Scheduling Optimization under Uncertainty - An Alternative Approach

J. Balasubramanian[*] and I. E. Grossmann[†]

Department of Chemical Engineering, Carnegie Mellon University

Pittsburgh, PA 15213, USA

March 2002

Abstract

The prevalent approach to the treatment of processing time uncertainties in production scheduling problems is through the use of probabilistic models. Apart from requiring detailed information about probability distribution functions, this approach also has the drawback that the computational expense of solving these models is very high. In this work, we present a non-probabilistic treatment of scheduling optimization under uncertainty, based on concepts from fuzzy set theory and interval arithmetic, to describe the imprecision and uncertainty in the task durations. We first provide a brief review on the fuzzy set approach, comparing it with the probabilistic approach. We then present MILP models derived from applying this approach to two different problems - flowshop scheduling and new product development process scheduling. Results indicate that these MILP models are computationally tractable for reasonably sized problems. We also describe tabu search implementations in order to handle larger problems.

1 Introduction

With increased interest in production scheduling in chemical engineering in recent years, there has been much work addressing the optimal scheduling of a wide range of sytems - from individual production units to geographically distributed multisite supply chains (see Shah, 1998, for review).

[*]jayanth@cmu.edu

[†]grossmann@cmu.edu - Corresponding author

A significant amount of the work in this area has focussed on the development of deterministic models, where the problem data is assumed to be known in advance. In reality, though, there can be uncertainty in a number of factors such as processing times and costs. As a result, a number of papers in the recent years have addressed scheduling in the face of uncertainties in different parameters - for e.g., demands (Ierapetritou and Pistikopoulos, 1996; Petkov and Maranas, 1997; Sand *et al.*, 2000) and processing times (Honkomp *et al.*, 1999; Schmidt and Grossmann, 2000; Balasubramanian and Grossmann, 2001).

The prevalent approach to the treatment of these uncertainties is through the use of probabilistic models that describe the uncertain parameters in terms of probability distributions. However, the evaluation and optimization of these models is computationally expensive, either because of the large number of scenarios resulting from a discrete representation of the uncertainty (Wets, 1974), or the need to use complicated multiple integration techniques when the uncertainty is represented by continuous distributions (Schmidt and Grossmann, 2000). Furthermore, the use of probabilistic models is realistic only when these descriptions of the uncertain parameters is available, say from historic data. When such data is not available (for example in New Product Development, when the tasks have never been or only rarely performed before), we do not have enough information for inferring or deriving the probabilistic models. In such situations, we have to resort to an alternative treatment of uncertainty. For example, the modeler may be able to approximate the duration of the tasks and specify the longest and shortest durations or the interval in which the duration belongs at different levels of confidence.

In this work, we draw upon concepts from fuzzy set theory and interval arithmetic to describe the imprecision and uncertainties in the durations of batch processing tasks. Indeed, this approach has been receiving increasing attention recently. McCahon and Lee (1992) were the first to illustrate the application of fuzzy set theory as a means of analyzing performance characteristics for a flowshop system. They modified the Campbell, Dudek and Smith sequencing heuristic (Campbell *et al.*, 1970) for the case of fuzzy processing times. However, their procedure was cumbersome for fairly large problem sizes. Most of the work in applying fuzzy set theory to scheduling optimization has primarily focussed on using heuristic search techniques such as simulated annealing and genetic algorithms to obtain near-optimal solutions. For example, Ishibuchi *et al.* (1994) examined flowshop scheduling problems with fuzzy due dates, where the membership function of the fuzzy due date indicates the grade of satisfaction of the decision maker with the completion time of a job. These authors presented results from applying genetic algorithms to two types of problems - i) maximizing the minimum grade of satisfaction over all jobs, and ii) maximizing the total grade of satisfaction. Fortemps (1997) addressed the problem of minimizing the makespan of jobshops with fuzzy durations using simulated annealing. The author also introduced a new comparison method for fuzzy numbers. A recent paper (Özelkan and Duckstein, 1999) investigates the necessary conditions for optimality of fuzzy counterparts of classical (deterministic) optimal scheduling rules and emphasizes the importance of using ranking functions that satisfy certain properties. In the area of project scheduling, Chanas and Kamburowski (1981) laid the theoretical foundations for approaching the problem through the fuzzy set framework. Lootsma (1989) contrasted the fuzzy set approach with the well-known stochastic PERT (Malcolm *et al.*, 1959) and discussed some of the pitfalls associated with stochastic PERT, including the computational complexity of PERT (Hagstrom, 1988). Hapke and Slowinski (1996) and Wang (1999) addressed the resource-constrained project scheduling problem. While the former applied heuristic dispatching rules to generate the schedules, the latter formulated the problem as a fuzzy constraint satisfaction problem based on possibility theory (Dubois and Prade, 1988) and used a beam search (Ow and Morton, 1988) to obtain a schedule with the least possibility of being late. We refer the reader to Slowinski and Hapke (2000) as an excellent overview of scheduling under fuzziness. In the chemical engineering literature, the theory of fuzzy sets has primarily been applied for process control (see for examples of recent works, Postlethwaithe and Edgar, 2000; Galluzo *et al.*, 2001) although there have also been applications in the area of design (Kubic and Stein, 1988; Tarifa and Chiotti, 1995). Majozi and Zhu (2001) recently presented an application of fuzzy set theory to the optimal allocation of operators to batch plants.

Unlike the previously mentioned papers, which have relied on heuristics to obtain near-optimal solutions, we show how it is possible to develop Mixed Integer Linear Programming (MILP) models for different scheduling problems that use a fuzzy representation of uncertainty. We also compare the MILP approach with heuristic techniques for flowshop problems. As a brief introduction to the concepts of this non-probabilistic approach, we present results from flowshop optimization under uncertainty. The durations of the tasks are described by fuzzy numbers (as opposed to deterministic values or probabilistic descriptions). We present MILP models for selecting the optimal schedule for two types of flowshop systems - (i) multistage flowshops with a single processing unit in each stage (Birewar and Grossmann, 1989; Pekny and Miller, 1991), and (ii) multistage flowshops with multiple parallel units in each stage (Pinto and Grossmann, 1995). We then address the problem of scheduling the New Product Development (NPD) process for an agricultural chemical under uncertainty (Schmidt and Grossmann, 1996). In the NPD process, the potential product must pass a series of tests that assess its safety, efficacy and environmental impact. The costs, durations and probabilities of success of various testing tasks are often not known with certainty when the schedule has to be constructed. We consider the problem of optimizing the schedule of testing tasks while taking into account the uncertainties in the durations and success of the tasks, a problem which has not been reported before. The optimization of the testing schedule consists in introducing precedence constraints between the tests, in addition to the given set of technological precedence constraints. We present an MILP formulation for the objective of maximizing the difference of the expected value of the fuzzy income and the expected cost of a testing schedule.

The paper is organized as follows. Section 2 defines the problems of interest. Section 3 presents a brief overview of fuzzy set theory and compares mathematical operations in the fuzzy framework with their probabilistic counterparts. After an overview of our approach to the problem in Section 4, we present results from optimization of flowshops in Sections 5 and 6 and the NPD process in Section 7. Finally, the conclusions on this work are drawn.

2 Problem Statement

We consider in this paper the following general problem. Given are a set of tasks (orders or tests) to be performed using certain resources (processing equipment). These tasks have their durations specified as fuzzy numbers. The objective is to determine an allocation of the resources to the tasks that is optimal with respect to a certain measure such as the makespan or the profit. Our principal objective is to demonstrate the application of concepts from fuzzy set theory to scheduling optimization when the processing times are uncertain. We present applications from flowshop and NPD testing optimization.

3 Comparison of Fuzzy Sets and Numbers with Probabilistic Approach

In this section, we review key concepts from the theory of fuzzy sets that will be used for the scheduling models. We also compare them with their probabilistic counterparts.

3.1 Definitions

A classical set S of a universe X is a collection of elements or objects that are well defined and possess some common properties. An element x of the universe X may either belong to (*true* or 1) S or not (*false* or 0). Zadeh (1965) introduced the concept of a fuzzy set in which the membership of an element to a set need not be just binary-valued (i.e., 0-1), but could be any value over the interval [0,1] depending on the degree to which the element belongs to the set. The higher the degree of belonging, higher the membership value.

A fuzzy set $\tilde{A}$ of the universe X is specified by a membership function $\mu_A(x)$ which takes its value in the interval [0,1]. For each element x of X, the quantity $\mu_A(x)$ specifies the degree to which x belongs in A. Thus, $\tilde{A}$ is completely characterized by the set of ordered pairs:

$$\tilde{A} = \{(x, \mu_{\tilde{A}}(x)) \mid x \in X\} \tag{1}$$

The support of a fuzzy set $\tilde{A}$ is a (crisp/classical) subset of X given by

$$supp(\tilde{A}) = \{x \in X \mid \mu_{\tilde{A}}(x) > 0\} \tag{2}$$

The α-level set (α-cut) of a fuzzy set $\tilde{A}$ is a crisp subset of X given by

$$\tilde{A}_\alpha = \{x \in X \mid \mu_{\tilde{A}}(x) \geq \alpha\} \qquad \forall \alpha \in (0,1] \tag{3}$$

A fuzzy number is defined as a bounded support fuzzy set of the real line $\Re$ whose α-cuts are closed intervals.

3.2 Arithmetic Operations - a Comparison with Probabilistic Computations

Arithmetic operations on fuzzy numbers that will be used for the objective function and the constraints, such as addition, subtraction, taking the maximum of two fuzzy numbers are defined through the extension principle (Zadeh, 1965), which allows the extension of operations on real numbers to fuzzy numbers. In the scheduling problems of interest to us, the principal arithmetic operations that are involved are addition (computing the fuzzy end-time of a task given the fuzzy start time and duration) and maximization (computing the start time of a task as the maximum of the end times of preceding tasks). Hence, we restrict our description of arithmetic operations on fuzzy numbers to these two operations.

3.2.1 Addition

1. Fuzzy Numbers

   If $\tilde{X}$ and $\tilde{Y}$ are two fuzzy numbers, their addition can be accomplished by using α-level cuts. If $\tilde{X}_\alpha = [x^L_\alpha, x^R_\alpha]$ and $\tilde{Y}_\alpha = [y^L_\alpha, y^R_\alpha]$, then the result $\tilde{Z}$ which is the addition of $\tilde{X}$ and $\tilde{Y}$ can be defined by the α-level sets as below:

$$\tilde{Z}_\alpha = \tilde{X}_\alpha(+)\tilde{Y}_\alpha = [x^L_\alpha + y^L_\alpha, x^R_\alpha + y^R_\alpha] \qquad \forall \alpha \in (0,1] \tag{4}$$

   This clearly shows that the lower bound of $\tilde{Z}$ is the sum of the lower bounds of $\tilde{X}$ and $\tilde{Y}$ and similarly, the upper bound of $\tilde{Z}$ is the sum of the upper bounds of $\tilde{X}$ and $\tilde{Y}$. Figure 1 provides an example.

