

Resource-Constrained Scheduling of Tests in New Product Development

Vipul Jain[†] and Ignacio E. Grossmann*

Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

The problem of resource-constrained scheduling of testing tasks for new product development is addressed. The problem is important because, in some industries like pharmaceutical and agrochemicals, a new product is required to pass all the tests by federal laws. If a product fails any of the tests, then all the remaining work on that product is halted and the investment in the previous tests is wasted. Continuous time mixed-integer linear-programming models based on two different representations of resource constraints are presented to solve this scheduling problem. These models take into account complex trade-offs and have the capability of deriving a schedule that satisfies the resource constraints and utilizes the option of outsourcing. It is demonstrated with the second model, which is based on graph representation and makes use of logic, that the proper combination of modeling and search strategy can make the difference in successfully tackling this problem. Finally, it is also shown that it is critical to incorporate resource constraints along with sequencing of testing tasks to obtain a globally optimal solution.

1. Introduction

New product development usually involves a series of testing tasks (environmental, safety, etc.) prior to product commercialization. If a product fails any of the tests, then all the remaining work on that product is halted and the investment in the previous tests is wasted. Recently, Schmidt and Grossmann¹ have considered the problem of optimal *sequencing* of testing tasks for new product development, assuming that unlimited resources are available. For a product involving a set of testing tasks with given costs, durations, and probabilities of success, these authors formulated a mixed-integer linear-programming (MILP) model based on continuous time representation to determine the sequence of those testing tasks. The objective of the model in its more general form is to maximize the expected net present value (NPV) associated with a product, while a special case considers the minimization of cost subject to a time completion constraint. Even though there may be a number of new products under consideration, the assumption of unlimited resources allows the problem, with either of the two objectives, to be decomposed by each product. In the case of NPV maximization, the fundamental trade-off is between the greater sales of products from a shorter, parallel schedule and the lower expected value of the total cost from a longer, sequential schedule. In this paper, we extend the work of Schmidt and Grossmann,¹ and develop an MILP model that performs the sequencing and scheduling of testing tasks for new product development under resource constraints.

As discussed by Schmidt and Grossmann,¹ this scheduling problem appears not to have been reported previously in the literature. The only other recent work which is close in spirit to the problem is that by Honkomp et al.² These authors considered the problem of *selecting*

process development projects from a pool of projects and *scheduling* the use of limited resources for development work to maximize the expected return from research and development operations. A process development project requires a specified sequence of tasks, each of which has a probability of failure. If a task fails, then the subsequent steps of the project are abandoned and the resources can be reallocated to other projects. Honkomp et al.² developed a MILP model based on a discrete time representation that uses the idea of overbooking of resources to solve the problem. It should be noted that this method cannot be applied to the new product development problem because it assumes a fixed sequence of tasks, and furthermore it cannot handle a continuous time domain.

The scheduling problem considered in this paper has a number of features in common with typical chemical-engineering batch-scheduling problems,^{3–9} especially in the handling of resource constraints. A major issue in any scheduling model is the time domain representation.⁴ There are two main approaches: the discrete time representation and the continuous time representation. In the discrete time representation, the time horizon is divided into a number of equally sized time intervals and an activity is forced to start only at the boundaries of these intervals. In the continuous time representation, the length of the time intervals is left as a variable. The advantage of discrete time representation is that it has a tighter linear-programming (LP) relaxation. However, for an accurate solution, it may require a large number of intervals which may lead to extremely large MILPs. The other body of literature that is somewhat relevant to this work is on project scheduling, stochastic scheduling, and stochastic programming. Schmidt and Grossmann¹ have presented a good review on the related literature in the context of this problem. The main limitation of previous work in this area is that sequencing has not been considered along with resource constraints.

The objective of this paper is to propose two new MILP models for sequencing and scheduling of testing

* To whom correspondence should be addressed. Phone: 412-268-2230. Fax: 412-268-7139. E-mail: grossmann@cmu.edu.

[†] E-mail: vipul.jain@cmu.edu.

Table 1. Cost, Duration, and Probability of Success for the Four Tests

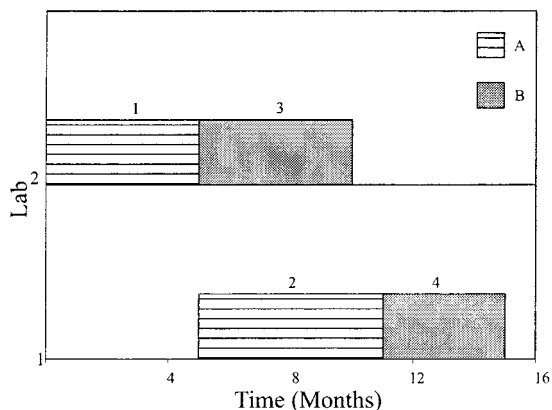
test	cost (1000 \$)	duration (months)	probability of success
1	200	5	0.60
2	150	6	0.98
3	150	5	0.80
4	150	4	0.90

tasks for new product development under resource constraints. In the next section, we present a small example problem to provide insights into the trade-offs in the scheduling problem and the effects of resource constraints. It is followed by a formal problem statement and an MILP model formulation for the problem. The resource constraints associated with the problem are modeled using the time-slot-based representation as in Pinto and Grossmann.⁴ A number of example problems are considered and computational results are presented. It is observed that computational performance of that model is unsatisfactory for larger problems. We then present a second alternative MILP model that models resource constraints as conditional arcs in a directed graph. We revisit the examples considered initially and solve it using different commercial solvers and different branching strategies. As shown by the numerical results, the second MILP model when solved using strong branching can effectively tackle problems with up to 30 testing tasks.

2. Motivating Example

Consider a trivial example problem that highlights the important issues associated with the problem discussed in this paper and the significance of resource constraints. Let us assume that a company wants to launch two products, A and B. For product certification, it is required to complete tests 1 and 2 for product A and tests 3 and 4 for product B. There is no specific sequence in which these tests should be performed. The cost, duration, and probability of success of each task are listed in Table 1.

Let us first assume that there are unlimited resources to perform these tests. Hence, the tests associated with product A can be scheduled independent of the tests associated with product B. There are three different ways to sequence the tests associated with each product (e.g., for product A: test 1 → test 2, test 2 → test 1, and parallel to each other). To compare the schedules, we need to determine the NPV of each schedule and choose the one with the highest NPV. The objective of maximizing the NPV is equivalent to minimizing the total cost which is the sum of the expected cost and the decrease in income because of delay in product commercialization.¹ The expected cost for a product is the sum of expected costs of all the associated testing tasks, which in turn are products of actual cost and probabilities of success of all the tasks preceding them. The decrease in income with time is represented by a

**Figure 1.** Scheduling after sequencing the tasks.

piecewise linear function which can be written as

$$1000t_A + 5000[\max\{t_A - 8, 0\}] + 10000[\max\{t_A - 11, 0\}] + 2000t_B + 6000[\max\{t_B - 9, 0\}] + 10000[\max\{t_B - 12, 0\}]$$

for products A and B, respectively. Here, t_A and t_B are the corresponding completion times.

The expected cost (ignoring the time value of money), completion time, income decrease, and total cost for the three possible scenarios of each product are listed in Table 2. Clearly, it is cheapest to perform test 1 before test 2 for product A and test 3 before test 4 for product B. The total cost for completing the tests for both the products using optimal sequencing and with no resource constraints is \$604 000. The MILP model developed by Schmidt and Grossmann¹ is a systematic way of determining the sequence of testing tasks that minimizes the total cost.

The above results do not account for resource constraints. It might appear that a two step procedure (strategy 1) that combines the work of Schmidt and Grossmann¹ and Honkomp et al.² may be used to perform scheduling under resource constraints. This is because a project considered by Honkomp et al.² has the same form as that of testing for new product development. The only difference is that they assume that the sequence of steps for a project is given. Therefore, if we combine the two methodologies, we can use the MILP model of Schmidt and Grossmann¹ to do the sequencing and then use the MILP model of Honkomp et al.² to do the scheduling under resource constraints.

Let us reconsider the two-product example presented earlier in this section. Let us now assume that we have only two labs available to do the testing. Furthermore, tests 1 and 3 can only be done in lab 1 and tests 2 and 4 can only be done in lab 2. A schedule derived using strategy 1 is shown in Figure 1. It uses the optimal

Table 2. Summary

product	sequence of tests	expected cost (1000 \$)	completion time (months)	income decrease (1000 \$)	total (1000 \$)
A	1 → 2	$200 + (0.6)150 = 290$	11	$(1)11 + (5)3 + (10)0 = 26$	316
	2 → 1	$150 + (0.98)200 = 346$	11	$(1)11 + (5)3 + (10)0 = 26$	372
	parallel	$200 + 150 = 350$	6	$(1)6 + (5)0 + (10)0 = 6$	356
B	3 → 4	$150 + (0.8)150 = 270$	9	$(2)9 + (6)0 + (10)0 = 18$	288
	4 → 3	$150 + (0.9)150 = 285$	9	$(2)9 + (6)0 + (10)0 = 18$	303
	parallel	$150 + 150 = 300$	5	$(2)5 + (6)0 + (10)0 = 10$	310

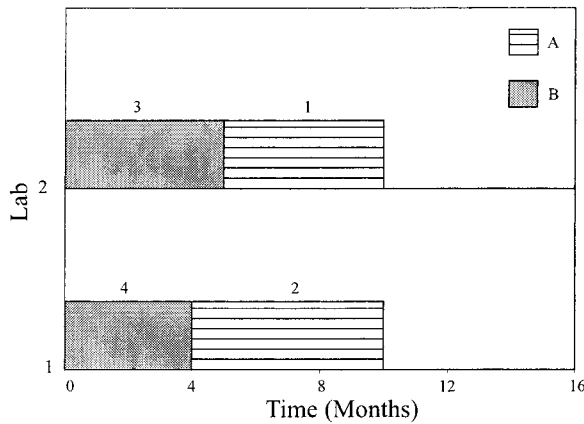


Figure 2. Scheduling without sequencing the tasks (strategy 2).

