Chapter 3
Stochastic Scheduling for a Network of Flexible Job Shops

Subhash C. Sarin, Hanif D. Sherali, Amrusha Varadarajan, and Lingrui Liao

Abstract In this chapter, we address the problem of optimally routing and sequencing a set of jobs over a network of flexible machines for the objective of minimizing the sum of completion times and the cost incurred, assuming stochastic job processing times. This problem is of particular interest for the production control in high investment, low volume manufacturing environments, such as pilot-fabrication of microelectromechanical systems (MEMS) devices. We model this problem as a two-stage stochastic program with recourse, where the first-stage decision variables are binary and the second-stage variables are continuous. This basic formulation lacks relatively complete recourse due to infeasibilities that are caused by the presence of re-entrant flows in the processing routes, and also because of potential deadlocks that result from the first-stage routing and sequencing decisions. We use the expected processing times of operations to enhance the formulation of the first-stage problem, resulting in good linear programming bounds and inducing feasibility for the second-stage problem. In addition, we develop valid inequalities for the first-stage problem to further tighten its formulation. Experimental results are presented to demonstrate the effectiveness of using these strategies within a decomposition algorithm (the L-shaped method) to solve the underlying stochastic program. In addition, we present heuristic methods to handle large-sized instances of this problem and provide related computational results.

Keywords Stochastic scheduling • Flexible job shop • Multi-site scheduling • L-shaped method • Branch-and-bound

S.C. Sarin (✉) • H.D. Sherali • A. Varadarajan • L. Liao
Grado Department of Industrial and Systems Engineering, Virginia Tech,
Blacksburg, VA 24061, USA
e-mail: sarins@vt.edu

G. Rabadi (ed.), *Heuristics, Metaheuristics and Approximate Methods in Planning and Scheduling*, International Series in Operations Research & Management Science 236, DOI 10.1007/978-3-319-26024-2_3

3.1 Introduction: Problem Statement and Related Literature

The production in high investment, low volume manufacturing environments, such as pilot-fabrication of microelectromechanical systems (MEMS) devices, gives rise to several special features of the underlying scheduling problem. Due to high prices of the processing equipments and complicated fabrication processes, it is impractical to assign dedicated equipment to each processing step. The relatively low volume of production during the pilot stage also implies that machine flexibility is highly desirable so that multiple product types can share processing equipments. Hence, each manufacturing facility is organized as a flexible job shop, serving multiple processing routes with flexible machines. Furthermore, due to novelty of the products and fabrication processes, a single facility often lacks the capability of performing all the required processing steps for a product from start to finish. To satisfy these special requirements, multiple manufacturing facilities are usually organized into a distributed fabrication network, where a central service provider coordinates production activities across facilities and directly deals with customers' requirements. Products are shipped from one facility to another until all processing requirements are met. For a given processing step, there may be multiple facilities that can provide the required service. The flexibility of cross-facility routing not only provides more pricing and quality options for the customers, but also makes transportation time and cost an important aspect of the scheduling problem. We designate this type of distributed fabrication network as the *Network of Flexible Job Shops* (**NFJS**).

The management of operations for an NFJS involves two types of decisions: (1) choosing a facility for each job operation (i.e., processing step) and assigning it to a compatible machine within the facility (i.e., *routing*) and (2) stipulating a processing sequence for the operations assigned to any given machine (i.e., *sequencing*). The routing decisions need to take transportation time and cost into consideration, as they can be quite significant between geographically dispersed facilities. On the other hand, sequencing decisions need to account for the fact that sequence-dependent set-up times are required to prepare for the processing of operations of different jobs on the same machine. In view of the pricing and quality options that are available in an NFJS, the customer specifies for each job a fixed budget, which can only be exceeded under a given penalty rate. The job arrival times, number of operations for each job, machines capable of processing each operation, transportation times and transportation costs between facilities, sequence-dependent set-up times, and customer budgets are assumed to be known (and thus, deterministic). On the other hand, since exact values of processing times are expected to vary due to the novelty of fabrication technologies, they are assumed to be stochastic. The problem that we address in this chapter can be succinctly stated as follows:

> Given a set of jobs and a network of flexible job shops, where operation processing times are uncertain but the sequence in which to process the operations of each job is known a priori, determine an allocation of job operations to facilities and a sequence in which to

process these operations on the machines in the facility so as to minimize a function of the completion times and the transportation and processing costs incurred.