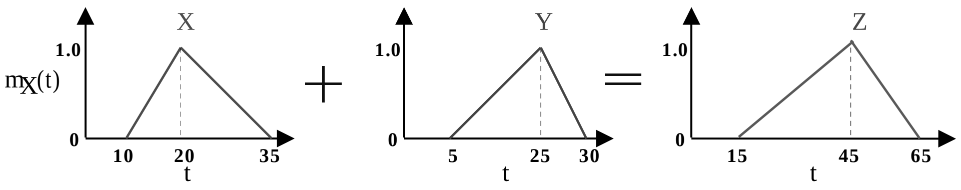

Figure 1: Addition of TFNs

2. Random Variables

   If X and Y are two independent random variables with probability density functions (PDFs) given by $f_X(t)$ and $f_Y(t)$, and Z is a random variable that is the sum of X and Y, the PDF of Z is obtained by convolving the distribution of X with that of Y. As is well known, the expected value of Z is given by the addition of the expected values of X and Y.

$$\begin{aligned} Z &= X+Y \\ f_Z(t) &= f_X(t) * f_Y(t) = \int_{-\infty}^{+\infty} f_X(u)\cdot f_Y(t-u)\cdot du \qquad (5) \\ E[Z] &= E[X]+E[Y] \qquad (6) \end{aligned}$$

3.2.2 Maximum Operator

1. Fuzzy Numbers

   If $\tilde{X}$ and $\tilde{Y}$ are two fuzzy numbers, their maximum can also be obtained by using α-level cuts. Thus,

$$\tilde{Z}_\alpha = \max(\tilde{X}_\alpha, \tilde{Y}_\alpha) = [\max(x_\alpha^L, y_\alpha^L), \max(x_\alpha^R, y_\alpha^R)] \qquad \forall\alpha \in (0,1] \qquad (7)$$

   In general, this operation requires infinite computations, i.e., evaluating maxima for every $\alpha \in (0,1]$. However, good approximations can be obtained by performing these computations at specific values of α, rather than at all values. As can be seen in Figure 2, the maximum of two TFNs need not be another TFN.

2. Random Variables

   If X and Y are two independent random variables with PDFs given by $f_X(t)$ and $f_Y(t)$, and Z is a random variable that is the maximum of X and Y, the cumulative distribution function (CDF) of Z is computed by multiplying the CDF of X with the CDF of Y. Unlike the case of the addition operation, the expected value of the maximum of two random

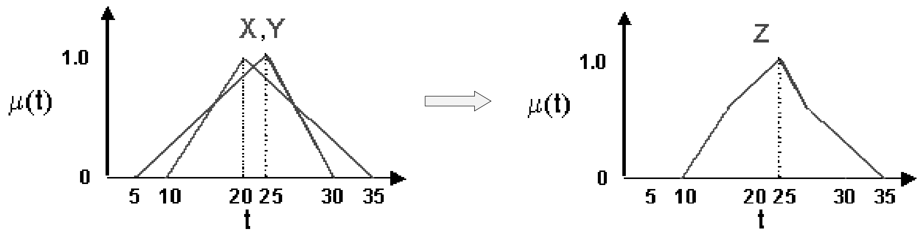

Figure 2: **Maximum of TFNs**

variables cannot be obtained by taking the maximum of the expected values. In fact, taking the maximum of the expected values results in a lower bound to the actual value.

$$Z = \max(X,Y)$$

$$F_Z(t) = F_X(t) \cdot F_Y(t) \tag{8}$$

$$E[max(X,Y)] \geq max[E(X),E(Y)] \tag{9}$$

3.3 Objective Function Comparisons

3.3.1 Fuzzy Numbers

Since we are interested in selecting a schedule with the optimum (minimum) makespan, we have to compare the fuzzy makespans of potential schedules. In a probabilistic treatment, we may be interested in minimizing the expected value of the makespan. However, with our approach, matters are not so straightforward.

There exists a large body of literature that deals with the comparison of fuzzy numbers (see Chen and Hwang, 1992, for an overview) and a number of indices have been proposed. More recently, Fortemps and Roubens (1996) proposed a method for comparing fuzzy numbers based on the compensation of the areas determined by the membership functions, which has a very desirable property that allows us to model the distance between two fuzzy numbers. This method is also related to the mean value of a fuzzy number as defined in Dubois and Prade (1987) in the framework of Dempster-Shafer theory. In the framework of Dempster-Shafer's theory, a fuzzy number $\tilde{X}$ is considered to define a set of possible probability distributions; hence the mean value of the fuzzy number $\tilde{X}$ is the set of mean values calculated according to all these probability distributions.

The lower mean value X_* and upper mean value X^* are calculated from the following distribution functions $F_*(x)$ and $F^*(x)$ as follows.

$$F^*(x) = sup\{\mu_{\tilde{X}}(t) | t \leq x\} \tag{10}$$

$$F_*(x) = inf\{1 - \mu_{\tilde{X}}(t) | t \geq x\} \tag{11}$$

$$X_* = \int_{-\infty}^{+\infty} x \cdot dF_*(x) \tag{12}$$

$$X^* = \int_{-\infty}^{+\infty} x \cdot dF^*(x) \tag{13}$$

Thus, the mean value of the fuzzy number $\tilde{X}$ is the interval $[X_*, X^*]$. Now, if we assume that all the values in the interval $[X_*, X^*]$ are equally likely, then the natural choice for a single real number would be the mid-point of this interval. The area compensation method basically involves computing the following integral:

$$AC(\tilde{X}) = 0.5 \cdot \int_0^1 (x_\alpha^L + x_\alpha^R)\, d\alpha \tag{14}$$

Fortemps and Roubens (1996) prove that the area compensation procedure provides a real number that is the mid-point of the interval corresponding to the mean value of a fuzzy number, in the sense of the Dempster-Shafer theory, where both sides of the possibility distribution induce a probability measure. Thus, we have:

$$\begin{aligned} AC(\tilde{X}) &= 0.5 \cdot \int_0^1 (x_\alpha^L + x_\alpha^R)\, d\alpha \\ &= 0.5 \cdot (X_* + X^*) \end{aligned} \tag{15}$$

The area-compensation integral in (14) can be used to compare the fuzzy makespans of potential schedules. If we are interested in minimizing the makespan, a schedule with fuzzy makespan $\tilde{X}_1$ is preferred over a schedule with fuzzy makespan $\tilde{X}_2$ if $AC(\tilde{X}_1) < AC(\tilde{X}_2)$.

3.3.2 Probabilistic Case

Suppose we wish to compute the expected value of a function of several random variables such as $Z = g(X_1, X_2, \ldots, X_N)$. The first step is to obtain the joint PDF of the random variables, i.e., $f(x_1, \ldots, x_N)$, which in the case of independent random variables is merely the product of the individual PDFs. Then, to obtain the CDF of Z, $F_Z(t)$, we have to compute a multiple integral over a polytope P that is parametric in t.

$$\begin{aligned} F_Z(t) &= \int\int \cdots \int_{P(t)} f(x_1, \ldots, x_N) \cdot dx_1 \cdots dx_N \\ P(t) &= \{a_1(t) \leq x_1 \leq b_1(t), \ldots, a_N(t) \leq x_N \leq b_N(t), g(x1, x2, \ldots, x_N) \leq t\} \end{aligned} \tag{16}$$

Once $F_Z(t)$ has been obtained, the PDF can be obtained through differentiation and the expected value of Z may be computed. However, computing the parametric multiple integral in (16) is an expensive operation and is not tractable, even when the number of random variables is moderately large.

3.3.3 Comparison

It is instructive to compare the steps involved in computing the makespan in the fuzzy set case with the probabilistic case.

When the processing times are given by fuzzy numbers, the makespan, which is a function of these fuzzy numbers, can be computed very efficiently (although approximately) through interval arithmetic at various α-levels, as in (4) and (7) and the objective computed through the one-dimensional integral in (14). However, in the probabilistic case, the distribution of the makespan is very difficult to obtain, since it has to be obtained through several parametric multiple integrals as in (16).

Furthermore, we note the following interesting properties of the area compensation operator, which have their counterparts in the probabilistic approach. The resemblance of $AC(\tilde{X})$ to the expected value operator $E(X)$ is indeed striking (see (6) and (9)).

$$AC(\tilde{X} + \tilde{Y}) = AC(\tilde{X}) + AC(\tilde{Y}) \tag{17}$$

$$AC(max[\tilde{X}, \tilde{Y}]) \geq max[AC(\tilde{X}), AC(\tilde{Y})] \tag{18}$$

4 Overview of Approach

The key ideas that we will use to apply the concepts presented in the previous section are as follows:

1. **Evaluation** of a given schedule under uncertainty

   This step involves the calculation of the start-times and end-times at various α-levels, through a set of recurrence relations. These computations typically involve the use of interval arithmetic.

2. **Generalization of Expressions**

   Once the expressions for calculating the start- and end-times for the tasks for a given schedule (sequence of tasks) have been derived, the next step is to generalize these to account for all possible schedules. This is done first by introducing binary variables that model either the precedence relationships between the tasks, or the allocation of tasks to specific processing units.

3. **Optimization**

   The area compensation integral will be used to compare the objective functions of potential schedules. The optimization models use a discretization approximation to the calculation of the one-dimensional integral in (14).[1] Furthermore, the benefits of using a large number of discretization points are also less pronounced - with very small improvements in the estimation of the integral and an order of magnitude larger computation time. In some cases, we also consider the generation of moves for local search techniques.

5 The Flowshop Problem

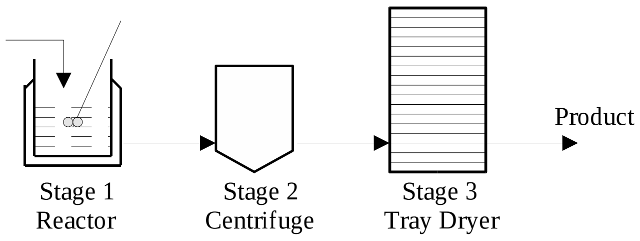

Figure 3: Flowshop Plant

Flowshop plants (see Fig.3) are multiproduct batch plants where the jobs associated with the manufacturing of product orders use the same set of processing units in the same order (Birewar and Grossmann, 1989). A solution to the flowshop scheduling problem specifies the order in which jobs are processed in each unit.

Given are N products $(A, B, \ldots)$, $i \in I$, that are to be manufactured in a flowshop plant with M processing stages $(1, 2, \ldots, M)$, $j \in J$. Each product requires processing in all of the M stages and follows the same sequence of operations, i.e., a permutation flowshop plant. In the case of Unlimited Intermediate Storage (UIS) mode of operation, an unlimited number of batches may be held in storage between stages. The computation of the makespan (of a given processing sequence) for the UIS flowshop plant involves a series of recurrence relations for the completion times ct_{pj} of product in position p in the sequence in stage j. The relations for the completion times for the deterministic case are given below (Rajagopalan and Karimi, 1989):

$$ct_{pj} = \max(ct_{p-1,j}, ct_{p,j-1}) + T_{pj} \qquad \forall p > 1, \forall j > 1 \tag{19}$$

$$ct_{1j} = ct_{1,j-1} + T_{1j} \qquad \forall j > 1 \tag{20}$$

$$ct_{p1} = ct_{p-1,1} + T_{p1} \qquad \forall p > 1 \tag{21}$$

$$ms = ct_{NM} \tag{22}$$

[1]A rectangular approximation of this integral leads to large errors, when the number of discretization points is low; while a piecewise quadratic approximation (Simpson's rule) gives much better results. In the results presented, Simpson's rule was used to approximate the integral in (14).

where T_{pj} is the processing time of the product in position p in the sequence in stage j and ms is the makespan of the corresponding schedule.