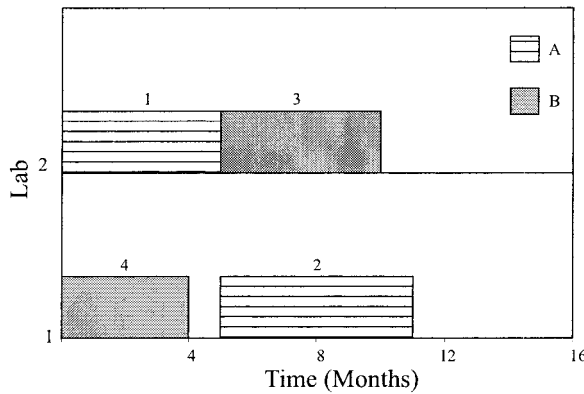


Figure 3. Simultaneous scheduling and sequencing.

sequences of testing tasks for each product and minimizes the total cost. The expected cost of all the testing tasks remains unchanged because it depends only on the sequence of testing tasks. However, the income decrease for product B rises because of the increase in the completion time. In this scenario, the total cost for each product is

$$\text{product A: } [290 + (1)11 + (5)3 + (10)0]10^3 = 316\ 000 \text{ (same as before)}$$

$$\text{product B: } [270 + (2)15 + (6)6 + (10)3]10^3 = 366\ 000 \text{ (increase of \$78 000)}$$

Hence, the total cost for the testing of both the products under resource constraints is then \$682 000.

One could argue an alternative approach (strategy 2) that can avoid the income decrease due to delay in product commercialization by ignoring the step involving sequencing of tasks for each product and derive a shorter parallel schedule such as that in Honkomp et al.² The optimal schedule obtained with strategy 2 is shown in Figure 2. The completion times for product B and product A are 5 and 10 months, respectively. However, in this case, the expected cost for completing tests is much higher because we did not exploit the fact that if a task fails, then we do not have to perform the subsequent tasks. The total cost for each product is

$$\text{product A: } [350 + (1)10 + (5)2 + (10)0]10^3 = 370\ 000$$

$$\text{product B: } [300 + (2)5 + (2)0 + (10)0]10^3 = 310\ 000$$

The total cost for the testing of both the products is \$680 000.

Although the solutions obtained using strategies 1 and 2 appear to be reasonable, the optimal schedule can only be obtained by considering resource constraints and sequencing of tasks for all the products, *simultaneously*. The optimal schedule, as shown in Figure 3, requires in fact test 1 to be performed before test 2 and test 4 before test 3. The completion times for product B and product A are 10 and 11 months, respectively. The total cost for each product is

$$\text{product A: } [290 + (1)11 + (5)3 + (10)0]10^3 = 316\ 000$$

$$\text{product B: } [285 + (2)11 + (2)2 + (10)0]10^3 = 311\ 000$$

The total cost for testing both products is \$627 000. This implies that the total cost is 8.8% higher in the case when sequencing is done prior to scheduling (strategy 1) and it is 8.4% higher when no sequencing is done prior to scheduling (strategy 2). The schedules are not optimal because, with strategy 1, there is a delay in product commercialization and, with strategy 2, the stochastic nature of the problem is not exploited while deriving the schedule.

This example clearly demonstrates the need to simultaneously incorporate resource constraints with the sequencing of testing tasks. The aim of this work is to develop an effective MILP model to accomplish this goal.

3. Problem Statement

Given is a set of potential products, each of which must undergo a set of testing tasks. The set of products is denoted by L and the set of testing tasks corresponding to a product $l \in L$ is denoted by I^l . Each task $i \in I$, where $I = \cup_{l \in L} I^l$ and $\cap_{l \in L} I^l = \phi$, has an associated duration d_i , cost c_i , and probability of success p_i . Some of the testing tasks for a potential new product may have technological precedence constraints. Let A represent the set of technological precedences, where $(i, j) \in A$ means that task i must precede task j . Furthermore, only limited resources are available to complete the tasks and they are divided into different resource categories. A set of resources in a particular category r is denoted by \mathcal{J}_r , and N_{ir} units of resources are needed from resource category r to complete test i . If desired, an individual test i may be outsourced at a higher cost \bar{c}_i , and in that case none of the internal resources are used. Finally, given the income for each product as a function of the time of product introduction, the aim is to determine a testing schedule that maximizes the NPV and meets the resource constraints.

An example problem involving two products is given in Figure 4. Here, product A requires 5 tests (1–5) and product B requires 6 tests (6–11). The precedence between the different tasks of a potential product is represented using the arrows. The resources available include 6 technicians and 4 laboratories, and testing tasks may need resources from one or both the resource categories.

It is assumed that the resources are discrete in nature (e.g., number of technicians) and that they can handle only one task at any instance. These assumptions, however, do not restrict the modeling capability. Since continuous resources can be modeled as discrete batches,

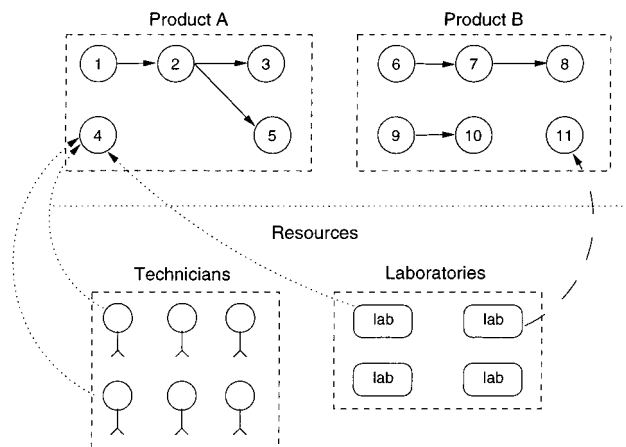


Figure 4. An example problem with two potential products.

and if a resource can handle more than one task, it can be decomposed into multiple resources of unit capacity each. This however might increase the size of the MILP model. Furthermore, it should be noted that because of the resource limitations, the schedule maximizing the total NPV is not equivalent to schedules maximizing separately the NPV of each individual product.

One of the critical attributes of this problem is to reschedule the use of all the resources when a product fails a given test. Recall that each task i has a certain probability of failure $1 - p_i$. If any task fails, then the entire testing of the product is cancelled and subsequently scheduled tasks are not performed. This results in freeing up a resource for other tasks. A strategy based on *overbooking of resources* has been proposed by Honkomp et al.² to take into account this attribute of the problem. In this strategy, the resource constraints are enforced on the *expected use* of all the resources. The expected resource requirements for a particular task are obtained by multiplying the probability of executing the task with the resource requirements for that task. Note that, for such a schedule, available resources are not likely to be sufficient to complete all the tests in the calculated time horizon if none of the task fails. Nevertheless, this idea can be useful if the duration of the tasks are very similar. This is because the authors had used discrete time representation, and if the durations are similar, then it is very easy to readjust the resource requirement for all the discrete time intervals. However, if this is not the case, rescheduling is needed whenever a test is completed, irrespective of the outcome of the test.

In this paper, we use an alternative strategy based on *expected cost and the option of outsourcing a test*. In this strategy, instead of enforcing the resource constraints on *expected* resource requirements, they are enforced on *exact* resource requirements. The objective is to minimize the total cost, which is calculated using the expected cost of each test. Recall that the expected cost of a test depends on the probability of conducting that test. If a test with high probability of failure is scheduled earlier, then it will reduce the expected cost of all the subsequent tests. This in turn may result in more tests to be outsourced as the *expected* outsourcing cost may not be as high as the *actual* outsourcing cost due to the lower probability of conducting the tests. One major advantage of this strategy over the one by Honkomp et al.² is that rescheduling is needed only when a product *fails* a test. When no product fails a test, the schedule is always feasible. Furthermore, as shown

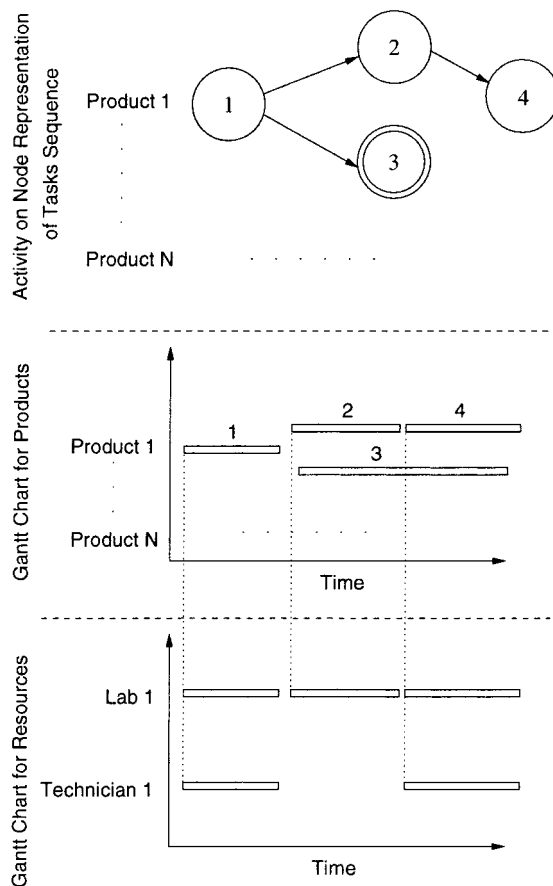


Figure 5. Representation of solution.

later in this paper, this strategy can be effectively used for developing scheduling models that are based on continuous time domain representation. One other application of this strategy is for the case when a time limit is specified on the length of the schedule. If outsourcing is available, feasible solutions are, in principle, always possible to obtain. However, if the option of outsourcing is not available, one can still consider artificial outsourcing at a very high cost to derive an initial schedule, which is then updated anytime a product fails a test. In this case, the proposed strategy is, in principle, the same as the strategy of overbooking of resources.

