents from inclusion, those projects that contribute negatively, when assessed across the realistic time lines of Sim-Opt.

Heuristic Step 2 involves the following steps:

• Characterize every project in the portfolio, in isolation in an unconstrained mode, in terms of its mean NPV and Probability{NPV > 0}

• If the least promising project (in terms of the characterization in isolation) pairs up as a *blocker* with a more promising *blocked* project, in Category 2, with significant probability and delay duration, prevent its inclusion into the Portfolio.

Heuristic Step 3 involves the following steps:

If Category 1 resource blocking is present in the inner loop time lines with significant probability and delay duration:

• Preserve the priorities from the Deterministic Optimizer, until a Category 1 blocking arises

• When a Category 1 blocking occurs, allow the resource-feasible lower priority activity to go ahead and engage resources.

Results with Statistical Significance. The 5,000 inner loop time lines are run with Steps 1, 2, and 3 in succession, and Figure 13 shows the portfolio NPV distribution resulting from the outer loop, after application of Heuristic Steps 1, 2, and 3. The serial cpu time requirement for these steps on a Pentium III 800 MHz workstation was approximately 40 h, approximately 30h, and approximately 30 h, respectively. The

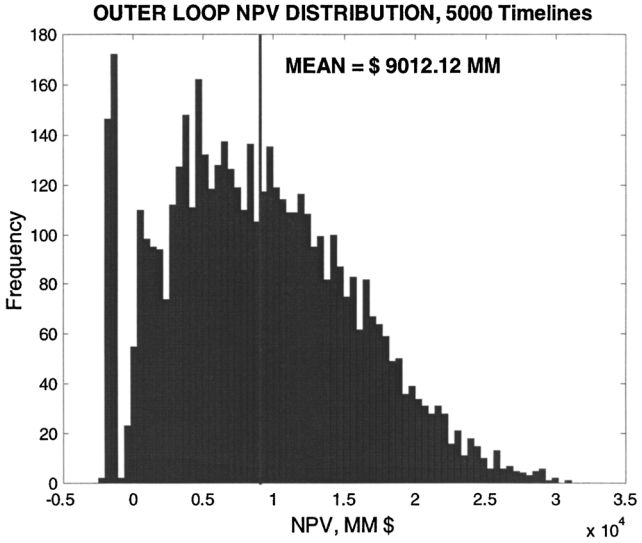

Figure 13. Outer Loop NPV distribution.

mean of the outer loop NPV distribution is 9012.12 MM $ with a maximum relative error of 0.02 and a confidence level of 95%. The outer loop also leads to a probability of 0.927 for delivering a positive NPV, with a confidence interval half-length of 0.0072 at a 95% confidence level. Figure 14 shows the improvement in the system performance in terms of both the portfolio mean NPV and the probability of the system delivering a positive NPV, from the inner loop and across Heuristic steps 1, 2 and 3. Figure 15 shows the individual performance of each project in the outer loop, compared against its inner loop and unconstrained performances. It can be seen that the outer loop three-step heuristic has led to an improvement in the performance of projects B, D, G, and H, while it has hurt the performance of Project I, while also leading to an overall improvement in terms of the portfolio performance. The statistical significance of the improvement obtained via information integration from the inner loop is quantified as follows. The $100(1\text{-}\alpha)$ percent confidence in-

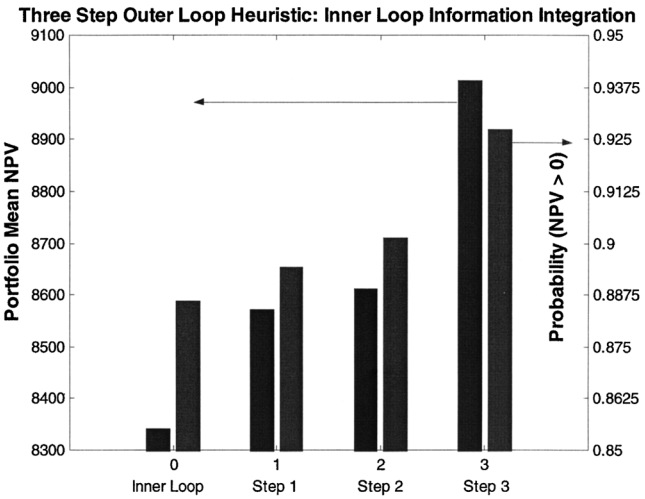

Figure 14. Three-step outer loop heuristic for information integration from inner loop.