For the case where the processing times are represented by fuzzy numbers, the above relations have to be modified using the extension principle. The integral in (14) is approximated by a summation of the function values at specific values of α. Thus, the interval [0,1] is discretized to points $\{a_1, a_2, \ldots, a_A\}$, with stepsize $SS = 1/(A-1)$.

$$ct_{pja} = \max(ct_{p-1,j,a}, ct_{p,j-1,a}) + T_{pja} \qquad \forall p > 1, \forall j > 1, \forall a \tag{23}$$

$$ct_{1ja} = ct_{1,j-1,a} + T_{1ja} \qquad \forall j > 1, \forall a \tag{24}$$

$$ct_{p1a} = ct_{p-1,1,a} + T_{p1a} \qquad \forall p > 1, \forall a \tag{25}$$

$$ms_a = ct_{NMa} \qquad \forall a \tag{26}$$

With the above relations for the completion times of the various operations for a specific processing sequence, we can formulate an MILP model for selecting the sequence with optimal makespan. In this MILP model, we introduce binary variables y_{ip} which denote the assignment of product i to position p in the processing sequence. A sequence with fuzzy makespan X_1 is preferred over a sequence with makespan X_2 if $Z(X_1) < Z(X_2)$, with $Z(X)$ calculated using (14). The MILP model (FSP) is as below:

$$\text{(FSP)} \quad \text{Min } z_{ms} = (SS/3) \cdot (1/2) \cdot \sum_a SC_a \cdot (ct_{NMaL} + ct_{NMaR}) \tag{27}$$

$$ct_{pjae} \geq ct_{p-1,j,a,e} + \sum_i T_{ijae} \cdot y_{ip} \qquad \forall p \geq 2, \forall j, \forall a, e \tag{28}$$

$$ct_{pjae} \geq ct_{p,j-1,a,e} + \sum_i T_{ijae} \cdot y_{ip} \qquad \forall p, \forall j \geq 2; \forall a, e \tag{29}$$

$$ct_{11ae} = \sum_i T_{i1ae} \cdot y_{i1} \qquad \forall a, e \tag{30}$$

$$ct_{1jae} = ct_{1,j-1,a,e} + \sum_i T_{ijae} \cdot y_{i1} \qquad \forall j \geq 2; \forall a, e \tag{31}$$

$$ct_{p1ae} = ct_{p-1,1,a,e} + \sum_i T_{i1ae} \cdot y_{ip} \qquad \forall p \geq 2; \forall a, e \tag{32}$$

$$\sum_p y_{ip} = 1 \qquad \forall i \tag{33}$$

$$\sum_i y_{ip} = 1 \qquad \forall p \tag{34}$$

$$ct_{pjae} \geq 0 \qquad \forall p, j, a, e \tag{35}$$

$$y_{ip} \in \{0, 1\} \qquad \forall i, p \tag{36}$$

where the subscripts e indicate the left and right end-points used in the interval-arithmetic; thus, $e \in E = \{l, r\}$. In (FSP), (27) represents the discrete approximation of the area-compensation integral applied to the makespan, with the SCs denoting the Simpson coefficients used in the approximation of the integral. Constraints (28)-(32) are the generalization of the completion time constraints in (23)-(25) for any sequence. Constraints (33) and (34) are the well-known assignment constraints that specify that each product i should be assigned to only 1 position, and that each position p should be assigned to only one product. Model (FSP) can also be extended to the case where there are set-up times between the processing of different products. These set-up times can either be sequence dependent or unit dependent.

5.1 Illustrative Example

Table 1: TFN Descriptions for 5-Product, 4-Stage Example

Product	Stage 1	Stage 2	Stage 3	Stage 4
1	(16.083, 17, 18.405)	(41.680, 44, 45.036)	(31.369, 32, 34.011)	(21.505, 22, 22.205)
2	(20.056, 22, 23.490)	(16.632, 19, 19.692)	(23.169, 24, 27.099)	(39.606, 44, 48.935)
3	(11.487, 13, 15.002)	(28.634, 30, 31,787)	(49.171, 50, 56,611)	(30,513, 33, 37.988)
4	(48.799, 50, 56,274)	(34.765, 40, 42,271)	(14.403, 15, 15,259)	(34.457, 36, 36.738)
5	(14.575, 16, 18.052)	(17.832, 20, 22.181)	(33.513, 37, 37.233)	(25.122, 27, 31.279)

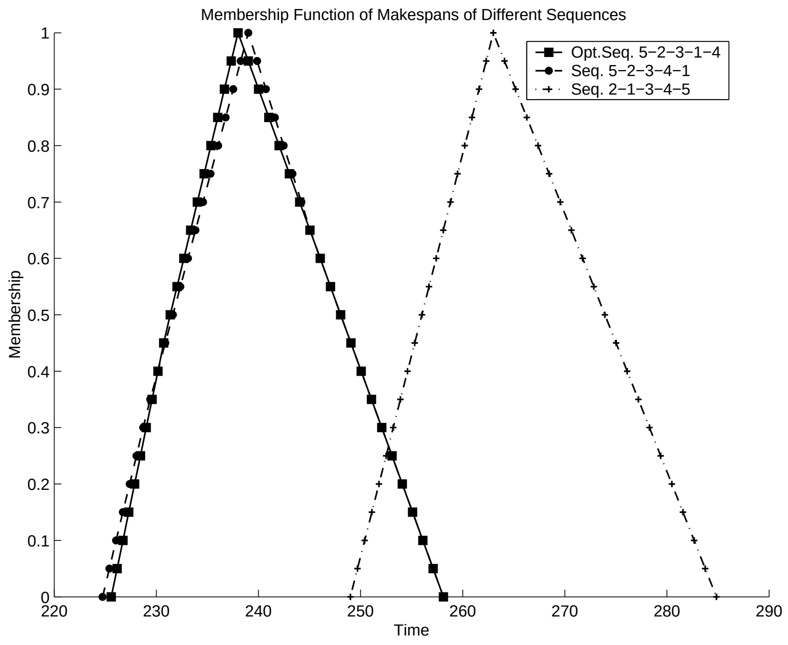

Figure 4: Membership Functions of Makespans

Figure 4 presents the membership functions of the makespans of three processing sequences for a 5-product, 4-stage example. These membership functions are obtained by solving the FSP model and looking up the ct values at the various α-levels. The data for this example can be found in Table 1. The processing times are specified as triplets corresponding to a Triangular Fuzzy Number (TFN) description. The computations were performed with a 21-point discretization. The CPU time for solving the FSP model was under 5 secs. Table 2 summarizes the key results from this example.

Table 2: Results for the Example Problem

Sequence	Makespan (time units)			
	Objective value (AC)	**Optimistic**	**Most Likely**	**Pessimistic**
5-2-3-1-4	239.809	225.59	238	258.108
5-2-3-4-1	239.967	224.734	239	258.108
2-1-3-4-5	264.961	249	263	284.845

The optimal sequence (AC value of 239.809 units) was determined to be '5-2-3-1-4' with a most likely makespan value of 238 units, an optimistic makespan of 225.59 units and a pessimistic makespan of 258.108 units. This information is again obtained by simply looking up the values of the variables ct_{54AL}, ct_{541L} and ct_{541R}. Also shown in Figure 4 is the membership function of the makespan of sequence '5-2-3-4-1': note that although this sequence has a better optimistic makespan than sequence '5-2-3-1-4', it is not optimal with respect to the AC of the makespan, with a marginally higher AC value. Indeed, if we change the objective function so that we want a sequence for optimal optimistic makespan, we get sequence '5-2-3-4-1' as optimal. If we wish to find the sequence with minimum pessimistic makespan, we obtain either of the above two sequences since both of them have the minimum pessimistic makespan of 258.108 units. Finally, sequence '2-1-3-4-5' is sub-optimal with respect to all measures - AC value, optimistic, most likely and pessimistic makespans. Clearly, sequences '5-2-3-1-4' and '5-2-3-4-1' are preferrable over sequence '2-1-3-4-5' on two counts: 1) in terms of the AC values, and 2) in terms of the spread of the makespan. While the makespan of sequence '5-2-3-1-4' has a spread of 32.518 units (i.e., 258.108-225.59), sequence '2-1-3-4-5' has a spread of 35.845 units.

5.2 Computational Results

In the following examples, all MILP models were formulated using GAMS and solved using CPLEX 7.0 on a Pentium III/930 MHz machine running Linux. Table 3 displays the solution times required for model (FSP) when applied to problems of differing sizes. In these examples, the durations were assumed to be given by asymmetric Triangular Fuzzy Numbers (TFNs). The results are from several examples, with the TFNs randomly generated. In these models, a 21-point discretization was used for evaluating the integral in (14). As can be seen, the computational times required are quite small. In the worst case about half an hour was required for one of the 12-product, 4-stage problems. Note that the 12-product examples contain as many as 48 tasks with uncertain durations, which would pose a formidable dimensionality problem to a probabilistic model.

Table 3: Computational Times for Flowshop Problems

Products	**Stages**	**CPU Time (secs)**		
		Min.	**Mean**	**Max.**
5	4	4	4.5	6
8	4	27	101	200
	5	160	178	237
12	3	8	31	122
	4	296	788	1598

5.3 Remarks - Local Search Techniques

For larger problems where the number of products to be processed is quite large, say 20-50 products, solving model (FSP) to optimality can be prohibitive. In these cases, the use of a local search algorithm such as Simulated Annealing (Kirkpatrick *et al.*, 1983; Aarts and Korst, 1989) or Tabu Search (Glover, 1990) may be preferrable, since these algorithms can obtain good quality solutions within reasonable time. However, these algorithms also have significant drawbacks - they do not provide any guarantee on the quality of the solution obtained, and it is often impossible to tell how far the current solution is from optimality.

In this section, we outline results from a Tabu Search implementation for flowshop optimization. We chose Tabu Search for the advantages that it offers over Simulated Annealing and Genetic Algorithms. Tabu Search is the most deterministic of the three techniques and also has fewer tunable parameters. We implemented a variant of Tabu Search called Reactive Tabu Search (RTS) (Battiti and Tecchiolli, 1994; Schmidt, 1998) which has the desirable property that the length of the tabu list is dynamically adjusted during the progress of the algorithm. The dynamic adjustment of the tabu list helps to automatically intensify the search in promising regions or diversify the search in unexplored regions. RTS stores all solutions (through the use of hashtables, digital search trees etc.) as they are visited during its progress, so that it is possible to recognize whether a particular solution has been encountered before. The frequency with which solutions reappear indicates whether the search should be intensified or diversified. RTS also assumes that if a number of solutions have been encountered several times, the search has been trapped in a basin of attraction. Consequently, a number of random moves are made to escape from the current region. There are a few tunable parameters used by the RTS algorithm that relate to the threshold for a ”frequently seen” solution, the number of frequently seen solutions before performing the escape sequence, the factors by which the tabu list size is altered etc. The reader is referred to Battiti and Tecchiolli (1994) for a more detailed description of the RTS algorithm.