4. Representation

Any solution to the problem stated in the previous section can be represented using a directed graph and two Gantt charts. The directed graph, similar to the one used by Schmidt and Grossmann,¹ is an activity on node representation of tasks and is used to display the optimal sequence of tasks. Note that for the sake of clarity we will not show transitive arcs in the graph. This graph also displays all the tests that have been outsourced (double circles). The first Gantt chart is from the perspective of products and is used to present the durations and start times of all the tasks associated with a product. The second Gantt chart is from the perspective of resources and is used to display the use of each resource over the period of time. As an example, consider a case where one of the products requires four tests and resources from two categories (laboratories and technicians). Using the proposed representation, the solution to the problem may resemble Figure 5. Note

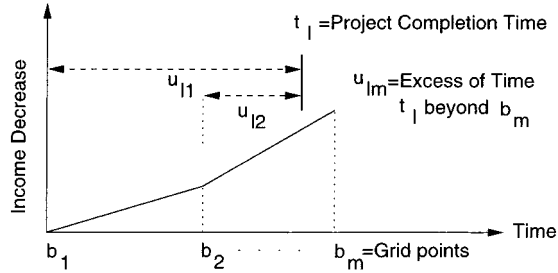


Figure 6. Variables for a modeling decrease in income.

that test 3 is outsourced and does not use any of the available resources.

There are three main decisions associated with the solution of the problem: first, the start time of task i , which is represented by s_i ; second, the sequencing relation between two tasks, i and i' , which is denoted by $y_{i'i}$ and $y_{i'}$. (Note that $y_{i'i}$ and $y_{i'}$ are binary variables and if $y_{i'i}$ is 1, it implies that task i' is executed after task i finishes. Conversely, if $y_{i'}$ is 1, then it implies that task i is executed after task i' finishes. If both of them are zero, then the start and/or end times of both the tasks may overlap.); third, the decision regarding the outsourcing of a task, which is denoted by the binary variable \hat{z}_i (If \hat{z}_i is 1, then the task is outsourced; otherwise, it is completed using the available resources.) To obtain a solution of this form for the problem stated in the previous section, we now present a mixed-integer linear-programming (MILP) model.

5. MILP Model

The problem at hand can be modeled as a mathematical programming problem that corresponds to an MILP model. The objective function and constraints for the problem are as follows.

5.1. Objective Function. The objective of this problem is to maximize the NPV, which in turn is equivalent to minimizing the sum of the expected cost and the decrease in income because of delay in product commercialization.¹ The expected cost of a testing task is the cost of completing a test that is adjusted for the time value of money and stochastic nature of the task. Let us denote the expected cost of the task i by C_i . The decrease in income because of delay in commercialization of product l , as shown in Figure 6, is a piecewise linear function of completion time of all the required tests, t_l . The function is the same as the one used by Schmidt and Grossmann.¹ The piecewise linear segments are defined at fixed discrete times b_m , where m is the index of the selected times. The objective function can be written as

$$\min \sum_{i \in I} C_i + \sum_{l \in L} \sum_{m \in M_l} f_{lm} u_{lm} \quad (1)$$

Here, u_{lm} is a nonnegative variable denoting the excess of completion time of product l , t_l , over b_m , if any. It can be evaluated using the following constraint:

$$u_{lm} \geq t_l - b_m \quad \forall (l \in L), (m \in M_l) \quad (2)$$

It should be noted that if t_l is less than b_{lm} , then this inequality is redundant because u_{lm} is a nonnegative variable; otherwise, this inequality will be satisfied at equality because of the nature of the objective function.

The parameter f_{lm} gives the marginal decrease in income because of t_l exceeding time b_m for product l .

The expected cost of completing test i is a function of an outsourcing decision, probability of conducting the test, and time value of money. It can be calculated using the following disjunction:

$$\left(C_i = c_i e^{-\bar{r}s_i} \prod_{i' \neq i, i' \in I_i} p_{i'} y_{i'i} \right) \vee \left(C_i = \bar{c}_i e^{-\bar{r}s_i} \prod_{i' \neq i, i' \in I_i} p_{i'} y_{i'i} \right) \quad (3)$$

where the binary variable \hat{z}_i denotes whether or not task i is outsourced. The first term in disjunction (3) models the case of no outsourcing ($-\hat{z}_i$) and the second term models the case of outsourcing (\hat{z}_i). Recall that c_i is the cost of conducting test i using available resources and \bar{c}_i is the cost of conducting test i if it is outsourced. Also, \bar{r} is the interest rate compounded continuously for the investment with similar risk, s_i is the start time of task i , and I_i is the product that requires test i . The expected cost function is very similar to the one used by Schmidt and Grossmann.¹ However, here the option of outsourcing is also included while evaluating the expected cost. Note that the expected cost is a disjunction over nonlinear functions. However, it can be linearized by first using the logarithmic transformation and approximating the exponential function using a piecewise linear function and then using the convex hull formulation. It was shown by Schmidt and Grossmann¹ that approximating the exponential function by a piecewise linear function introduces very little error in the value of the objective function. First, using the logarithmic transformation, we rewrite eq 3 as

$$\left(C_i = c_i e^{w_i} \right) \vee \left(C_i = \bar{c}_i e^{w_i} \right) \quad \forall (i \in I) \quad (4)$$

where

$$w_i = -\bar{r}s_i + \sum_{i' \neq i, i' \in I_i} \ln(p_{i'}) y_{i'i} \quad \forall (i \in I) \quad (5)$$

By approximating the exponential function using a piecewise function we can rewrite eq 4 as

$$\left(\begin{array}{l} -\hat{z}_i \\ C_i = c_i \left(\sum_n e^{a_{in} \hat{\lambda}_{in}} \right) \\ w_i = \sum_n a_{in} \hat{\lambda}_{in} \\ \sum_n \hat{\lambda}_{in} = 1 \\ \hat{\lambda}_{in} \geq 0 \quad \forall n \in N_i \end{array} \right) \vee \left(\begin{array}{l} \hat{z}_i \\ C_i = \bar{c}_i \left(\sum_n e^{a_{in} \hat{\lambda}_{in}} \right) \\ w_i = \sum_n a_{in} \hat{\lambda}_{in} \\ \sum_n \hat{\lambda}_{in} = 1 \\ \hat{\lambda}_{in} \geq 0 \quad \forall n \in N_i \end{array} \right) \quad \forall (i \in I) \quad (6)$$

Here, n is the index for grid points, a_{in} are the grid points, $\hat{\lambda}_{in}$ are the linearization variables, and N_i is the set of indices for grid points. Finally, as shown in Appendix A, by using the convex hull formulation and

by eliminating variables, disjunction (6) can be expressed in mixed-integer form as follows:

$$C_i = c_i \left(\sum_n e^{a_{in}} \lambda_{in} \right) + \bar{c}_i \left(\sum_n e^{a_{in}} \bar{\lambda}_{in} \right) \quad \forall (i \in I) \quad (7)$$

$$w_i = \sum_n a_{in} (\lambda_{in} + \bar{\lambda}_{in}) \quad \forall (i \in I) \quad (8)$$

$$\sum_n \lambda_{in} = (1 - \hat{z}_i) \quad \forall (i \in I) \quad (9)$$

$$\sum_n \bar{\lambda}_{in} = \hat{z}_i \quad \forall (i \in I) \quad (10)$$

$$\lambda_{in}, \bar{\lambda}_{in} \geq 0 \quad \forall (i \in I), (n \in N) \quad (11)$$

It should be noted that if W_i is the lower bound on w_i , then grid points a_{in} lie in the range $[W_i, 0]$. Equations 5 and 7–11 are the equivalent linearized constraints to evaluate the value of expected cost.

5.2. Timing Constraints. All the constraints in this category relate the start time of various tasks and the completion time of a product. Recall that the decrease in income because of delay in commercialization of product l is based on its overall completion, t_l . Clearly, t_l must be greater than the completion time of each task i required for product l ,

$$s_i + d_i \leq t_l \quad \forall (l \in L), (i \in I^l) \quad (12)$$

Furthermore, the start time of a task depends on the start time of all the tasks that have been scheduled to finish before it. If U_i is the upper bound on the start time of task i , then

$$s_i + d_i \leq s_{i'} + U_i(1 - y_{ii'}) \quad \forall (i \in I), (i' \neq i), (i' \in I^i) \quad (13)$$

Note that this is a big-M constraint and is only enforced if task i' is executed after i , that is, if $y_{ii'} = 1$. As mentioned in the problem statement, some tasks may have technological precedence. This can be easily enforced by fixing the following variables:

$$y_{ii} = 1, \quad y_{i'i} = 0 \quad \forall (i, i') \in A \quad (14)$$

5.3. Logic Cuts. All the constraints in this category are exactly the same as in Schmidt and Grossmann.¹ Although these constraints are not essential, including them in the model speeds up the search procedure. The constraints are based on the fact that any solution to the problem will never have a directed cycle in the activity-on-node representation (Figure 5) of the solution. The following constraints eliminate directed cycles of lengths 2 and 3, respectively:

$$y_{ii} + y_{i'i} \leq 1 \quad \forall (i \in I), (i' \in I^i), i' > i \quad (15)$$

$$y_{ii'} + y_{i'i''} + y_{i''i} \leq 2 \quad \forall (i \in I), (i' \in I^i), (i'' \in I^{i'}), i'' > i' > i \quad (16)$$

$$y_{i'i} + y_{i'i''} + y_{i''i} \leq 2 \quad \forall (i \in I), (i' \in I^i), (i'' \in I^{i'}), i'' > i' > i \quad (17)$$