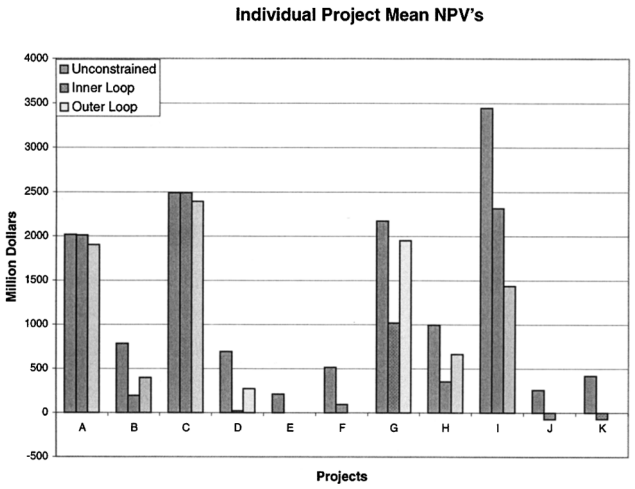

Figure 15. Individual Project Mean NPVs: unconstrained performance, inner loop performance, and outer loop performance

terval half-length on the difference between the mean NPV of the outer loop and the inner loop is given as (Montgomery and Runger, 1999)

$$\Delta_d(n_1,n_2,\alpha) = z_{\alpha/2}\sqrt{\frac{\sigma_1^2}{n_1}+\frac{\delta_2^2}{n_2}} \tag{9}$$

where n_1 and n_2 are the number of random time lines corresponding to the outer loop and inner loop, respectively; σ_1^2 and σ_2^2 are the variances for the two outer loop and inner loop distributions, assumed to be known from the corresponding sample variances. This gives a full-length confidence interval of [410 MM \$, 928 MM \$] at the 95% confidence level, on the improvement (difference) in the portfolio mean NPV delivered by the outer loop, with a mean difference of 669 MM \$. Similarly, the $100(1-\alpha)$ percent, binomial proportion based, confidence interval half-length on the difference between the outer loop and the inner loop probabilities of delivering a positive NPV, is given as (Montgomery and Runger, 1999)

$$\Delta_{d,p}(n_1,n_2,\alpha) = z_{\alpha/2}\sqrt{\frac{p_1(1-p_1)}{n_1}+\frac{p_2(1-p_2)}{n_2}} \tag{10}$$

where n_1 and n_2 are the number of random time lines corresponding to the outer loop and inner loop, respectively; p_1 and p_2 are the sample proportions of interest (NPV ≥ 0) from the outer loop and inner loop distributions, respectively. This gives a full-length confidence interval of [3.01%, 5.27%] at the 95% confidence level, on the improvement (difference) in the probability of achieving a positive NPV, delivered by the outer loop, with a mean difference of 4.14%. The above results show that information integration from the inner loop time lines that are controlled by the deterministic optimizer, when incorporated into the policy of operation via the outer loop leads to statistically significant improvements in the quality of the solutions that may be obtained for the underlying stochastic optimization problem.

Conclusions

The main conclusion of this article is that it demonstrates the benefit of explicitly viewing the R&D pipeline as the control problem of a performance-oriented, resource-constrained, stochastic, discrete-event dynamic system. Sim-Opt (Subramanian et al., 2001) has been shown as a practical approach for obtaining realistic solutions to effectively manage the pipeline system. There are two classes of information that can be obtained from the inner loop time lines, with respect to portfolio selection and resource crowding associated with the chosen policy of operation. Quantifying these helps develop intuition for designing heuristics for the system. This was demonstrated with a three-step heuristic that delivers improving solutions with respect to the mean net present value (NPV) of the portfolio and the probability of delivering a positive NPV. The use of the above method was illustrated on a case study resulting in statistically significant improvements. The integrated information from the inner loop resulted in a mean improvement of 669 MM \$ in the average NPV in the outer loop, with full-length confidence interval of improvement of [410 MM \$, 928 MM \$] at the 95% confidence level. It also resulted in a mean improvement of 4.14% in the probability of achieving a positive NPV in the outer loop, with a full-length confidence interval of improvement of [3.01%, 5.27%] at the 95% confidence level.