For the flowshop problem, a permutation string of the products is a natural characterization of the solution. A move was defined to be any pair-wise interchange of the products in the permutation. Thus the neighborhood of allowable moves from a solution is of size $O(N^2)$. Furthermore, the computation of the makespan of a given solution can be done in $O(N \cdot M \cdot A)$, where A is the number of α-levels at which the calculation is performed. The worst case complexity of one RTS iteration is determined by the evaluation of the makespans (an $O(N \cdot M \cdot A)$ operation) of all the $O(N^2)$ neighbors. Thus, the complexity of one RTS iteration is $O(N^3 \cdot M \cdot A)$. Figure 5 shows the CPU time required for 1 RTS iteration as a function of the problem size (number of products) and the number of α-levels at which the computation is performed. The CPU time for one move when the makespan evaluations are carried out at 101 α-levels for 3-stage problems is given by (37), where N denotes the number of products. This relation was obtained by regression. The CPU time reported is for a Java (Arnold and Gosling, 1998) implementation on a Pentium III machine running Linux.

$$t_{101} = 3.847 \cdot 10^{-4} N^3 - 8.588 \cdot 10^{-3} N^2 + 0.09315 N - 0.30186 \tag{37}$$

Figure 6 displays the initial progress of the RTS for a 20-product, 4-stage problem with sequence-dependent set-up times on all stages. The evaluation of a solution was performed with a 21-point discretization. In the initial stages, the algorithm behaves much like a steepest descent heuristic where there is a rapid decrease in the objective function. Surprisingly enough, even in this early stage of the algorithm, the escape sequences (sequences of random moves which seek to escape the region of a complex attractor and take the search into a new, unexplored region) are performed, as indicated by the spikes in Figure 6. The search then proceeds to locate the local minimum in the new search area. Figure 7 displays the long-term progress of the RTS over the course of 5000 iterations. One can clearly observe the two different time-scales in the search. The solution varies rapidly over a short time horizon, and the escape sequences move the search into unexplored

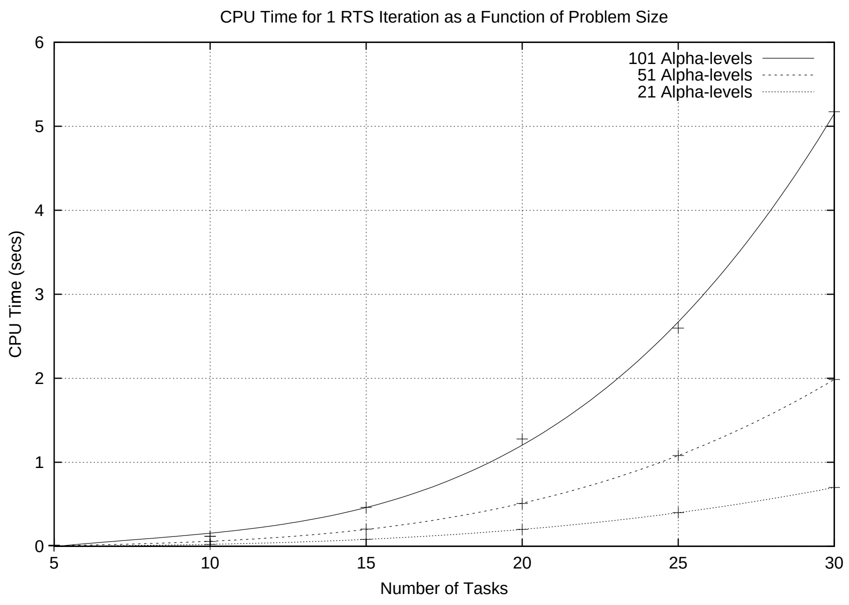

Figure 5: RTS Iteration Time

regions over the longer time horizons. The search was terminated at the end of 5000 iterations and the best solution was found at iteration 863, with an objective value of 303.005. The total CPU time required for this problem was around 1400 secs, thus averaging about 0.28 secs per RTS iteration.

An indication of the quality of the solution can be obtained by solving the corresponding FSP model to optimality. However, when we attempted to solve the 21-point FSP model for this particular problem with GAMS/CPLEX, we could not even find an integer feasible solution in 1400 seconds. The LP relaxation of the objective function for this problem was 280.3036, which indicates that the RTS has done quite well, coming within 8% of the best possible solution in the same time (1400 secs). The arbitrary termination criteria for the RTS implies that the CPU time can be adjusted as desired. Furthermore, even after 50,000 CPU secs, CPLEX could not solve the 21-point FSP model to optimality - the best solution obtained was 295.3863 and the gap was 4.5% (the best node had a value of 282.0548). Thus, we can note that solving these MILP models to optimality can be computationally expensive. Consequently, the use of local search algorithms such as RTS that yield near-optimal solutions in significantly smaller computational times is clearly advantageous.

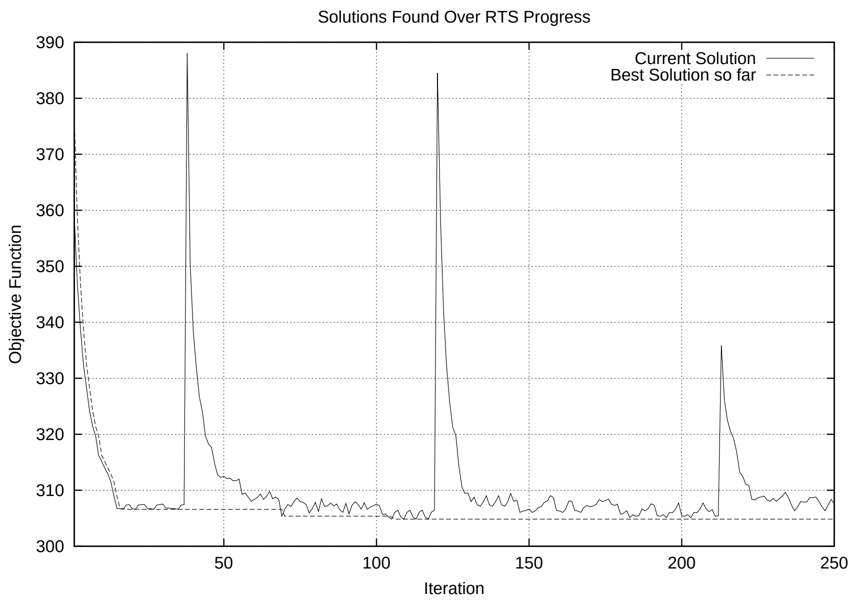

Figure 6: Initial Progress of RTS

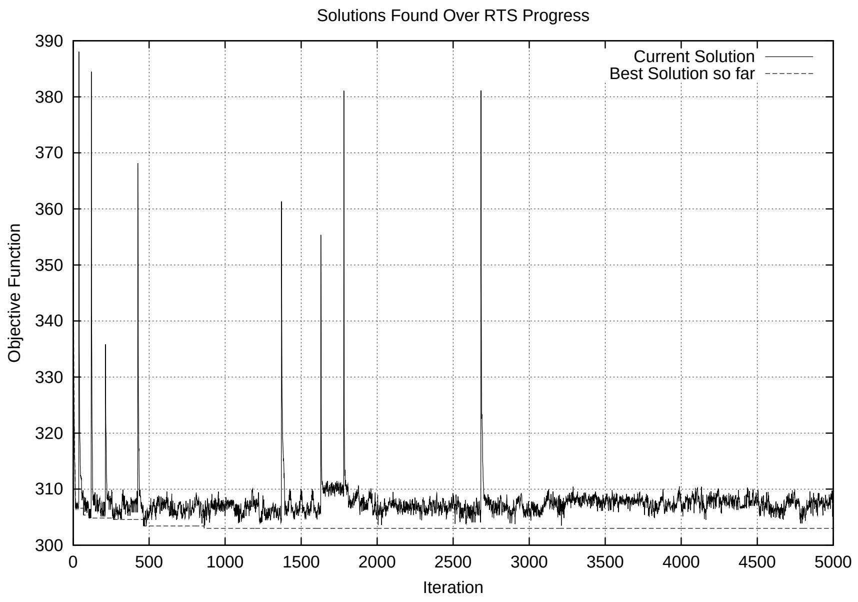

Figure 7: Long-term Progress of RTS

6 Multistage Flowshop with Parallel Units

6.1 MILP Model

In this section, we address the problem of optimizing the scheduling of multistage flowshop plants which have parallel units in several stages (see Fig. 8) - an extension of two classical scheduling

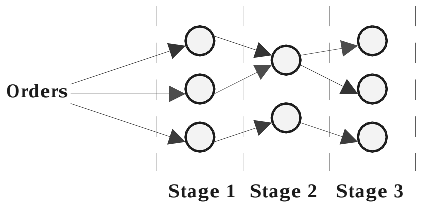

Figure 8: Multistage Flowshop with Parallel Units

problems (flowshop and parallel machines problems). Given a production horizon with due dates for the demands of several products, the key decisions that are to be made are the assignment of the products to the units as well as the sequencing of products that are assigned to the same unit. Each product is to be processed only once by exactly one unit of every stage that it goes through. The objective is usually minimizing the makespan or the total lateness of the orders. We present a MILP model for obtaining the schedule with minimum total lateness, when the processing times of the orders are given by fuzzy numbers. The assumptions involved in the model are:

1. The processing times of the various products in the units are given by fuzzy numbers.
2. Due dates are deterministic.
3. Transition times are deterministic and only equipment dependent.
4. There is unlimited intermediate storage between stages.

Assumption 1 seems reasonable, especially when only estimates of the processing times can be obtained, for instance, when new products are being processed for the first time. Relaxing assumption 2 will change the nature of the problem. Transition times can also be made fuzzy, without much problem - however, introducing sequence dependent transitions will considerably complicate the model.

Much work has been done on deterministic versions of this problem of scheduling flowshop plants with parallel units. Pinto and Grossmann (1995) presented a continuous-time MILP formulation for minimizing the total weighted earliness and tardiness of the orders. This formulation was a slot-based model which allocated units and orders on two parallel time coordinates and matched them with tetra-indexed (stage-order-slot-unit) variables. A later paper by the same authors (Pinto and Grossmann, 1996) incorporated pre-ordering constraints in the model to replace the tetra-indexed variables by tri-indexed variables (order-stage-unit) and thus reduce the computational effort required to solve the model. McDonald and Karimi (1997) developed an MILP formulation for the short-term scheduling of single-stage multi-product batch plants with parallel semi-continuous processing units. More recently, Hui *et al.* (2000) presented an MILP formulation which uses tri-indexed variables (order-order-stage) to minimize the total earliness and tardiness of the orders. With the elimination of the unit index, the formulation required fewer binary variables and reduced computational time.