The NFJS problem combines the characteristics of three well-known problems: the multi-site planning and scheduling problem, the flexible job shop scheduling problem, and the stochastic job shop scheduling problem. Multi-site *planning* problems are extensions of capacitated lot-sizing problems, with emphasis on transportation requirements and site-specific holding cost. Production is assigned to machines at multiple sites to satisfy demands during each period of the time horizon. The multi-site *scheduling* problem further addresses underlying production issues, such as inventory interdependency and change-over setup. To deal with the integrated multi-site planning and scheduling problem, iterative methods have been applied that alternate between solving the long-term planning problem and solving the short-term scheduling problem (see, for example, Roux et al. 1999; Guinet 2001; Gnoni et al. 2003). Others have considered the monolithic approach, either using the approach of variable time scale (Timpe and Kallrath 2000; Lin and Chen 2006), or relying on heuristic methods (Gascon et al. 1998; Sauer et al. 2000; Jia et al. 2003) to handle the resulting complexity. Lee and Chen (2001) provided a comprehensive study on scheduling with transportation considerations for the single facility environment. They considered two particular cases pertaining to transportation within a flow shop environment and transportation during final product distribution. To the best of our knowledge, no previous research in the multi-site planning and scheduling area has considered routing flexibility and stochastic processing times, both of which are very pertinent to the NFJS problem.

In a flexible job shop environment, for each processing step of a job, there are multiple alternative machines that are capable of providing the required service. Various methods have been applied to solve problems of this type. For example, Iwata et al. (1980) and Kim (1990) have considered dispatching rules; Nasr and Elsayed (1990) have applied greedy heuristic methods, Hutchison et al. (1991) have devised a hierarchical decomposition method that determines the assignment of operations to machines and then generates sequences. With regard to iterative local search methods, Brandimarte (1993) considered re-assignment and re-sequencing as two different types of moves, while Dauzère-Pérès and Paulli (1997) and Mastrolilli and Gambardella (2000) did not explicitly treat them as different. Subramaniam et al. (2000) have performed a simulation study with dynamic job arrival and machine break downs. One can also find applications of meta-heuristic methods to solve this problem including, but not limited to, particle swam optimization (Xia and Wu 2005) and genetic algorithms (Pezzella et al. 2008 and Wang et al 2005). The routing flexibility that characterizes the flexible job shop problem is also present in the NFJS problem, but with an important distinction that the alternative machines may be located at different facilities (sites), thereby requiring consideration of transportation time and cost into the scheduling problem.

The stochastic scheduling problem has been addressed in the literature in the classical flow shop and job shop environments. Optimal policies, dominance relations, and dispatching rules for two- and three-machine flow shop scheduling problems having stochastic processing times have been developed by Ku and Niu (1986), Weiss (1982), Mittal and Bagga (1977), Cunningham and Dutta (1973), Bagga (1970), Talwar (1967), Makino (1965), Prasad (1981), Forst (1983), Pinedo (1982), Jia (1998), Elmaghraby and Thoney (1999), and Kamburowski (1999, 2000). These studies vary either in the distribution of the processing times used, or in the objective function, or in the amount of intermediate storage available between machines. Optimal rules have also been developed by Foley and Suresh (1984) and Pinedo (1982) to minimize the expected makespan for the m-machine flow shop problem with stochasticity in processing times. For work in stochastic job shops, see Golenko-Ginzburg et al. (1995, 1997, 2002), Singer (2000), Luh et al. (1999), Kutanoglu and Sabuncuoglu (2001), Yoshitomi (2002), Lai et al. (2004), and Tavakkoli-Moghaddam et al. (2005).

The remainder of this chapter is organized as follows. In Sect. 3.2, we model the NFJS problem as a two-stage stochastic program and present the L-shaped method for its solution. Besides developing the pertinent feasibility and optimality cuts, we also introduce an alternative approach to induce second-stage feasibility. In Sect. 3.3, the formulation of the first-stage problem is further tightened by using three types of valid inequalities, all of which rely upon the special structure of the NFJS problem. Computational results are provided in Sect. 3.4 to demonstrate the efficacy of our model formulation and solution approach. For even large-sized problem instances, we present heuristic methods and the results on their performances in Sect. 3.5. Concluding remarks are made in Sect. 3.6.