5.4. Resource Constraints. The constraints presented so far are sufficient to solve the problem without

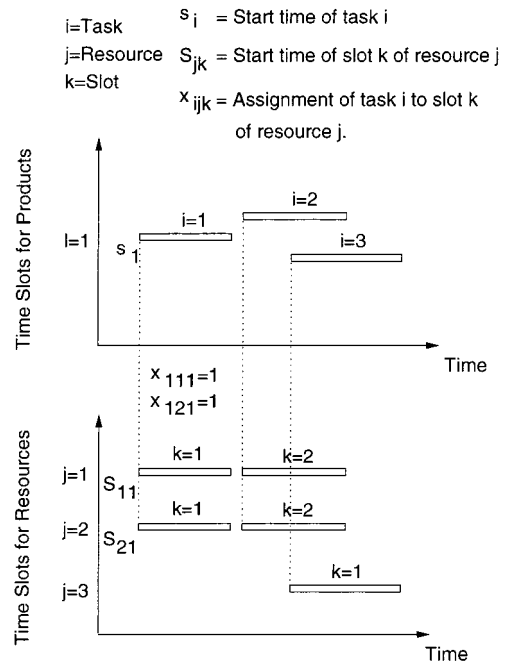


Figure 7. Time slots for resource constraints.

resource constraints. However, additional constraints are needed to model the resource limitations. The first approach used in this paper is similar to the one proposed by Pinto and Grossmann⁴ for short-term scheduling of batch plants. These authors used the idea of time slots to model the assignment of an order to a process unit. They also assumed that only one unit is needed to complete each step in the manufacturing process. The same authors, Pinto and Grossmann,⁵ later presented another model which took into account additional resource constraints such as labor and electricity. The extra resource constraints were handled using a logic-based algorithm. The algorithm proved to be very effective because the extra resource constraints had often few violations in the schedule obtained initially. A general resource task network (RTN) representation, in which all the resources are treated uniformly, has been proposed by Pantelides.¹⁰ In this paper, we extend the slot-based model of Pinto and Grossmann⁴ to a general case in which a task may require multiple resources from one or more resource categories. In the proposed model, all the resources are treated uniformly as in Pantelides.¹⁰ It is interesting to note that, for a fixed sequence of tasks, the problem reduces to a short-term batch-scheduling problem.

The central idea for modeling resource constraints in this paper is the representation of time. Two different time coordinates are used to handle the assignment of resources to a task. The first time coordinate is from the perspective of a task. It is represented by a continuous variable s_i that has already been defined. The second time coordinate is from the perspective of a resource and is represented through time slots. A time slot represents a possible use of a resource for a task. The variable S_{jk} is used to denote the start time of slot k on resource j . If slot k of resource j is used for task i , then the length of the time slot is equal to the duration of task i . The binary variable x_{ijk} is used to denote this decision. It is 1 when slot k of resource j is assigned to task i , and zero otherwise. Furthermore, when an assignment is made, the two different time coordinates are mapped on to each other. Figure 7 illustrates the idea of time

slots and associated variables. The inequalities required to model the resource constraints using this idea can be divided into three different subcategories.

5.4.1. Assignment Constraints. The first constraint in this category ensures that if a task is not outsourced ($\hat{z}_i = 0$), then it is allocated the required number of resources from each category.

$$\sum_{j \in (\mathcal{J}^r \cap \mathcal{J})} \hat{x}_{ij} = N_{ir}(1 - \hat{z}_i) \quad \forall (i \in \mathcal{I}), (r \in \mathcal{R}) \quad (18)$$

Here, \hat{x}_{ij} is a binary variable that is 1 if resource j is assigned to task i , and zero otherwise. Also, \mathcal{J}^r is the set of resources that can be used to complete task i , \mathcal{R} is the set of resource categories, and \mathcal{J} is the set of resources belonging to resource category r . Recall that N_{ir} is the number of resources required by task i from category r . The second constraint ensures that if a particular resource is allocated to a task, then exactly one slot of that resource is assigned to that task.

$$\hat{x}_{ij} = \sum_{k \in K_j} x_{ijk} \quad \forall (i \in \mathcal{I}), (j \in \mathcal{J}) \quad (19)$$

Here, K_j is the set of slots defined for resource j . In principle, $|K_j|$ is equal to the number of tasks that can be assigned to that resource. However, if there are multiple resources in each category, then the size of the model can be reduced by assuming fewer slots. The next constraint ensures that only one task is assigned to each slot of every resource.

$$\sum_{i \in \mathcal{I}} x_{ijk} + z_{jk} = 1 \quad \forall (j \in \mathcal{J}), (k \in K_j) \quad (20)$$

Here, z_{jk} is a slack variable that is 1 if a slot is empty, and zero otherwise. Furthermore, \mathcal{I} is the set of tests that can potentially be assigned to resource j . The next constraint is not necessary. However, it reduces the degeneracy in the model by enforcing that an earlier slot be used first.

$$z_{jk} \leq z_{j,k+1} \quad \forall (j \in \mathcal{J}), (k \in K_j \setminus \{k_j^f\}) \quad (21)$$

Here, k_j^f is the final slot for resource j .

5.4.2. Timing Constraints. This constraint ensures that if task i is assigned to slot k of resource j , then the length of slot k is at least equal to the duration of task i .

$$S_{jk} + \sum_{i \in \mathcal{I}} d_i x_{ijk} \leq S_{j,k+1} \quad \forall (j \in \mathcal{J}), (k \in K_j \setminus \{k_j^f\}) \quad (22)$$

5.4.3. Time-Matching Constraints. If task i is assigned to slot k of resource j , then time-matching constraints ensure that the start time of task i is the same as that of slot k of unit j . The following two Big-M inequalities are sufficient to enforce this constraint.

$$S_{jk} - s_i \geq -U_i(1 - x_{ijk}) \quad \forall (j \in \mathcal{J}), (k \in K_j), (i \in \mathcal{I}) \quad (23)$$

$$S_{jk} - s_i \leq U_i(1 - x_{ijk}) \quad \forall (j \in \mathcal{J}), (k \in K_j), (i \in \mathcal{I}) \quad (24)$$

Alternatively, these time-matching constraints can also

be written as follows:

$$s_i = \sum_{k \in K_j} \tau_{ijk} + \beta_{ij} \quad \forall (i \in \mathcal{I}), (j \in \mathcal{J}) \quad (25)$$

$$S_{jk} = \sum_{i \in \mathcal{I}} \tau_{ijk} + \gamma_{jk} \quad \forall (j \in \mathcal{J}), (k \in K_j) \quad (26)$$

$$0 \leq \tau_{ijk} \leq U_i x_{ijk} \quad \forall (i \in \mathcal{I}), (j \in \mathcal{J}), (k \in K_j) \quad (27)$$

$$0 \leq \beta_{ij} \leq U_i(1 - \hat{x}_{ij}) \quad \forall (i \in \mathcal{I}), (j \in \mathcal{J}) \quad (28)$$

$$0 \leq \gamma_{jk} \leq U_j z_{jk} \quad \forall (j \in \mathcal{J}), (k \in K_j) \quad (29)$$

These constraints are derived from alternative nonlinear time-matching constraints by following a reasoning similar to the one presented by Pinto and Grossmann.⁴ The derivation of these constraints is presented in Appendix B. Here, τ_{ijk} , β_{ij} , and γ_{jk} are extra variables required for reformulation. Furthermore, U_i is the upper bound on the start time of task i and U_j is the upper bound on the start time of slots defined for unit j .

5.5. Bounds. The upper bound (U_i) on the start time of task i and the upper bound (U_j) on the start time of slots defined for unit j can be calculated as follows:

$$U_i = \sum_{l \in \mathcal{I}, l \neq i} d_l \quad \forall (i \in \mathcal{I}) \quad (30)$$

$$U_j = \sum_{i \in \mathcal{I}} d_i - \min_{i \in \mathcal{I}} \{d_i\} \quad \forall (j \in \mathcal{J}) \quad (31)$$

The lower bound (W_i) on the variable w_i can be calculated using eq 5 as follows:

$$W_i = -\bar{r}U_i + \sum_{l \neq i, l \in \mathcal{I}^i} \ln(p_l) \quad \forall (i \in \mathcal{I}) \quad (32)$$

Bounds on all the variables involved can be summarized as follows:

$$0 \leq s_i \leq U_i \quad \forall (i \in \mathcal{I}) \quad (33)$$

$$0 \leq S_{jk} \leq U_j \quad \forall (j \in \mathcal{J}) \quad (34)$$

$$W_i \leq w_i \leq 0 \quad \forall (i \in \mathcal{I}) \quad (35)$$

$$C_i \geq 0 \quad \forall (i \in \mathcal{I}) \quad (36)$$

$$0 \leq t_l \leq \sum_{i \in \mathcal{I}} d_i \quad \forall (l \in \mathcal{L}) \quad (37)$$

$$u_{lm} \geq 0 \quad \forall (l \in \mathcal{L}), (m \in \mathcal{M}) \quad (38)$$

$$\hat{x}_{ij} \in \{0, 1\} \quad \forall (i \in \mathcal{I}), (j \in \mathcal{J}) \quad (39)$$

$$x_{ijk} \in \{0, 1\} \quad \forall (i \in \mathcal{I}), (j \in \mathcal{J}), (k \in K_j) \quad (40)$$

$$y_{i\ell} \in \{0, 1\} \quad \forall (i, \ell \in \mathcal{I}), i \neq \ell \quad (41)$$

$$\hat{z}_i \in \{0, 1\} \quad \forall (i \in \mathcal{I}) \quad (42)$$

$$z_{jk} \in \{0, 1\} \quad \forall (j \in \mathcal{J}), (k \in K_j) \quad (43)$$