For our problem where the processing times are given by fuzzy numbers, we present an MILP model which is a generalizations of the tri-indexed model (Hui *et al.*, 2000). We also formulated an MILP model (M1) using the slot-based approach of Pinto and Grossmann (1995), but due to space restrictions, we restrict our presentation to the tri-indexed model (M2). As before, the timing computations are performed for different α-levels and the objective is a discrete approximation of the area compensation integral applied to the total lateness.

6.2 Tri-indexed model - (M2)

Given are products (orders) $i \in I$ and processing stages $l \in L$ which have processing units $j \in J_l$. Each product i has to be processed on stages L_i, and can be processed only on units J_i. The parameters of interest are the processing times T_{ijae}, transition times SU_j and due dates D_i.

The model presented in Hui *et al.* (2000) uses tri-indexed binary variables $x_{i_1,i_2,l}$ to indicate if product i_1 is processed before product i_2 in stage l. Binaries s_{ij} are used to represent the assignment of product i to the first processing position in unit j, while continuous variables w'_{ij} are used to represent assignment of product i to unit j. Here we present the model for makespan minimization. Continuous variables ts_{ilae} and te_{ilae} respectively denote the start and end times of product i in stage l at α-level a.

$$\text{(M2)} \quad \min \; Z_{ms} = (SS/3) * 0.5 * \sum_a SC_a * (ms_{aL} + ms_{aR}) \tag{38}$$

$$\text{s.t.} \quad w'_{ij} + \sum_{j' \in (J_i \cap J_l)} w'_{i'j'} + x_{ii'l} \leq 2 \qquad \forall i, i', i \neq i', j \in J_i \cap J'_i \cap J_l, l \in L \tag{39}$$

$$\sum_{i' \in I_l, i' \neq i} x_{ii'l} \leq 1 \qquad \forall i \in I, l \in L \tag{40}$$

$$\sum_{i' \in I_l, i' \neq i} x_{ii'l} + \sum_{j \in (J_i \cap J_l)} s_{ij} = 1 \qquad \forall i \in I, l \in L \tag{41}$$

$$w'_{ij} \geq s_{ij} \qquad \forall i \in I, j \in J_i \tag{42}$$

$$\sum_{i \in I} s_{ij} \leq 1 \qquad \forall j \in J_i \tag{43}$$

$$\sum_{j \in (J_i \cap J_l)} w_{ij} = 1 \qquad \forall i \in I, l \in L \tag{44}$$

$$ts_{ilae} + \sum_{j \in (J_i \cap J_l)} w'_{ij} \cdot (T_{ijae} + SU_j) \leq ts_{il'ae} \qquad \forall l', l \in L, l' > l, i \in I, a, e \tag{45}$$

$$ts_{ilae} + \sum_{j \in (J_i \cap J_l)} w'_{ij} \cdot (T_{ijae} + SU_j) \leq (1 - x_{ii'l}) \cdot U + ts_{i'lae} \qquad \forall i, i' \in I, i \neq i', l \in L, a, e \tag{46}$$

$$ts_{i\bar{l}_i ae} + \sum_{j \in (J_i \cap J_l)} w'_{ij} \cdot (T_{ijae} + SU_j) \leq ms_{ae} \qquad \forall i \in I, a, e \tag{47}$$

$$w'_{ij} \geq 0 \qquad \forall i, j \in J_i \tag{48}$$

$$w'_{ij} \leq 1 \qquad \forall i, j \in J_i \tag{49}$$

$$x_{ii'l} \in \{0,1\} \qquad \forall i, i' \in I, i \neq i', l \in L_i \tag{50}$$

$$s_{ij} \in \{0,1\} \qquad \forall i \in I, j \in J_i \tag{51}$$

$$ts_{ilae} \geq 0 \qquad \forall i, l, a, e \tag{52}$$

(53)

Constraints (39) specify that if orders i and i' are consecutive orders and order i is assigned to unit j, then order i' is also assigned to unit j. Constraints (40) and (41) ensure that each order i has at most one successor and one predecessor respectively in each stage of processing, while (42) specifies the relation between the assignment and s variable. Note that with constraints (39) and (42), the assignment variables can be relaxed as continuous variables with 0 and 1 as lower and upper bounds (for proof, please refer to Hui *et al.*, 2000). Constraints (43) state that each unit processes at most one starting order and (44) state that every order has to be processed by a unique unit in each stage. Constraints (45), (46) and (47) govern the relationships between the starting times of an order in successive stages, the starting times of successive orders in the same stage and the computation of the makespan respectively. The objective (38) is to minimize the area compensation of the makespan - this ensures that the makespan of the schedule is the largest end time of all orders on the final stage.

6.3 Computational Results

Unlike the case of the flowshop problem discussed in Section 5, the MILP models for the multi-stage flowshop with parallel units do not exhibit good computational performance as the problem size increases (see Table 4). This is primarily due to the large number of binary variables that are required for modeling this problem. Although the number of binary variables can be reduced by taking in to account the forbidden assignments and postulating an appropriate number of slots for the various units, it increases dramatically with problem size as can be seen from Table 4. The solution times reported are for makespan minimization.

Model (M2) was a better formulation than (M1), as can be seen from the reduced computational times required for the solution (except for the 10-product, 15-units problem). Both models had the same LP relaxation. Solution of model (M1) for the 10- and 12-product problems was terminated after 50,000 CPU secs. Model (M2) for the 10-product (25 units) problem was solved in about 4000 CPU secs, whereas for model (M1), the optimality gap was 11.7% even after 50,000 CPU secs. For the 12-product problem, both models could not be solved to optimality, although for

(M2) the optimality gap was smaller than for (M1) after 50,000 CPU secs. The optimality gap for the 10- and 12-product problems with 25 units is smaller than that for the 10-product problem with 15 units mainly because the 25-unit problems had many assignments that were forbidden, resulting in more structure for the solution. Even then, these MILP models require excessive computational resources, as can be seen from the CPU times.

Table 4: Model Characteristics for Parallel Flowshop Problems

N-L-(Units in Stgs)	Binaries		CPU secs		Optimality Gap (%)	
	M1	M2	M1	M2	M1	M2
5-5-(6,3,10,3,3)	241	225	15	18	0	0
10-5-(6,3,10,3,3)	721	700	$> 50{,}000$	4086	11.7	0
10-5-(3,2,3,4,3)	730	700	$> 50{,}000$	$> 50{,}000$	53.5	56.95
12-5-(6,3,10,3,3)	1000	960	$> 50{,}000$	$> 50{,}000$	14.5	6.5

The lateness minimization MILP models also exhibit similar computational performances. Recall that, for the earliness minimization problem, Pinto and Grossmann (1995) developed a decomposition strategy which determines feasible assignments that minimize in-process time and then further determines a minimum-earliness schedule that eliminates unneccesarry set-ups. They also used pre-ordering constraints to obtain near-optimal solutions. However, an extension of these ideas to the case of uncertain processing times is not straightforward. Thus, a local search technique such as the RTS is particularly relevant for solving larger instances of these problems.

6.4 RTS Implementation

In this section, we discuss some the issues that are critical to the success of the RTS implementation for this problem.

6.4.1 Solution Characterization and Evaluation

A solution for the multistage flowshop with parallel units is characterized by a feasible allocation and sequencing of all the products to different processing units in all the stages. One possible representation of a solution is a set of $\sum_{l \in L} \mid J_L \mid$ strings, with each string denoting the sequencing of the products allocated to the unit. Of course, the allocation of the products to the various units must satisfy the constraints that forbid certain allocations as well as (39)-(44). Figure 9 illustrates one possible solution for a problem where 4 products are to be processed in 2 stages, where the first stage has 2 units in parallel and the second stage has only 1 unit. Thus, product A is processed first on unit 1 in stage 1 and is followed by B, while C and D are processed on unit 2 in stage 1. The processing sequence on the unit in stage 2 is C-D-B-A.

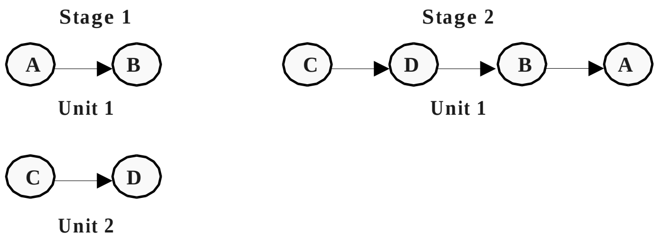

Figure 9: Solution Representation - Example

Once we have a feasible allocation and sequencing of the orders in various units, the start and end times of the products in the stages can be evaluated with expressions similar to (45) and (46). The makespan may be calculated as in (47) and the area compensation applied to the fuzzy makespan to represent the objective value. The makespan of a given solution may be computed in $O(N \cdot M \cdot A)$ time.

6.4.2 Neighborhood of a Solution

We consider insertion moves, where we select a product p_1 on a unit in stage l_1 and insert it in another position on the same unit or any other unit on the same stage. Clearly, then the neighborhood defined by the insertion move is of size $O(N^2 \cdot M)$, since l_1 can be chosen in M ways, p_1 can be chosen in N ways, and the position of insertion can be chosen in $O(N)$ ways. Since the computation of the makespan of a given solution takes $O(N \cdot M \cdot A)$ time, the evaluation of the complete insertion neighborhood takes $O(N^3 \cdot M^2 \cdot A)$ time. Figures 10 and 11 display two possible insertion moves from the solution in Figure 9 for the 4-product 2-stage example discussed earlier. Figure 10 shows product A being removed from Unit 1 on Stage 1 and inserted ahead of products C and D on Unit 2 in the same stage, while Figure 11 shows the insertion of product D at the end of the processing sequence on Unit 1 in Stage 2.

The computational complexity of evaluating the complete insertion neighborhood was confirmed by tests on a number of problems, with the result that evaluating the entire insertion neighborhood was deemed too time-consuming. As a result, we used a reduced neighborhood based on the insertion move. The gains in computational efficiency for the reduced neighborhood in comparison with the entire neighborhood are quite significant. For instance, the CPU time for one iteration of a 25-product, 5-stage, 15-machine problem with a full evaluation of the insertion neighborhood

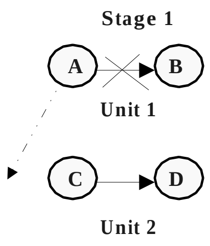

Figure 10: **Insertion Move - Different Units**

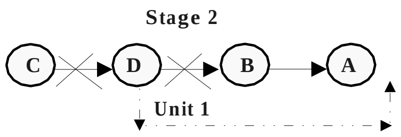

Figure 11: **Insertion Move - Same Unit**

was 4.9 secs, while for the reduced neighborhood implementation, the CPU time for one iteration was 0.50 secs. Please refer to Appendix A for a discussion of the reduced neighborhood.

6.4.3 Computational Results

To see how well the RTS algorithm performs on the multistage parallel flowshop problems, the 10-product, 5-stage, 15-units problem solved in Section 6.3 was solved with a limit of 50000 iterations (about 2800 CPU secs).