3.2 Stochastic Model for a Network of Flexible Job Shops

We model the stochastic NFJS problem as a two-stage stochastic program with recourse, where the first-stage variables are binary and pertain to the assignment of job operations to machines and to the sequencing of job operations for processing on these machines, while the second-stage variables are continuous and relate to the completion times and budget over-runs of the jobs, and where the uncertainty in processing time durations influences the job completion times. Multiple facilities are incorporated in our formulation by assigning to each machine a unique identification number that distinguishes it from the other machines in all the facilities, and by appropriately considering the inter-machine transportation times and costs. Stochastic processing times are modeled by a finite set of scenarios for the entire problem, and each of these scenarios assigns durations to every possible processing step and has an associated probability value.

We present the overall problem formulation and decompose it into two stages using Benders' decomposition (Benders 1962). Besides constructing the feasibility cuts and optimality cuts, we further reinforce the first-stage problem by including additional valid inequalities that induce feasibility in the second stage.

3.2.1 Model Formulation for the NFJS Problem

Notation

Indices:

Job index: $i = 1, \ldots, N$
Operation index for job i: $j = 1, \ldots, J_i$
Machine index: $m = 1, \ldots, |M|$ (where M is the set of machines)
Scenario index: $s = 1, \ldots, S$

Decision Variables

$$x^m_{(i,j)} = \begin{cases} 1, & \text{if operation } j \text{ of job } i \text{ is assigned to machine } m, \\ 0, & \text{otherwise.} \end{cases}$$

$$y^m_{(i,j,k,l)} = \begin{cases} 1, & \text{if operation } j \text{ of job } i \text{ directly precedes operation } l \text{ of} \\ & \text{job } k \text{ on machine } m, \\ 0, & \text{otherwise.} \end{cases}$$

$$v^{(e,f)}_{(i,j,j+1)} = \begin{cases} 1, & \text{if operation } j \text{ of job } i \text{ is performed on machine } e \text{ and} \\ & \text{operation } j+1 \text{ of job } i \text{ is performed on machine } f, \\ 0, & \text{otherwise.} \end{cases}$$

$t^s_{(i,j)}$ = completion time of operation j of job i under scenario s.
Δ^s_i = budget over-run for job i under scenario s.

Parameters

$H^{(m,s)}_{(i,j,k,l)}$ = an appropriately large positive number; its value is specified in (3.16) below.
w_i = number of parts in job i.
M = set of all the machines.
ϕ_s = probability of occurrence for scenario s.
$c^m_{(i,j)}$ = cost per unit processing time of operation j of job i on machine m.
$p^{(m,s)}_{(i,j)}$ = processing time of operation j of job i on machine m under scenario s.
Z_m = set of job operations that can be processed on machine m.
$M_{(i,j)}$ = set of machines capable of processing operation j of job i.
$u^m_{(i,j,k,l)}$ = changeover time to switch from operation j of job i to operation l of job k on machine m.
b_i = budget for job i.
r_i = ready time for the first operation of job i.
$d_{(e,f)}$ = transportation time between machines e and f.
$q_{(e,f)}$ = per part transportation cost between machines e and f.
α_i = cost coefficient for job i that is ascribed to its completion time.
β_i = penalty coefficient for job i corresponding to its budget over-run.

Formulation NFJSP

$$\text{Minimize } z = \sum_{i=1}^{N} \alpha_i \left(\sum_{s=1}^{S} \phi_s t^s_{(i,J_i)} \right) + \sum_{i=1}^{N} \beta_i \left(\sum_{s=1}^{S} \phi_s \Delta^s_i \right)$$
$$+ \sum_{s=1}^{S} \phi_s \sum_{i=1}^{N} \sum_{j=1}^{J_i} \sum_{m \in M_{(i,j)}} p^{(m,s)}_{(i,j)} x^m_{(i,j)}$$
$$+ \sum_{m \in M} \sum_{(i,j) \in Z_m} \sum_{\substack{(k,l) \in Z_m \\ (k,l) \neq (i,j)}} u^m_{(i,j,k,l)} y^m_{(i,j,k,l)}$$
$$+ \sum_{i=1}^{N} \sum_{j=1}^{J_i - 1} \sum_{e \in M_{(i,j)}} \sum_{f \in M_{(i,j+1)}} d_{(e,f)} v^{(e,f)}_{(i,j,j+1)} \tag{3.1}$$