The MILP model M1 comprising eq 1 as the objective and eqs 2, 5, 7–22, and 25–43 as constraints can be used to solve the problem at hand. This model is very general because it takes into consideration the option of outsourcing a testing task. Notice that if the option of outsourcing is not available, then the desired model

Table 3. Problem Data

product	test	cost (10000 \$)	cost of outsourcing (10000 \$)	duration (days)	probability of success	resource requirement
P1	1	8	16	150	1	lab 1
	2	8	16	100	1	lab 2
	3	5	10	120	1	lab 3
	4	1	2	10	0.84	lab 4
	5	49	98	90	0.98	lab 1
	6	111	222	180	1	lab 2
	7	6	12	30	0.95	lab 3
	8	174	348	200	1	lab 4
	9	62	124	270	1	lab 1
	10	1	2	20	1	lab 2
P2	11	16	32	90	1	lab 1
	12	113	226	15	0.87	lab 2
	13	1	2	5	0.91	lab 3
	14	13	26	90	1	lab 4
	15	53	106	60	1	lab 1
	16	9	18	30	1	lab 2
	17	117	234	90	1	lab 3
	18	40	80	90	1	lab 4
	19	57	114	150	1	lab 3
	20	23	46	90	1	lab 4
P3	21	10	20	30	1	lab 2
	22	15	30	25	1	lab 2
	23	6	12	60	1	lab 3
	24	1	2	10	0.84	lab 4
	25	92	184	40	1	lab 1
	26	46	92	5	1	lab 2
	27	1	2	20	0.95	lab 3
	28	38	76	120	1	lab 4
	29	1	2	60	0.94	lab 1
	30	42	84	15	1	lab 4

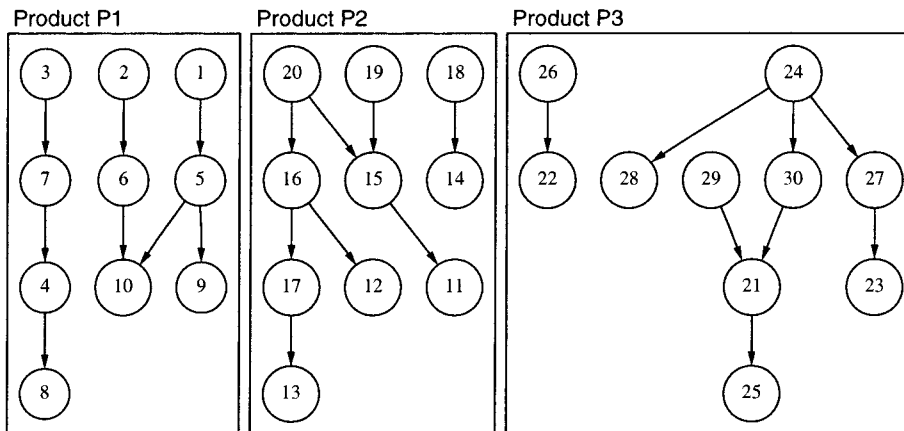


Figure 8. Technological precedence for each product.

can be obtained by just fixing outsourcing variables z_i to zero. In the next section, we consider several numerical examples to study the effectiveness of the proposed model.

6. Numerical Examples

Consider three different products that require 10 tests each. Let us denote these 3 products by P1, P2, and P3, respectively. Product P1 requires tests 1–10, product P2 requires tests 11–20, and product P3 requires tests 21–30. There are 4 different resource categories each corresponding to a particular class of laboratories. Let us denote these resource categories by lab 1, lab 2, lab 3, and lab 4. Furthermore, there is 1 laboratory available in each category. Therefore, each of the resources can be referred to by the name of the resource category itself. The in-house costs, outsourcing costs, durations, probabilities of success, and resource requirements associated with all the tasks are summarized in Table

Table 4. Problem Data

parameter	product (<i>l</i>)	discretization points (<i>m</i>)		
		1	2	3
f_{lm} (\$1000/day)	1	1	5	10
	2	1	5	12
	3	1	5	12
b_{lm} (days)	1	0	300	600
	2	0	300	600
	3	0	300	600

3. The technological precedence between the various tests is shown in Figure 8. The values of the parameters for the piecewise income decrease functions are summarized in Table 4.

Seven different problems of varying complexity are generated using this data. First, we can consider the problem of scheduling testing tasks for each product individually. Since each product has testing tasks that require the same resources, deriving an optimal schedule requires incorporating resource constraints. The

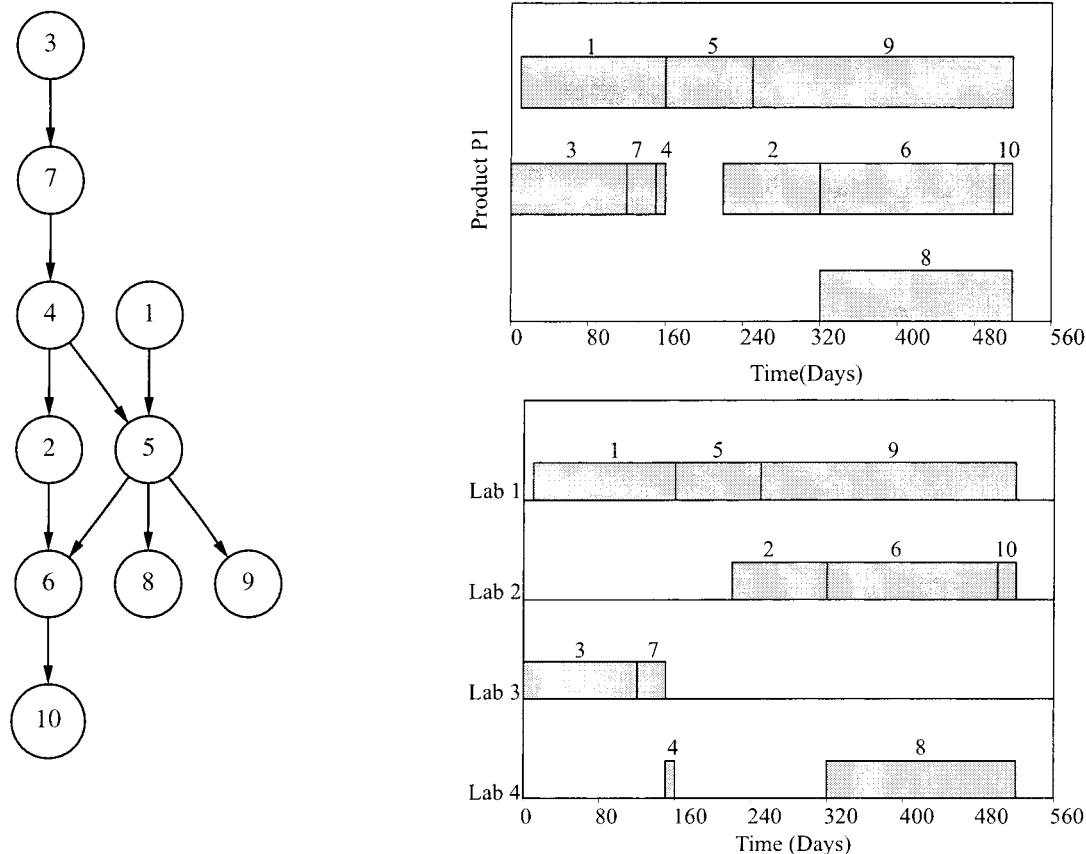


Figure 9. Example 1.

Table 5. Examples Generated

example 1	only product P1 has to be tested
example 2	only product P2 has to be tested
example 3	only product P3 has to be tested
example 4	products P1 and P2 have to be tested
example 5	products P2 and P3 have to be tested
example 6	products P1 and P3 have to be tested
example 7	products P1, P2, and P3 have to be tested

next level of complexity arises when two products have to undergo testing at the same time. Three different problems can be generated at this level of complexity. Finally, the most complex case arises when all three products have to undergo testing at the same time. Note that the importance of the resource constraints increases with the number of products to be tested. Details of the seven different examples generated in this manner have been summarized in Table 5.

First, let us consider the case when only product P1 has to undergo testing (example 1). The problem can be modeled using the MILP model M1. The optimal sequence, start times of the associated tasks (1–10), and use of various resources can be obtained after solving the MILP model to optimality. The optimal solution is shown in Figure 9. The completion time for testing is 520 days, the cost of completing the tests is \$2 261 413, and none of the tests are outsourced. It is interesting to observe that tests 4, 5, and 7 have been scheduled earlier. This is because these tasks have a non-zero probability of failure and if any of them fails, then product P1 has to be abandoned. This will save the company the expenses for the remaining tests. Also, note that even though lab 1 is the bottleneck, none of the tests 1, 5, and 9 have been outsourced. This is because there is technological precedence between these tests and outsourcing will not yield any benefits.

Interestingly, there is no arc between tests 1 and 2 even, though test 2 starts after test 1 finishes. This is because test 1 has unit probability of success and adding an arc will not reduce the expected cost of test 2.

The MILP model for this example has 108 binary variables, 309 continuous variables, and 547 constraints. The problem was modeled using GAMS modeling language¹¹ and it was solved using two different commercial solvers, CPLEX 4.0¹² and XPRESS-MP 10.0.¹³ It took 239 nodes and 1.1 CPU sec. to solve the problem using CPLEX 4.0 and took 111 nodes and 1 CPU sec. to solve the problem using XPRESS-MP 10.0 on a HP C110 workstation.