Recall that for model (M1), even after 50,000 CPU secs of solution time, there was an optimality gap of over 50% (i.e., the best solution possible was 113.38 and the best solution obtained was 250.086), while for (M2), the best solution obtained had an objective value of 280.932. It is likely that the large number of binary variables and the Big-M constraints in the MILP model result in there being such poor LP relaxations: thus, the best solution possible may have a significantly larger objective value than that indicated when the branch and bound algorithm terminated. Indeed, fixing the ordering of a subset of products on even one unit leads to much better bounds on the objective.

Using the RTS algorithm, the best solution (obtained in 2800 CPU secs) had a makespan of 182.878. Note that the RTS algorithm obtained a much better solution in a fraction of the time required by the mathematical programming approach. The key observation that we can derive from this example is that local-search techniques such as the RTS algorithm are well suited for solving the parallel flowshop problems since they are able to provide good quality solutions with reduced computational requirements. However, MILP models are not completely useless for these problems, since they can be used to derive bounding information that indicate the quality of the solutions obtained using the local search techniques.

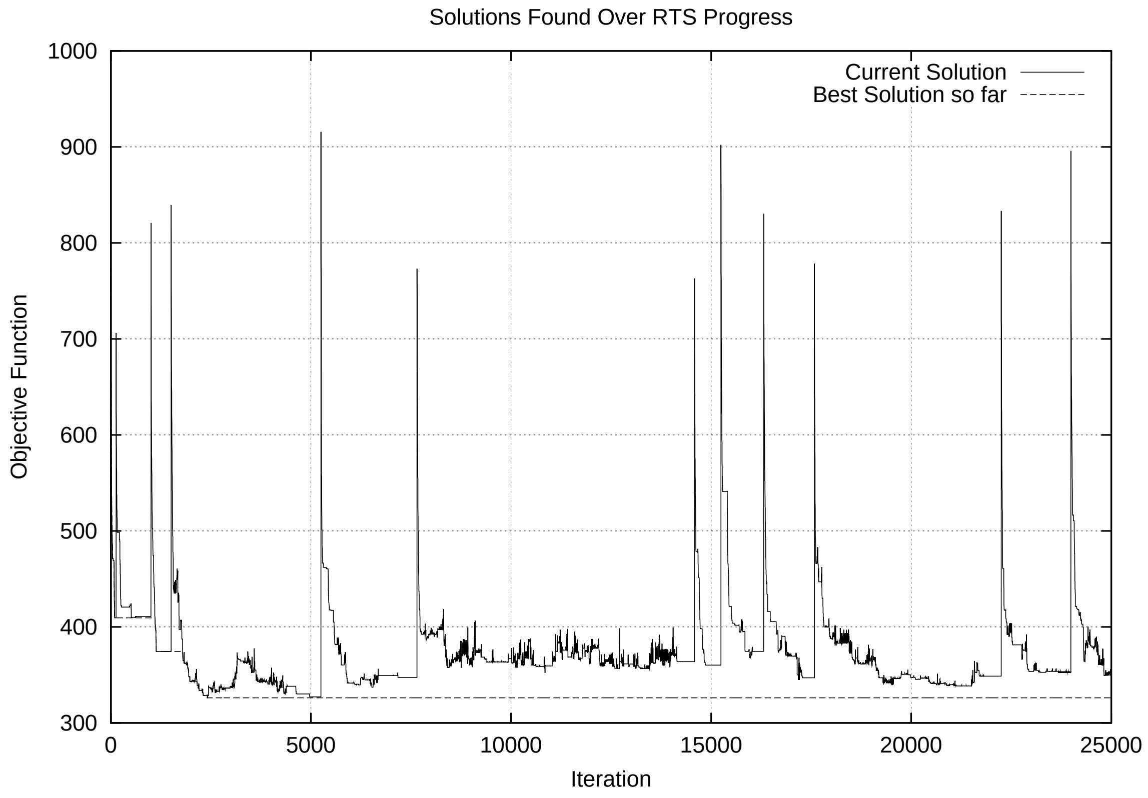

Figure 12: RTS Progress

Another advantage with techniques such as RTS is that they usually scale quite well with problem size. Often, the iteration complexity grows only polynomially with problem size. Figure 12 displays the progress of an RTS implementation for a 25-product, 5-stage, 15-machine problem. The RTS algorithm used the reduced neighborhood of the insertion move, as described earlier; the CPU time for 25,000 iteration was 11,600 secs, thus averaging under 0.5 secs per iteration. Clearly then, our RTS implementation shows promise for tackling even bigger problems. The starting solution which was obtained by a random balanced allocation of the jobs to the various units in the stages had an objective value of 776.197 units. The best solution (shown below) obtained had a value of 326.236 units and was obtained at iteration 2408. Thus, on unit 1 in stage 1, product P_9 is processed first, and is followed by products P_3, P_{16} etc. It is instructive to note from Figure 12 that this solution was obtained soon after one of the random escapes was performed, thereby validating the utility of the escape sequences in the RTS algorithm.

Best Solution obtained from the RTS algorithm

Stage 1

Unit 1: $P_9 \to P_3 \to P_{16} \to P_{13} \to P_5 \to P_{19} \to P_{18} \to P_0 \to P_{24} \to P_{14}$

Unit 2: $P_7 \to P_{11} \to P_1 \to P_{12} \to P_2 \to P_{21}$

Unit 3: $P_8 \to P_{17} \to P_{10} \to P_{22} \to P_6 \to P_{23} \to P_4 \to P_{20} \to P_{15}$

Stage 2

Unit 1: $P_7 \to P_3 \to P_{11} \to P_{10} \to P_6 \to P_{13} \to P_{22} \to P_{23} \to P_{20} \to P_4 \to P_{15} \to P_{14}$

Unit 2: $P_9 \to P_8 \to P_{17} \to P_{16} \to P_{12} \to P_1 \to P_5 \to P_{19} \to P_0 \to P_{24} \to P_{21} \to P_{18} \to P_2$

Stage 3

Unit 1: $P_7 \to P_3 \to P_{11} \to P_1 \to P_{10} \to P_{19} \to P_0 \to P_4 \to P_{14}$

Unit 2: $P_{16} \to P_6 \to P_5 \to P_{24} \to P_{18}$

Unit 3: $P_9 \to P_8 \to P_{17} \to P_{12} \to P_{13} \to P_{23} \to P_{20} \to P_{22} \to P_{15} \to P_{21} \to P_2$

Stage 4

Unit 1: $P_1 \to P_5 \to P_{20} \to P_0 \to P_4 \to P_{14}$

Unit 2: $P_{17} \to P_{24}$

Unit 3: $P_3 \to P_{16} \to P_{12} \to P_{13} \to P_{23} \to P_{21} \to P_2$

Unit 4: $P_9 \to P_8 \to P_7 \to P_6 \to P_{11} \to P_{10} \to P_{19} \to P_{22} \to P_{15} \to P_{18}$

Stage 5

Unit 1: $P_7 \to P_1 \to P_{12} \to P_{20} \to P_{22} \to P_{21} \to P_2$

Unit 2: $P_{11} \to P_{13} \to P_{10} \to P_{19} \to P_0 \to P_{18} \to P_{15}$

Unit 3: $P_9 \to P_8 \to P_{17} \to P_{16} \to P_3 \to P_6 \to P_5 \to P_{23} \to P_{24} \to P_4 \to P_{14}$

Figure 13 displays the variation of the size of the tabu list over the course of the RTS. The tabu list size typically oscillates between 5 and 60 as the search varies between intensification and diversification.

Although we have only illustrated the RTS implentation for the makespan minimization problem, extending it to lateness minimization is a straightforward task, requiring only minor changes in the computations required at each iteration.

7 The New Product Development Problem

In the NPD process, the potential product must pass a series of tests that assess its safety, efficacy and environmental impact. The costs, durations and probabilities of success of various testing tasks are often not known with certainty when the schedule has to be constructed. The optimization of the testing schedule can be conceived as introducing precedence constraints between the tests, in addition to the given set of technological precedence constraints. There is an inherent cost-income trade-off associated with the scheduling of the NPD process. This trade-off is between the greater

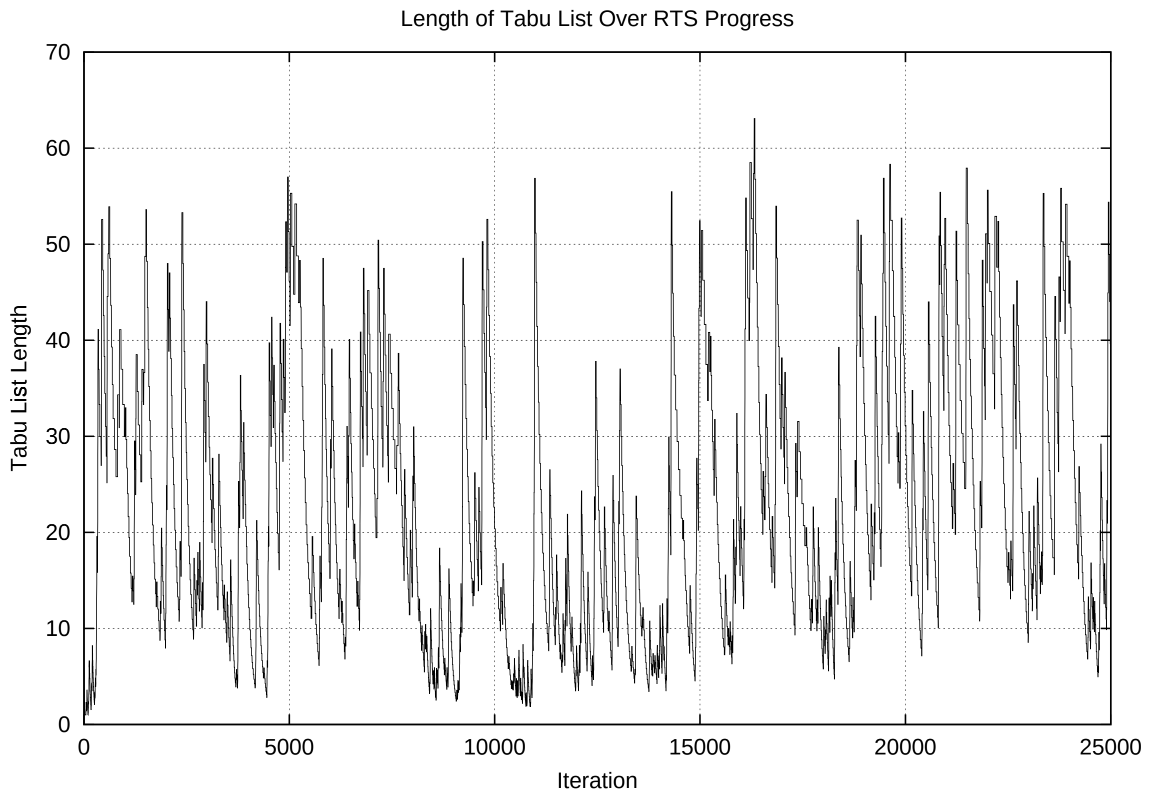

Figure 13: **Tabu List Size**

income resulting from a shorter schedule (where many of the tests are performed in parallel) and the lower expected value of the total cost from a longer sequential schedule.