subject to:

$$t^s_{(i,j)} + \sum_{m \in M_{(i,j+1)}} \left(p^{(m,s)}_{(i,j+1)} x^m_{(i,j+1)} \right) + \sum_{e \in M_{(i,j)}} \sum_{f \in M_{(i,j+1)}} d_{(e,f)} v^{(e,f)}_{(i,j,j+1)}$$
$$\leq t^s_{(i,j+1)}, \quad \forall i = 1, \ldots, N, \;\; j = 1, \ldots, J_i - 1, \;\; s = 1, \ldots, S \tag{3.2}$$

$$r_i + \sum_{m \in M_{(i,1)}} \left(p^{(m,s)}_{(i,1)} x^m_{(i,1)} \right) \leq t^s_{(i,1)}, \quad \forall i = 1, \ldots, N, \;\; s = 1, \ldots, S \tag{3.3}$$

$$t^s_{(i,j)} + p^{(m,s)}_{(k,l)} + u^m_{(i,j,k,l)} \leq t^s_{(k,l)} + \left(1 - y^m_{(i,j,k,l)} \right) H^{(m,s)}_{(i,j,k,l)},$$
$$\forall m \in M, \forall\, (i,j) \neq (k,l) \in Z_m, \;\; \forall s = 1, \ldots, S \tag{3.4}$$

$$\sum_{m \in M_{(i,j)}} x^m_{(i,j)} = 1, \quad \forall i = 1, \ldots, N, \;\; j = 1, \ldots, J_i \forall i = 1, \ldots, N, \;\; j = 1, \ldots, J_i \tag{3.5}$$

$$\sum_{\substack{(i,j) \in Z_m \\ (i,j) \neq (k,l)}} y^m_{(i,j,k,l)} \leq x^m_{(k,l)}, \quad \forall k = 1, \ldots, N, \;\; l = 1, \ldots, J_k, \;\; m \in M_{(k,l)} \tag{3.6}$$

$$\sum_{\substack{(i,j) \in Z_m \\ (i,j) \neq (k,l)}} y^m_{(k,l,i,j)} \leq x^m_{(k,l)}, \quad \forall k = 1, \ldots, N, \;\; l = 1, \ldots, J_k, \;\; m \in M_{(k,l)} \tag{3.7}$$

$$\sum_{(i,j)\in Z_m} \sum_{\substack{(k,l)\in Z_m \\ (k,l)\neq(i,j)}} y^m_{(i,j,k,l)} \geq \sum_{(i,j)\in Z_m} x^m_{(i,j)} - 1, \quad \forall \ m \in M \tag{3.8}$$

$$\sum_{j=1}^{J_i} \sum_{m\in M_{(i,j)}} c^m_{(i,j)} p^{(m,s)}_{(i,j)} x^m_{(i,j)} + \left(\sum_{j=1}^{J_i-1} \sum_{e\in M_{(i,j)}} \sum_{f\in M_{(i,j+1)}} q_{(e,f)} v^{(e,f)}_{(i,j,j+1)} \right) w_i - \Delta^s_i \leq b_i,$$
$$\forall i = 1,\ldots,N, \ \ s = 1,\ldots,S \tag{3.9}$$

$$v^{(e,f)}_{(i,j,j+1)} \leq x^e_{(i,j)}, v^{(e,f)}_{(i,j,j+1)} \geq x^e_{(i,j)} + x^f_{(i,j+1)} - 1,$$
$$\forall i = 1,\ldots,N, j = 1,\ldots,J_i - 1, e \in M_{(i,j)}, f \in M_{(i,j+1)} \tag{3.10}$$

$$\Delta^s_i \geq 0, \quad \forall i = 1,\ldots,N, s = 1,\ldots,S \tag{3.11}$$

$$x^m_{(i,j)} \in \{0,1\}, \quad \forall i = 1,\ldots,N, j = 1,\ldots,J_i, m \in M_{(i,j)} \tag{3.12}$$

$$y^m_{(i,j,k,l)} \in \{0,1\}, \quad \forall m = 1,\ldots,M, \forall\ (i,j) \neq (k,l) \in Z_m \tag{3.13}$$

$$v^{(e,f)}_{(i,j,j+1)} \in [0,1], \quad \forall i = 1,\ldots,N, j = 1,\ldots J_i - 1, e \in M_{(i,j)}, f \in M_{(i,j+1)}. \tag{3.14}$$