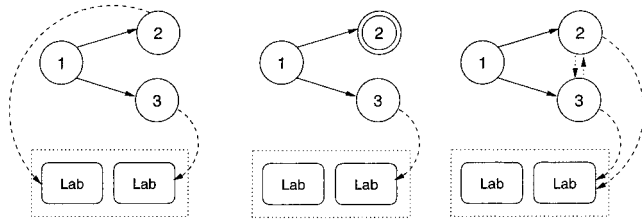
Examples 2–7 can also be modeled using MILP model M1. The computational results for all the examples are summarized in Table 6. The largest problem (example 7) has 474 binary variables, 1671 continuous variables, and 10 104 constraints. Observe that it took less than 2 s in CPU time to solve each of the single-product scheduling examples (examples 1–3) and the integrality gap for these example problems was less than 5%. However, most of the 2-product and 3-product examples (example 4, example 6, and example 7) could not be solved to optimality because either the tree size became more than 256 MB, the computational time of 24 h was not sufficient, or more than 1 000 000 nodes were needed for the branch and bound search.

These computational results clearly indicate that the proposed model is only useful for rigorously solving small problems. To solve actual industrial-sized problems, there is a need to explore alternative models and algorithms. We present an alternative representation for resource constraints that can be used to model the problem at hand.

Table 6. Computational Results

products	bin. var., cont. var., constraints	LP relaxation (\$10 000)	solver	best solution (\$10 000)	nodes	CPU time (s) ^a
P1	108/309/547	218.616	CPLEX	226.141 ^b	239	1.1
			XPRESS-MP	226.141 ^b	111	1
P2	108/309/547	157.361	CPLEX	167.489 ^b	282	1.62
			XPRESS-MP	167.489 ^b	137	1
P3	110/307/547	62.629	CPLEX	65.7995 ^b	210	1.36
			XPRESS-MP	65.7995 ^b	385	1
P1, P2	264/865/3249	454.881	CPLEX	589.91 ^c	489000	16804.82
			XPRESS-MP	490.9723 ^d	1000000	23544
P2, P3	268/865/3251	230.651	CPLEX	259.11	208838	4592.69
			XPRESS-MP	259.11 ^b	868634	12623
P1, P3	268/865/3251	321.750	CPLEX	350.89 ^c	663000	19723.39
			XPRESS-MP	347.432 ^c	1000000	29888
P1, P2, P3	474/1671/10104	677.129	CPLEX	923.935 ^c	304000	37167.10
			XPRESS-MP	812.204 ^c	5900	603

^a On a HP 9000/C110 workstation. ^b Optimal solution. ^c Suboptimal solution. ^d Optimal solution but optimality could not be proved.

**Figure 10.** Resolving resource constraints using a graph.

7. Graph-Based Representation for Resource Constraints

This representation exploits the fact that any two tests that need resources from the same resource category will not violate resource constraints if they do not share any resources in that category. The feasibility of the pair of tasks is given when at least one of them is outsourced or they use different units in that resource category. However, if they do share the same resources, then the resource constraints can be enforced by adding a precedence arc between those two tasks. This can be clearly seen in Figure 10. In this example, tests 2 and 3 need a unit resource from the same resource category. In the first 2 graphs, resource constraints are not violated because, in the first graph, tests 2 and 3 are not assigned to the same resource, and in the second graph, test 2 has been outsourced. However, if they are assigned to the same resource, then we need to add 1 of the 2 dotted arcs between tests 2 and 3 shown in the third graph. An arc between these two tests enforces the precedence between them, that is, either test 2 is before test 3 or test 3 is before test 2.

This observation can now be used to write resource constraints in mathematical form. The first constraint ensures that if a test is not outsourced ($\hat{z}_i = 0$), then the required number of units from each resource category are assigned to that test.

$$\sum_{j \in (J^i \cap J^i)} \hat{x}_{ij} = N_{ir}(1 - \hat{z}_i) \quad \forall (i \in I), (r \in R) \quad (44)$$

Here, \hat{x}_{ij} is a binary variable and has the same meaning as defined earlier; that is, it is 1 if resource j is assigned to test i , and zero otherwise. Recall that N_{ir} is the number of units needed from resource category r to perform test i . To model resource constraints using a graph-based representation, we need to define a new binary variable, $\hat{y}_{i\ell}$. It denotes an arc between the two

tests i and ℓ corresponding to two different products. It should be noted that this arc does not affect the cost equation (3) in the same manner as an arc between the tests of the same product that was modeled using the binary variable $y_{i\ell}$. The logical relationship that enforces the resource constraints can be stated as follows (refer to Figure 10 with $i = 3$ and $\ell = 2$): *If test i is not outsourced and is assigned to resource j , then for resource constraints to hold, any other test ℓ is either outsourced or is not assigned to resource j or there is an arc between tests i and ℓ .*

This relationship can be very easily written using logic propositions. The following two logic propositions enforce this relationship:

$$\neg \hat{z}_i \wedge \hat{x}_{ij} \Rightarrow \hat{z}_\ell \vee \neg \hat{x}_{\ell j} \vee y_{i\ell} \vee y_{\ell i} \\ \forall j \in J, i \in I, \ell \in (I \cap I^i), i > \ell \quad (45)$$

$$\neg \hat{z}_i \wedge \hat{x}_{ij} \Rightarrow \hat{z}_\ell \vee \neg \hat{x}_{\ell j} \vee \hat{y}_{i\ell} \vee \hat{y}_{\ell i} \\ \forall j \in J, i \in I, \ell \in I \setminus I^i, i > \ell \quad (46)$$

The first logic proposition is for the tests needed for the same product, and the second logic proposition is for the tests required for two different products. These logical relationships can be rewritten as linear inequalities using the transformation presented by Raman and Grossmann.¹⁴ Hence,

$$\hat{x}_{ij} + \hat{x}_{\ell j} - y_{i\ell} - y_{\ell i} - \hat{z}_i - \hat{z}_\ell \leq 1 \\ \forall j \in J, i \in I, \ell \in (I \cap I^i), i > \ell \quad (47)$$

$$\hat{x}_{ij} + \hat{x}_{\ell j} - \hat{y}_{i\ell} - \hat{y}_{\ell i} - \hat{z}_i - \hat{z}_\ell \leq 1 \\ \forall j \in J, i \in I, \ell \in I \setminus I^i, i > \ell \quad (48)$$

$$\hat{y}_{i\ell} \in \{0, 1\} \quad (49)$$

Finally, the following constraint is needed to enforce the sequence that is implied by an arc between two tests that do not correspond to the same product.

$$s_i + d_i \leq s_\ell + U_i(1 - \hat{y}_{i\ell}) \quad \forall r, i \in I, \ell \in I \setminus I^i \quad (50)$$

Note that a similar constraint (eq 13) has already been defined for the tests needed for the same product. Hence,

Table 7. Computational Results for the Graph-Based Model

products	bin. var., cont. var., constraints	LP relaxation (\$10 000)	solver	best solution (\$10 000)	nodes	CPU time ^a (s)
P1	82/243/457	218.616	CPLEX	226.141 ^b	41	0.24
			XPRESS-MP	226.141 ^b	31	0
P2	82/243/457	157.361	CPLEX	167.489 ^b	53	0.38
			XPRESS-MP	167.489 ^b	39	0
P3	84/241/457	62.629	CPLEX	65.7995	27	0.26
			XPRESS-MP	65.7995 ^b	19	0
P1, P2	212/685/3085	454.881	CPLEX	490.9723 ^b	7511	39.57
			XPRESS-MP	490.9723 ^b	106528	671
P2, P3	216/683/3088	230.651	CPLEX	283.335 ^c	517000	3183.24
			XPRESS-MP	259.11 ^b	868634	12611
P1, P3	216/683/3088	321.750	CPLEX	344.51 ^b	271728	1260.07
			XPRESS-MP	344.51 ^b	941705	13246
P1, P2, P3	396/1325/9891	677.129	CPLEX	778.39 ^c	427000	5245.48
			XPRESS-MP	747.188 ^c	1000000	28517

^a On a HP 9000/C110 workstation. ^b Optimal solution. ^c Suboptimal solution.

Table 8. Computational Results Using Strong Branching Option

products	bin. var., cont. var., constraints	LP relaxation (\$10 000)	optimal value (\$10 000)	nodes		CPU time (s)	
				CPLEX	CPLEX	CPLEX	CPLEX
P1	82/243/457	218.616	226.141	16	16	0.51	0.51
P2	82/243/457	157.361	167.489	21	21	1.04	1.04
P3	84/241/457	62.629	67.7995	16	16	0.60	0.60
P1, P2	212/685/3085	454.881	490.9723	412	412	40.9	40.9
P2, P3	216/683/3088	230.651	259.110	411	411	50.16	50.16
P1, P3	216/683/3088	321.750	344.51	727	727	80.22	80.22
P1, P2, P3	396/1325/9891	677.129	728.287	13565	13565	4067.51	4067.51

the MILP model M2 comprising eq 1 as the objective and eqs 2, 5, 7–17, 44, 47–50, 30–33, 35–39, and 41–42 as constraints can be used to solve the problem at hand.

8. Examples Revisited

In this section, we resolve all the examples using the alternative model M2. The computational results for all the example problems are summarized in Table 7. Using the alternative model, we were able to solve all the single-product and two-product examples to optimality. However, the three-product example (example 7) was still unsolvable to optimality with default choices of the branch-and-bound search. The tree size became more than 256 MB in the case of CPLEX and more than 100 000 nodes were needed in the case of XPRESS-MP. The linear-programming relaxations for all 7 examples were the same as before. Furthermore, for all 7 examples, the size of the model is comparable to the size of the earlier model. Even though model M2 is computationally faster than model M1, the computational performance is not good enough because the largest problem is still unsolvable.