We consider the problem of optimizing the schedule of testing tasks while taking into account the uncertainties in the durations and success of the tasks, a problem which has not been reported before. In previous work, Schmidt and Grossmann (1996) developed an MILP model assuming known durations. These authors also considered the evaluation of the distribution of the completion time for a fixed schedule, given uncertainties in durations (Schmidt and Grossmann, 2000). The extension of their method to optimization is non-trivial, given the complexity of their procedure for evaluating the completion time with uncertain durations.

To effectively address the problem, we use a two-level description of the uncertainties inherent in the NPD process. We retain the probability description of the success of the tests, i.e., for every test there is a given probability that the product will pass the test. However, instead of assuming the discrete/continuous probability distributions that Schmidt and Grossmann (2000) used to model the uncertainty in test durations, we draw upon concepts of fuzzy set theory to represent the imprecision and uncertainty of information in the time parameters of the various activities. Thus, the processing times are represented by intervals or fuzzy numbers instead of deterministic values or probability distributions. For a given schedule, the completion time is computed through interval arithmetic by the extension principle. This computation can be performed efficiently, in contrast to the case when we use a probabilistic description of the testing durations. The expected value of the cost of a schedule is evaluated using the probabilistic framework, and associated with this successful schedule is a fuzzy description of the project's completion time. This completion time is then translated into an equivalent income description, thereby completing the evaluation of a given schedule in terms of the expected cost and the completion time (income).

The optimization of the schedule considers as the objective the difference of the expected value of the fuzzy income and the expected cost. Formulation (PDP) represents a MILP model for selecting the optimal schedule under task uncertainty. It is assumed that the reader is familiar with the deterministic version of the problem (Schmidt and Grossmann, 1996).

$$
\begin{array}{rcllr}
\text{(PDP)} \quad \text{Max } profit & = & MI - \sum_i C_i \cdot \sum_n e^{A_{in}} \cdot \lambda_{in} & & \\
 & & -(SS/3) \cdot (1/2) \cdot \sum_m ID_m \cdot \sum_a SC_a \cdot (u_{maL} + u_{maR}) & & (54) \\
ts_{iae} + T_{iae} & \leq & ct_{a,e} & \forall i, a, e & (55) \\
B_m + u_{mae} & \geq & ct_{a,e} & \forall m, a, e & (56) \\
ts_{iae} + T_{iae} & \leq & ts_{jae} + U_{iae} \cdot (1 - y_{ij}) & \forall (i,j) | i \neq j ; \forall a, e & (57) \\
y_{ij} & = & 1 & \forall (i,j) \in PREC & (58) \\
\sum_n A_{in} \cdot \lambda_{in} & = & \sum_{j|(j \neq i)} ln(P_j) \cdot y_{ji} & \forall i & (59) \\
\sum_n \lambda_{in} & = & 1 & \forall i & (60) \\
y_{ij} + y_{ji} & \leq & 1 & \forall (ij) | i < j & (61) \\
y_{ij} + y_{jl} + y_{li} & \leq & 2 & \forall (i,j,l) | (i < j < l) & (62) \\
y_{ji} + y_{il} + y_{lj} & \leq & 2 & \forall (i,j,l) | (i < j < l) & (63) \\
ct_{ae} & \geq & 0 & \forall a, e & (64) \\
u_{mae} & \geq & 0 & \forall m, a, e & (65) \\
ts_{iae} & \geq & 0 & \forall i, a, e & (66) \\
0 \leq & \lambda_{in} & \leq 1 & \forall i, n & (67) \\
y_{ij} & \in & \{0, 1\} & \forall i, j & (68)
\end{array}
$$

In formulation (PDP), the precedence relationship between two tasks i and j is modeled with the help of the binary variable y_{ij}. Thus $y_{ij} = 1$ if task i precedes task j, else $y_{ij} = 0$. In (PDP), (54) represents the objective function, which is calculated as the difference of the maximum possible income, MI, and the cost of the schedule and the total decrease in income due to delay in product release. Since the durations of the various testing tasks are given by fuzzy numbers, the schedule completion time and the decrease in income are also fuzzy numbers. The mean value of the total decrease in income as calculated by the area compensation integral is included as part of the objective function. The SCs denote the Simpson coefficients used in the approximation of the integral; B_m denote the points in time when the income decrease rate ID_m changes and P_j indicates the probability of success of task j. Constraints (55) forces the completion time, ct, (at every α-level) of the schedule to be greater than the completion time of each task (at every α-level), while (56) aid in the calculation of the excess time over B_m. The timing relationship between two tasks i and j is modeled using the Big-M formulation in (57), which specifies that either task j starts after i is complete, or task j starts before i, or the relationship between the two tasks is unspecified. Constraints (58) fix the technological precedence relationships (given by set $PREC$) between certain tasks i and j, while (59) and (60) approximate the exponential in calculating the cost term. Finally, (61)-(63) are logic cuts and valid inequalities that eliminate directed cycles of length 2 or 3, that strengthen the relaxation of the MILP. We now present two examples that illustrate the performance of model (PDP).

7.1 The 9-Task Example

The data for this example are taken from Schmidt and Grossmann (1996) and are modified to reflect the uncertainty in testing durations. In particular, the testing success probabilities and most likely values of the durations are the same as those in the above paper. However, the durations are assumed to be given by asymmetric TFNs, with a left spread of 5% and a right spread of 20%. Figure 14 displays the technological precedences for this problem, while Figure 15 displays the optimal schedule when the computations are performed with 20 intervals (21 points). Computational results from two discretizations are shown in Table 5. Both models provide the same optimal solution, although the 21-pt discretization model provides a better estimate of the mean value of the profit. The number of equations and continuous variables in the 21-pt model is 160% and 85% more than the 7-pt model and the solution time is an order of magnitude larger. However, the improvement in the estimation of the income is less than 1%, which does not really warrant the order of magnitude increase in the computational time.

7.2 The 65-Task Example

The task data for this example are taken from Schmidt (1998) and are modified to reflect the uncertainty in testing durations. The durations are assumed to be given by asymmetric TFNs, with a left spread of 5% and a right-spread of 20% and with the central value at the data-point in

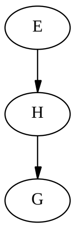

Figure 14: Technological Precedences

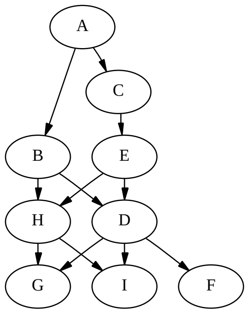

Figure 15: Optimal Schedule

Table 5: Computational Results for the 9-Task Example

Model Characteristics	7-pt Model	21-pt Model
Equations	1402	3754
Variables	492	856
Binaries	70	70
CPU secs	601	7611
Schedule Duration (as TFN)	(19, 20, 24)	(19, 20, 24)
Schedule Cost ($1000s)	1033.6445	1033.6445
Income Loss (Mean) ($1000s)	1156.9444	1120
Profit ($1000s)	3809.4110	3846.355

Schmidt (1998). The maximum income is $25 million and the income decrease is over a period of 72 months. Figure 16 displays the technological precedences between the various tasks, while Figure 17 shows the optimal schedule obtained by solving the MILP with a 7-pt discretization. The other models with finer discretization provided the same optimal solution, although with better estimates of the objective function. These models solve comparatively faster than the smaller 9-task example because there are a number of technological precedence constraints (see Figure 16), which leads to more structure in the schedule. The number of equations and continuous variables in the 21-pt model is 80% and 70% more than the 7-pt model and the solution time is almost an order of magnitude more. The only benefit of using the 21-pt model is a more accurate estimation of the duration, income (2% higher than the 7-pt model). Similarly, the 51-pt discretization model provides less than a 1% improvement in the estimation of the income loss when compared with the 21-pt model, although the solution time has increased by a factor of 3.

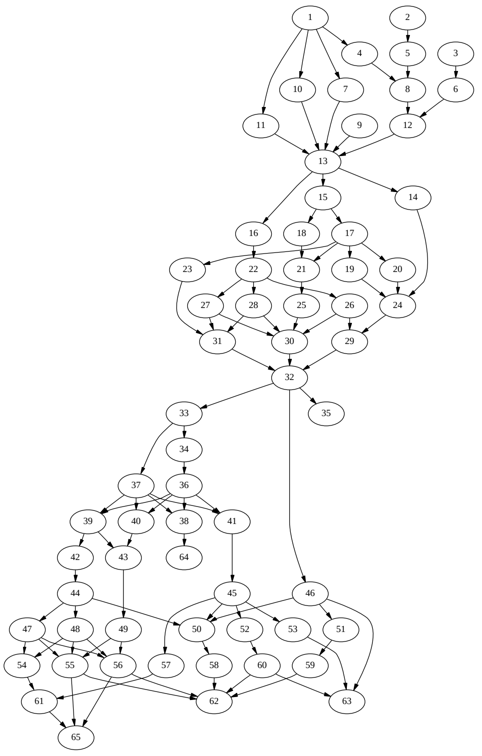

Figure 16: Technological Precedences

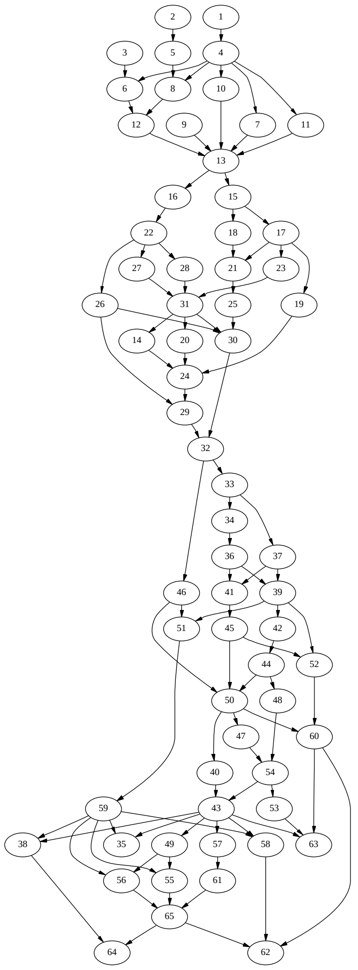

Figure 17: Optimal Schedule

Table 6: Computational Results for the 65-Task Example

Model Characteristics	7-pt Model	21-pt Model	51-pt Model
Equations	148780	267192	520932
Variables	6834	8794	12994
Binaries	4061	4061	4061
CPU secs	324	2786	8909
Schedule Durn (TFN)	(58.11,61.17,73.4)	(58.11,61.17,73.4)	(58.11,61.17,73.4)
Schedule Cost ($1000s)	3147.2161	3147.2161	3147.2161
Income Loss (Mean) ($1000s)	8487.9514	8194.8833	8119
Profit ($1000s)	13364.8326	13657.9006	13732.86

8 Conclusions

In this paper we have applied a non-probabilistic approach to the treatment of processing time uncertainty in two problems - the scheduling of (i) flowshop plants and (ii) the new product development process. The benefits of using the non-probabilistic approach are as follows:

1. Concepts such as fuzzy sets, interval arithmetic allow us to model uncertainty in cases where historical (probabilistic) data are not readily available. These models are fairly flexible in their description of uncertainty - for example, different types of functions can be used to model the uncertainty in the parameters. Here we have used only Triangular Fuzzy Numbers (TFNs) to capture the uncertainty in the task durations. An advantage of this representation is the ease of interpretation of the results, for instance, in terms of the most likely, optimistic and pessimistic estimates of the makespan, although an interpretation of the objective function value is not as straight forward as in the probabilistic case (for e.g., in terms of the expected value).