The second conclusion is related to basic algorithm and software engineering that is necessary for realizing the practical utility of Sim-Opt. The necessity for effective data structures for controlling the time-complexity of both formulation generation and solution has been demonstrated. Efforts undertaken to achieve significant improvements in the performance of formulation generation, and the generation of a heuristic lower bound along with identification of cut families for effective application of branch-and-cut methods for formulation solution have been described. A reduction of close to 99.5% in the computational time requirement of formulation generation of any single MILP was observed in the case study with the use of the suggested data structures on a Pentium III 800 MHz Windows workstation. The cut families along with the heuristic lower bound reduced the cpu time requirement for the solution of the largest MILP in the case study to approximately 3 min, against a conventional branch-and-bound strategy that required upwards of 6 h on a Pentium III 800 MHz. The overall serial cpu time requirement for 5,000 inner loop time lines and the information integration was approximately 100 h with the above computational improvements. Sim-Opt, as a process management tool, is applicable not only to the R&D pipeline system, but also to other stochastic process management systems, such as batch manufacturing and supply chain systems.

Literature Cited

Bixby, R. E., M. Fenelon, Z. Gu, and E. Rothberg, *System Modelling and Optimization: Methods, Theory and Applications*, M. J. D. Powell and S. Scholtes, eds., Kluwer, The Netherlands, pp. 19–49 (2000).

Blau, G. E., B. Mehta, S. Bose, J. F. Pekny, G. Sinclair, K. Kuenker, and P. Bunch, "Risk Management in the Development of New Products in Highly Regulated Industries," *Comput. and Chem. Eng.*, **24**, 659 (July 2000).

Blau, G. E., and G. Sinclair, "Account for Uncertainty in New Product Development," *Chem. Eng. Prog.*, **97**, 80 (June 2001).
Cassandras, C. G., *Discrete Event Systems: Modeling and Performance Analysis*, McGraw Hill, New York (1993).
Case Study, Available on the Web at http://atom.ecn.purdue.edu/~varmav/AicheCaseStudyData.html (2002).
Cormen, T. H., C. E. Leiserson, and R. L. Rivest, *Introduction To Algorithms*, McGraw Hill Higher Education, p. 263 (1990).
Garey, M. R., and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of Np-Completeness*, Freeman, San Francisco (1979).
Gu, Z., G. L. Nemhauser, and M. W. P. Savelsbergh, "Lifted Cover Inequalities for 0-1 Linear Programs: Computation," *INFORMS J. on Computing*, **10**, 427 (1998).
Gu, Z., G. L. Nemhauser, and M. W. P. Savelsbergh, "Lifted Cover Inequalities for 0-1 Linear Programs: Complexity," *INFORMS J. on Computing*, **11**, 117 (1999).
Gu, Z., "Lifted Cover Inequalities for 0-1 and Mixed 0-1 Integer Programs," PhD Thesis, Dept. of Industrial and Systems Engineering. Georgia Institute of Technology (1995).
Honkomp, S. J., "Solving Mathematical Programming Planning Models Subject to Stochastic Task Success," PhD Thesis, School of Chemical Engineering, Purdue University (1998).
ILOG, Available on the Web at http://www.ilog.com (2002).
Jain, V., and I. E. Grossman, "Resource-Constrained Scheduling of Tests in New Product Development," *Ind. and Eng. Chemistry Res.*, **38**(8), 3013 (Aug. 1999).
Kall, P., and S. W. Wallace, *Stochastic Programming*, Wiley, New York (1995).
Law, A. M., and D. W. Kelton, *Simulation Modeling and Analysis*, McGraw-Hill, Boston (2000).
Mesquite Software, Inc., Available on the Web at http://www.mesquite.com (2002).
Montgomery, D. C., and G. C. Runger, *Applied Statistics and Probability for Engineers*, Wiley, New York (1999).
Puterman, M. L., *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, Wiley, New York, p. 1 (1994).
Schmidt, C. W., and I. E. Grossmann, "Optimization Models for the Scheduling of Testing Tasks in New Product Development," *Ind. and Eng. Chemistry Res.*, **35**(10), 3498 (Oct., 1996).
Subramanian, D., J. F. Pekny, and G. V. Reklaitis, "A Simulation-Optimization Framework for Addressing Research and Development Pipeline Management," *AIChE J.*, **47**, 2226 (Oct., 2001).
Subramanian, D. A., "Computational Architecture to Address Combinatorial and Stochastic Aspects of Process Management Problems," PhD Thesis, School of Chemical Engineering, Purdue University (2002).
Varma, V., D. Subramanian, J. F. Pekny, and G. V. Reklaitis, "An XML-Based Language for the Research & Development Pipeline Management Problem," Technical Report, Computer Integrated Process Operations Consortium. School of Chemical Engineering, Purdue University, West Lafayette, IN (2002).
World Wide Web Consortium, Available on the Web at http://www.w3.org (2002).

Manuscript received Nov. 12, 2001, and revision received July 25, 2002.