However, one of the advantages of the alternative model M2 is that the sequencing, assignment, and outsourcing variables are very strongly interrelated because of the graph-based representation. This fact can be used to our advantage by changing the variable selection rule of the branch-and-bound method for solving the MILP model. So far, we have used the default variable selection strategy (branching based on pseudo reduced costs) to solve the model. One of the variable selection strategies available in CPLEX is strong branching.¹² The key feature of this strategy is that it first evaluates the effects of setting a variable to a certain value and then branches on the variable that

looks most promising. In some cases, inferences can be drawn about the infeasibility and optimality of a branch-and-bound node, while the effects of fixing a variable to a certain value are evaluated. If a model helps in drawing such inferences, then the solution time can be significantly reduced because bounds improve much more rapidly and the optimal solution can be found earlier in the branch-and-bound search. The computational results for solving all seven examples using model M2 with the strong branching variable selection strategy in CPLEX are summarized in Table 8. Comparing with the computational results in Table 6, it is clear that solutions obtained for larger problems using model M1 were indeed suboptimal. All the examples in Table 8 were solved to optimality, and the computational times with CPLEX are an order of magnitude lower in some cases: It is worth emphasizing that it is the combination of the modeling and the strong branching solution strategy that yields such results. For example, if we solve all the examples with model M1 using the strong branching search strategy, not only all the examples that were unsolvable initially (Table 6) remain unsolvable, but also it takes even more time to solve the examples that were solvable to optimality. Note that XPRESS-MP does not have strong branching.

The details of the optimal solution obtained for the largest problem, example 7, are as follows. The schedule is as shown in Figure 11. First of all, it should be noted that the sequence of tasks for product P1 in this case is different from the one obtained for example 1. This is because resource availability here is dependent on the testing schedules for products P2 and P3. This does not come as a surprise because this behavior was also demonstrated in the motivating example. Another interesting observation that can be made is that in this example 7 tasks have been outsourced. These tasks are marked by double circles in the activity-on-node representation of the sequence of tasks and with dark blocks

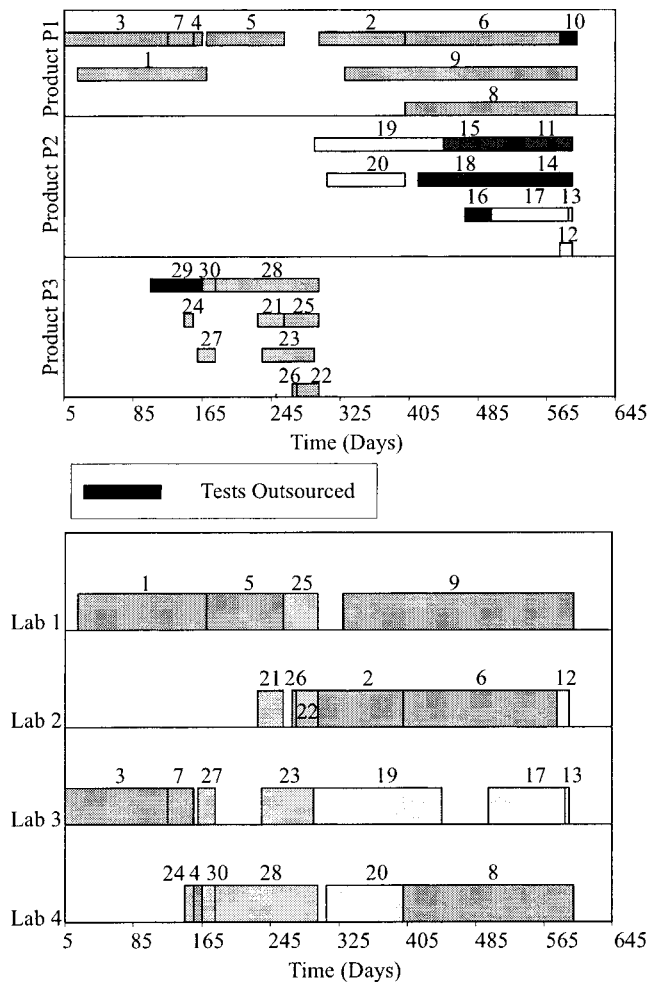
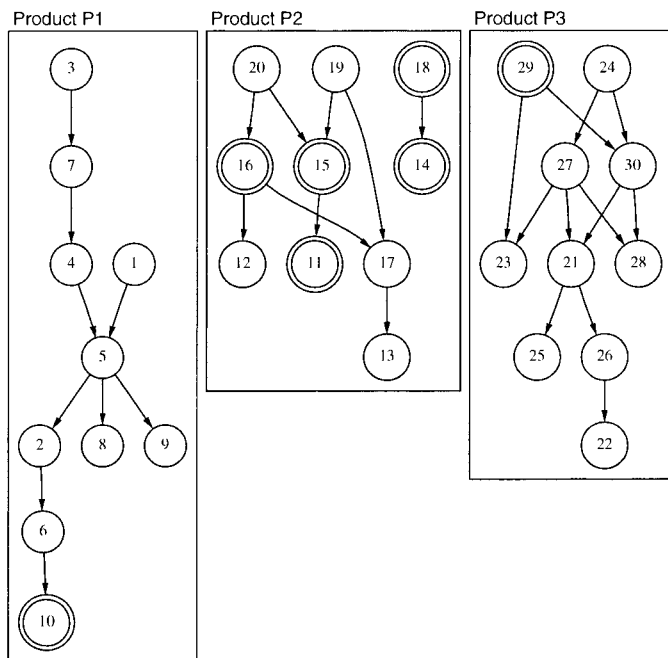


Figure 11. Example 7.

in the Gantt chart. In this example, there are 2 main reasons for outsourcing. Tests 11, 15, and 29 were outsourced because they need lab 1 and it is clearly a bottleneck. However, the same logic does not hold for tests 10 and 16 as lab 2 is available for the first 240 days of the schedule. They had to be outsourced because the technological precedence does not allow them to be scheduled earlier. This example clearly highlights the complex decision making involved in scheduled testing for new product development.

9. Conclusions

The problem of resource-constrained scheduling of testing tasks for new product development has been addressed in this paper. The problem is important because in some industries like pharmaceutical and agrochemicals a new product is required to pass all the tests by federal laws. If a product fails any of the tests, then all the remaining work on that product is halted and the investment in the previous tests is wasted. The models presented take into account complex trade-offs and have the capability of deriving a schedule that satisfies the resource constraints and utilize the option of outsourcing.

Two different MILP models were developed. The main difference was in the representation of the resource constraints. The first MILP model relied on a slot-based representation to handle the resource constraints. This

model was only useful for solving small problems. The second MILP model exploited a graph-based representation for handling resource constraints. This MILP model, even though similar in size to the first model, proved to be computationally more efficient, particularly when combined with a specialized variable selection strategy (strong branching). Thus, it was demonstrated that the proper combination of modeling and search strategy made the difference in successfully tackling problems with up to 30 testing tasks. It is recognized that, in practice, a considerably larger number of tasks may be involved. In such a case, the proposed model could only be applied either by merging tasks or else by imposing additional precedence constraints, which practical problems may exhibit in the first place.¹⁵ Finally, it was shown that it is critical to incorporate resource constraints along with sequencing of testing tasks to obtain a globally optimal solution.

Acknowledgment

We acknowledge financial support from the National Science Foundation under Grant CTS-9810182.

Nomenclature

- Indices*
- i, i', i'' = a task
- j = a resource
- r = a resource category

k = a slot

k_j^f = the final slot for resource j

I = a product

m = linearization point for decrease in income function

n = linearization point for cost

Sets

I = the set of tests

J = the set of resources

L = the set of products

R = the set of resource categories

A = the set of precedence constraints

\mathcal{P} = the set of tests that can be scheduled on unit j

\mathcal{P}^l = the set of tests corresponding to product l

\mathcal{P}^i = the set of tests that correspond to the product that has i as one its test

J^r = the set of resources in category r

J^i = the set of resources that can be used for test i

K_j = the set of slots defined for resource j

M_i = the set of linearization points for a decrease in income function

N_i = the set of linearization points for cost function

Parameters

a_{in} = the location of grid point n for variable w_i

b_{lm} = the location of grid point m for the piecewise linear decrease in income of product l

c_i = the cost of completing the test i by using available resources

\bar{c}_i = the cost of outsourcing test i

d_i = the duration of test i

f_{lm} = the value of the slope of profit function for product l at linearization point m

N_{ir} = the units of resource r required by test i

p_i = the probability of success of test i

\bar{r} = the rate of interest compounded continuously

U_i = the upper bound on start time of test i

U_j = the upper bound on start time of any test on unit j

W_i = the lower bound on the variable w_i

Continuous Variables

C_i = cost for completing test i

u_{lm} = an extra variable to calculate income

s_i = start time of test i

S_{jk} = start time of slot k of unit j

t_l = completion time of product l

w_i = an extra variable for piecewise derivation of cost function

$\hat{\lambda}_{in}$ = a piecewise linear variable for test i at point n in λ formulation of cost function

λ_{in} = a disaggregated piecewise linear variable for test i at point n in λ formulation of cost function

$\bar{\lambda}_{in}$ = a disaggregated piecewise linear variable for test i at point n in λ formulation of outsourcing cost function