2. The most significant gain is the computation time required to solve the optimization models under uncertainty. These models neither suffer from the combinatorial explosion of scenarios that discrete probabilistic uncertainty representations exhibit, nor do they require complicated integration schemes which continuous probabilistic models require. The MILP models developed in this work could be solved with reasonable computational effort for problems where the solution was more structured. These models did not perform as well when we attempted to solve general problems to optimality.

3. Another significant advantage of the fuzzy-set approach (over the probabilistic approach) is that heuristic search algorithms for combinatorial optimization such as tabu search can be easily applied to obtain good quality solutions in reasonable computing time. This is primarily due to the ease in computing the objective function of a solution.

4. The examples considered show that very good estimates of the uncertain makespan and income can be obtained by using fairly coarse discretizations and that these models can be solved with little computational effort. In these examples the improvement in the estimation of the completion time by using a denser discretization was not significant enough to warrant the order of magnitude increase in computation time required.

Appendix A - Reduced Neighborhood

The idea of using a reduced neighborhood to improve the computational characteristics of local search algorithms is not new. For instance, van Laarhoven *et al.* (1992) used the concept of *critical arcs* in jobshop scheduling problems for eliminating moves that will definitely not improve the solution. This idea was then extended to the fuzzy jobshop scheduling problem by Fortemps (1997). A recent paper by Nowicki and Smutnicki (1998) also employs a specific neighborhood definition based on critical paths for flow shops with parallel machines in a deterministic setting. With this reduced neighborhood and further advanced implementations of tabu list querying and searching, the authors were able to dramatically improve the speed of their tabu search algorithm.

In our implementation, we build upon the ideas in both Fortemps (1997) and Nowicki and Smutnicki (1998) to define our reduced neighborhood. Thus, for a given solution, we identify the critical path(s) and consider only the insertion of jobs that lie on the critical path(s). A move that seeks to insert a job that does not lie on any of the critical paths is considered useless, in that it will not improve the makespan. The main difference between the deterministic version and our case is that there may be many more critical jobs, since different paths may be critical at different α-levels. The identification of the critical path(s) for a solution is a straightforward operation and does not contribute significantly to the computational complexity of an iteration.

However, we have to account for the memory required for storing the additional information about the critical jobs with every solution. If we store information about all critical paths at all α-levels, the memory required becomes excessive very quickly. Therefore, in our implementation we have stored the predecessor information to be used for computing the critical path from only the $\alpha = 1$ level, which corresponds to the most possible realization of processing times. Checking if a move is useless is done in constant time using a hashtable containing the critical jobs associated with the solution. Fortunately, with these algorithmic improvements, the size of the reduced neighborhood is only $O(N \cdot W)$, where W is the number of jobs on the critical path. Now, W is often much smaller than $N \cdot M$, thus when compared with the $O(N^2 \cdot M)$ size of the complete insertion neighborhood we have a considerable improvement. Also, the solutions obtained by performing the useless moves do not need to be evaluated in the reduced neighborhood implementation unlike the complete implementation, thereby leading to a further reduction in the computational effort.

References

Aarts, E.H.L. and Korst, J. (1989) Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, John Wiley, New York.

Balasubramanian, J. and Grossmann, I.E. (2002) A novel branch and bound algorithm for scheduling flowshop plants with uncertain processing times. *Computers and Chemical Engineering*, **26**, 41.

Arnold, K. and Gosling, J. (1998) The Java Programming Language, Second Edition, Addison-Wesley, Reading, MA.

Battiti, R. and Tecchiolli, G. (1994) The reactive tabu search. *ORSA Journal on Computing*, **6**, 126.

Birewar, D. B. and Grossmann, I. E. (1989) Efficient optimization algorithms for zero-wait scheduling of multiproduct batch plants. *Industrial and Engineering Chemistry Research*, **28**, 1333.

Campbell, H.G., Dudek, R.A. and Smith, M.L. (1970) A heuristic algorithm for the n-job m-machine sequencing problem. *Management Science*, **16**, 630.

Chanas, S. and Kamburowski, J. (1981) The use of fuzzy variables in PERT. *Fuzzy Sets and Systems*, **5**, 11.

Chen, S.-J. and Hwang, C.-L. (1992) Fuzzy Multiple Attribute Decision Making, Springer, Berlin, 1992.

Dubois, D. and Prade, H. (1987) The mean value of a fuzzy number. *Fuzzy Sets and Systems*, **24**, 279.

Dubois, D. and Prade, H. (1988) Possibility Theory: An Approach to Computerized Processing of Uncertainty, Plenum Press, New York, 1988.

Fortemps, P. (1997) Jobshop scheduling with imprecise durations: a fuzzy approach. *IEEE Transactions on Fuzzy Systems*, **v 5 n 4**, 557.

Fortemps, P. and Roubens, M. (1996) Ranking and defuzzification methods based on area compensation. *Fuzzy Sets and Systems*, **82**, 319.

Galluzzo, M.; Ducato, R.; Bartolozzi, V. and Picciotto, A. (2001) Expert control of DO in the aerobic reactor of an activated sludge process. *Computers and Chemical Engineering*, **25**, 619.

Glover, F. (1990) Tabu search: a tutorial. *Interfaces*, **20**, 74.

Hagstrom, J. N. (1988) Computational complexity of PERT problems. *Networks*, **18**, 139.

Hapke, M. and Slowinski, R. (1996) Fuzzy priority heuristics for project scheduling. *Fuzzy Sets and Systems*, **83**, 291.

Honkomp, S.J.; Mockus, L.; Reklaitis, G. V. (1999) A framework for schedule evaluation with processing uncertainty *Computers and Chemical Engineering*, **23**, 595.

Hui, C.-W., Gupta, A. and van der Meulen, H.A.J. (2000) A novel MILP formulation for short-term scheduling of multi-stage multi-product batch plants with sequence-dependent constraints. *Computers and Chemical Engineering*, **24**, 2705.

Ierapetritou, M. G. and Pistikopoulos, E. N. (1996) Batch plant design and operations under uncertainty. *Industrial and Engineering Chemistry Research*, **35**, 772.

Ishibuchi, H., Yamamoto, N., Murata, T. and Tanaka, H. (1994) Genetic algorithms and neighborhood search algorithms for fuzzy flowshop scheduling problems. *Fuzzy Sets and Systems*, **67**, 81.

Kubic, W. L. (Jr.) and Stein, F. P. (1988) A theory of design reliability using probability and fuzzy sets. *AIChE Journal*, **34**, 583.

Lootsma, F.A. (1989) Stochastic and fuzzy PERT. *European Journal of Operational Research*, **43**, 174.

Malcolm, D.G., Roseboom, J.H., Clark, C.E. and Fazar, W. (1959) Applicatin of a technique for research and development program evaluation. *Operations Research*, **12**, 16.

Majozi, T. and Zhu, X. X. (2001) A combined fuzzy set theory and MILP approach in integration of planning and scheduling of batch plants. Presented at AIChE Annual Conference, Reno, 2001.

McCahon, C.S. and Lee, E.S. (1992) Fuzzy job sequencing for a flow shop. *European Journal of Operational Research*, **62**, 294.

McDonald, C.M. and Karimi, I. (1997) Planning and scheduling of parallel semi-continuous process. 2: short-term scheduling. *Industrial and Engineering Chemistry Research*, **36**, 2701.

Nowicki, E. and Smutnicki, C. (1998) The flowshop with parallel machines: a tabu search approach. *European Journal of Operational Research*, **106**, 226.

Ow, P.S. and Morton, T.E. (1988) Filtered beam search in scheduling. *International Journal of Production Research*, **26**, 35.

Özelkan, E.C. and Duckstein, L. (1999) Optimal fuzzy counterparts of scheduling rules. *European Journal of Operational Research*, **113**, 593.

Pekny, J.F.; Miller, D.L. (1991) Exact solution of the no-wait flowshop scheduling problem with a comparison to heuristic methods *Computers and Chemical Engineering*, **15**, 741..

Petkov, S. B. and Maranas, C. D. (1997) Multiperiod planning and scheduling of multiproduct batch plants under demand uncertainty. *Industrial and Engineering Chemistry Research*, **36**, 4864.

Pinto, J. and Grossmann, I.E. (1995) A continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants. *Industrial and Engineering Chemistry Research*, **34**, 3037.

Pinto, J. and Grossmann, I.E. (1996) An alternate MILP model for short-term scheduling of batch plants with pre-ordering constraints. *Industrial and Engineering Chemistry Research*, **35**, 338.

Postlethwaite, B. and Edgar, C. (2000) MIMO fuzzy model-based controller *Chemical Engineering Research and Design, Part A: Transactions of the Institute of Chemical Engineers*, **78**, 557.

Rajagopalan, D. and Karimi, I. A. (1989) Completion times in serial mixed-storage multiproduct processes with transfer and set-up times. *Computers and Chemical Engineering*, **13**, 175.

Sand, G, Engell, A., Märkert, A, Schultz, R. and Schulz C. (2000) Approximation of an ideal online scheduler for a multiproduct batch plant. *Computers and Chemical Engineering*, **24**, 361.

Schmidt, C. (1998) Optimal Scheduling of the Agricultural Chemical New Product Development Process. Ph. D. Thesis, Department of Chemical Engineering, Carnegie Mellon University, Pittsburgh, PA.

Schmidt, C. and Grossmann, I.E. (1996) Optimization models for the scheduling of testing tasks in new product development. *Industrial and Engineering Chemistry Research*, **35**, 3498.

Schmidt, C. and Grossmann, I.E. (2000) The exact overall time distribution of a project with uncertain task durations. *European Journal of Operational Research*, **v 126 n 3**, 614.

Shah, N. (1998) Single- and multisite planning and scheduling: current status and future challenges. In Proceedings of the Conference on Foundations of Computer Aided Process Operations FOCAPO98, Snowbird, USA, 75-90.

Slowinski, R. and Hapke, M. (Eds.) (2000) Scheduling under Fuzziness, Physica Verlag, Heidelberg.

Tarifa, E. and Chiotti, O. (1995) Flexibility vs costs in multiproduct batch plant design: a calculation algorithm. *Chemical Engineering Research and Design, Part A: Transactions of the Institute of Chemical Engineers*, **73**, 931.

van Laarhoven, P.J.M., Aarts, E.H.L. and Lenstra, J.K. (1992) Jobshop scheduling by simulated annealing. *Operations Research*, **40**, 113.

Wang, J.R. (1999) A fuzzy set approach to activity sheduling for product development. *Journal of the Operational Research Society*, **50**, 1217.

Wets, R. J.-B. (1974) Stochastic programming with fixed recourse : the equivalent deterministic program *SIAM Review*, **16**, 309.

Zadeh, L. (1965) Fuzzy sets. *Information and Control*, **8**, 338.