τ_{ijk} = a disaggregated variable in time-matching constraints

β_{ij} = a disaggregated variable in time-matching constraints

γ_{jk} = a disaggregated variable in time-matching constraints

Boolean Variables

x_{ijk} = 1 if test i is assigned to slot k of unit j

\hat{x}_{ij} = 1 if test i uses unit j

y_{ir} = 1 if test i is finished after test r

\hat{y}_{ir} = 1 if test i is finished after test r

\hat{z}_i = 1 if test i is outsourced

z_{jk} = 1 if slot k of unit j is empty

A. Linearization of Disjunction (6)

Using the convex hull formulation,^{16,17} disjunction (6) can be written in mixed-integer form as

$$C_i = C_i^1 + C_i^2 \quad \forall (i \in I) \quad (51)$$

$$\hat{\lambda}_{in} = \lambda_{in} + \bar{\lambda}_{in} \quad \forall (i \in I), (n \in N_i) \quad (52)$$

$$w_i = w_i^1 + w_i^2 \quad \forall (i \in I) \quad (53)$$

$$C_i^1 = c_i \left(\sum_n e^{a_{in}} \lambda_{in} \right) \quad \forall (i \in I) \quad (54)$$

$$C_i^2 = \bar{c}_i \left(\sum_n e^{a_{in}} \bar{\lambda}_{in} \right) \quad \forall (i \in I) \quad (55)$$

$$w_i^1 = \sum_n a_{in} \lambda_{in} \quad \forall (i \in I) \quad (56)$$

$$w_i^2 = \sum_n a_{in} \bar{\lambda}_{in} \quad \forall (i \in I) \quad (57)$$

$$\sum_n \lambda_{in} = 1 - \hat{z}_i \quad \forall (i \in I) \quad (58)$$

$$\sum_n \bar{\lambda}_{in} = \hat{z}_i \quad \forall (i \in I) \quad (59)$$

$$C_i^1 \leq U^{C_i} (1 - \hat{z}_i) \quad \forall (i \in I) \quad (60)$$

$$C_i^2 \leq U^{C_i} \hat{z}_i \quad \forall (i \in I) \quad (61)$$

$$W_i (1 - \hat{z}_i) \leq w_i^1 \leq 0 \quad \forall (i \in I) \quad (62)$$

$$W_i \hat{z}_i \leq w_i^2 \leq 0 \quad \forall (i \in I) \quad (63)$$

$$\lambda_{in} \geq 0 \quad \forall (i \in I), (n \in N_i) \quad (64)$$

$$\lambda_{in} \leq (1 - \hat{z}_i) \quad \forall (i \in I), (n \in N_i) \quad (65)$$

$$\bar{\lambda}_{in} \geq 0 \quad \forall (i \in I), (n \in N_i) \quad (66)$$

$$\bar{\lambda}_{in} \leq \hat{z}_i \quad \forall (i \in I), (n \in N_i) \quad (67)$$

Here, C_i^1 , C_i^2 , w_i^1 , w_i^2 , λ_{in} and $\bar{\lambda}_{in}$ are reformulation variables and U^{C_i} is the valid upper bound on C_i . It can be observed that bounding constraints (65) and (67) are redundant because they are dominated by constraints (58) and (59), respectively. Furthermore, inequalities (60)–(63) are also redundant because they are dominated by eqs 54–57, respectively. Also, we can reduce the equations (51), (54), and (55) by eliminating C_i^1 and C_i^2 . Similarly, eqs 53, 56, and 57 can be reduced by eliminating w_i^1 and w_i^2 . Finally, eq 52 can be dropped because there are no other constraints that involve $\hat{\lambda}_{in}$.

Therefore, the reduced set of constraints is as follows:

$$C_i = c_i \left(\sum_n e^{a_{in} \lambda_{in}} \right) + \bar{c}_i \left(\sum_n e^{a_{in} \bar{\lambda}_{in}} \right) \quad \forall (i \in I) \quad (68)$$

$$w_i = \sum_n a_{in} (\lambda_{in} + \bar{\lambda}_{in}) \quad \forall (i \in I) \quad (69)$$

$$\sum_n \lambda_{in} = (1 - \hat{z}_i) \quad \forall (i \in I) \quad (70)$$

$$\sum_n \bar{\lambda}_{in} = \hat{z}_i \quad \forall (i \in I) \quad (71)$$

$$\lambda_{in}, \bar{\lambda}_{in} \geq 0 \quad \forall (i \in I), (n \in N) \quad (72)$$

B. Reformulation of Time-Matching Constraints

The time-matching constraints (23) and (24) can also be enforced using the following nonlinear constraint:

$$(s_i - S_{jk})x_{ijk} = 0 \quad \forall (i \in I), (j \in J), (k \in K) \quad (73)$$

This constraint can be rewritten as

$$s_i x_{ijk} = S_{jk} x_{ijk} \quad \forall (i \in I), (j \in J), (k \in K) \quad (74)$$

Let us introduce a new variable τ_{ijk} such that

$$\tau_{ijk} = s_i x_{ijk} \quad \forall (i \in I), (j \in J), (k \in K) \quad (75)$$

Using (75), we can rewrite (74) as

$$\tau_{ijk} = S_{jk} x_{ijk} \quad \forall (i \in I), (j \in J), (k \in K) \quad (76)$$

Adding (75) over the time slots (k) and (76) over the tests ($i \in I$) yields

$$\sum_{k \in K_j} \tau_{ijk} = s_i \sum_{k \in K_j} x_{ijk} \quad \forall (i \in I), (j \in J) \quad (77)$$

$$\sum_{i \in I} \tau_{ijk} = S_{jk} \sum_{i \in I} x_{ijk} \quad \forall (j \in J), (k \in K) \quad (78)$$

Equations 77 and 78 can be rewritten by using eqs 19 and 20, respectively:

$$\sum_{k \in K_j} \tau_{ijk} = s_i \hat{x}_{ij} \quad \forall (i \in I), (j \in J) \quad (79)$$

$$\sum_{i \in I} \tau_{ijk} = S_{jk} (1 - z_{jk}) \quad \forall (j \in J), (k \in K) \quad (80)$$

Adding and subtracting s_i from the right-hand side of (79), eqs 79 and 80 can be rewritten as follows,

$$s_i = \sum_{k \in K_j} \tau_{ijk} + \beta_{ij} \quad \forall (i \in I), (j \in J) \quad (81)$$

$$S_{jk} = \sum_{i \in I} \tau_{ijk} + \gamma_{jk} \quad \forall (j \in J), (k \in K) \quad (82)$$

where $\beta_{ij} = s_i(1 - \hat{x}_{ij})$ and $\gamma_{jk} = S_{jk}z_{jk}$.

Finally, we establish the connections between the continuous variables τ_{ijk} , β_{ij} , and γ_{jk} and the binary variables x_{ijk} , \hat{x}_{ij} , and z_{jk} , respectively:

$$0 \leq \tau_{ijk} \leq U_i x_{ijk} \quad \forall (i \in I), (j \in J), (k \in K) \quad (83)$$

$$0 \leq \beta_{ij} \leq U_i(1 - \hat{x}_{ij}) \quad \forall (i \in I), (j \in J) \quad (84)$$

$$0 \leq \gamma_{jk} \leq U_j z_{jk} \quad \forall (j \in J), (k \in K) \quad (85)$$

Hence, the constraints (81)–(85) can be used instead of constraints (23)–(24) to enforce the time matching.

Literature Cited

- Schmidt, C. W.; Grossmann, I. E. Optimization models for the scheduling of testing tasks in new product development. *Ind. Eng. Chem. Res.* **1996**, *35* (10), 3498–3510.
- Honkomp, S. J.; Reklaitis, G. V.; Pekny, J. F. Robust planning and scheduling of process development projects under stochastic conditions. Presented at AIChE Annual Meeting, Los Angeles, CA, 1997.
- Kondili, E.; Pantelides, C. C.; Sargent, R. W. H. A general algorithm for scheduling of batch operations-I. *Comput. Chem. Eng.* **1993**, *17* (2), 221–227.
- Pinto, J. M.; Grossmann, I. E. A continuous time mixed integer linear programming model for short term scheduling of multistage batch plants. *Ind. Eng. Chem. Res.* **1995**, *34* (9), 3037–3051.
- Pinto, J. M.; Grossmann, I. E. A logic based approach to scheduling problems with resource constraints. *Comput. Chem. Eng.* **1997**, *21* (8), 801–818.
- Shah, N.; Pantelides, C. C.; Sargent, R. W. H. A general algorithm for scheduling of batch operations-II. *Comput. Chem. Eng.* **1993**, *17* (2), 229–224.
- Schilling, G.; Pantelides, C. C. A simple continuous-time process scheduling formulation and a novel solution algorithm. *Comput. Chem. Eng.* **1996**, *20* (Suppl.), S1221–S1226.
- Zhang, X.; Sargent, R. W. H. The optimal operation of mixed production facilities—a general formulation and some approaches for the solution. *Comput. Chem. Eng.* **1996**, *20* (6/7), 897–904.
- Zhang, X.; Sargent, R. W. H. The optimal operation of mixed production facilities—extensions and improvements. *Comput. Chem. Eng.* **1996**, *20* (Suppl.), S1287–S1293.
- Pantelides, C. C. Unified frameworks for optimal process planning and scheduling. In *Foundations of Computer Aided Process Operations*; Rippin, D. W. T., Hale, J. C., Davis, J. F., Eds.; CACHE: Austin, TX, 1994; pp 253–274.
- Brooke, A.; Kendrick, D.; Meeraus, A. *GAMS: A Users Guide*; The Scientific Press Series, 1992.
- CPLEX. *Using the CPLEX Callable Library*; CPLEX Optimization, Inc., 1995.
- XPRESS-MP. *User Guide and Reference Manual*; Dash Associates, 1997.
- Raman, R.; Grossmann, I. E. Relation between milp modelling and logical inference for chemical process synthesis. *Comput. Chem. Eng.* **1991**, *15* (2), 73–84.
- Schmidt, C. W.; Grossmann, I. E.; Blau, G. E. Optimization of industrial scale scheduling problems in new product development. *Comput. Chem. Eng.* **1998**, *22* (Suppl.), S1027–S1030.
- Balas, E. Disjunctive programming and a hierarchy of relaxations for discrete optimization problems. *SIAM J. Alg. Disc. Meth.* **1985**, *6*, 466–486.
- Türkay, M.; Grossmann, I. E. Disjunctive programming techniques for the optimization of process systems with discontinuous investment costs-multiple size regions. *Ind. Eng. Chem. Res.* **1996**, *35* (8), 2611–2623.

Received for review December 14, 1998
Revised manuscript received May 3, 1999
Accepted May 3, 1999

IE9807809