The objective function (3.1) is composed of five terms. The first and the second terms penalize the sum of job completion times and budget over-runs, respectively. The penalty coefficients reflect the customer's emphasis on the lead-time and costs incurred, and they also scale the first two terms to be commensurate with the next three terms, which are time based. The third term represents expected processing time for the operations of all the jobs; the fourth term computes the total set-up time on the machines, and the final term determines the sum of travel times incurred by all the jobs. Note that the last three terms in the objective function support the first term by aiding the achievement of lower completion times, while at the same time, reflect costs incurred by consuming machine and transportation capacities of the system. Constraints (3.2) capture precedence relationships between operations of the same job. Specifically, they state that under each scenario s, the completion time of operation $j+1$ of job i, $\forall i = 1,\ldots,N$, must be at least equal to the completion time of operation j of that job plus the processing time of operation $j+1$ and any travel time incurred between the two operations (set-up time is assumed to be job-detached, and hence, is not included here). Constraints (3.3) ensure (for each scenario) that each job does not commence its first operation earlier than its ready time. Constraints (3.4) establish relationships among the operations to be performed on the same machine. Given two distinct job-operations, say (i, j) and (k, l) in Z_m for a certain machine m, if (i, j) were to directly precede (k, l), (i.e., $y^m_{(i,j,k,l)} = 1$), then the completion time of (k, l) under any scenario s must be at least equal to the completion time of (i, j) for that scenario, plus the processing time of (k, l) and the sequence-dependent set-up time between the two operations. Observe that when $y^m_{(i,j,k,l)} = 0$, i.e., (i, j) does not directly precede (k, l), the constraint becomes redundant by the choice of a suitably large value of $H^{(m,s)}_{(i,j,k,l)}$ (see (3.16) below). Constraints (3.5) ensure that each job-operation is assigned to exactly one machine out of the several alternative machines that can process it. Constraints (3.6) and (3.7) state that if a job-operation, say (k, l), is assigned to a machine m, it can be preceded (respectively succeeded) by at most one job-operation from the set of operations that the machine is capable of processing. Note that if (k, l) is the first operation to be processed on this machine, it will not be preceded by any other operation; and likewise if (k, l) is the last operation to be processed, it will not be succeeded by any other operation. In both of these cases, the left-hand sides of (3.6) and (3.7) will be zero, which trivially yield valid relationships. Also, if (k, l) is not assigned to machine m, then all the direct precedence $\boldsymbol{y}$-variables that relate (k, l) to other operations on machine m are validly set equal to zero by (3.6) and (3.7). Constraints (3.8) guarantee that if a machine has some $\sum_{(i,j)\in Z_m} x^m_{(i,j)}$ operations assigned to it for processing, then there must exist one less than this number of direct precedence variables that are set equal to 1 for this machine. These constraints are written as inequalities rather than as equalities to account for the case where the number of operations assigned to a machine is actually zero. Also, together with (3.6) and (3.7), these constraints establish the definitional role of the $\boldsymbol{y}$-variables. Constraints (3.9) enforce budgetary restrictions on each job i under every processing time scenario s. These constraints permit the sum of processing costs and travel costs for all operations of a job to exceed the budget by an amount of Δ^s_i, but with a corresponding penalty in the objective function. Note that the travel cost for each job i is assumed to be proportional to the number of parts, w_i, in that job. Constraints (3.10) enforce the relationship between the $\boldsymbol{x}$- and $\boldsymbol{v}$-variables according to $v^{(e,f)}_{(i,j,j+1)} = x^e_{(i,j)} x^f_{(i,j+1)}$ using a standard linearization technique whereby $v^{(e,f)}_{(i,j,j+1)} = 1$ if and only if both $x^e_{(i,j)} = 1$ and $x^f_{(i,j+1)} = 1$. Note that the $\boldsymbol{v}$-variables account for the required transfer between the machines in the objective function (3.1) and in Constraints (3.2) and (3.9). As such, because of the positive coefficients associated with these variables in the objective function (3.1) and the less-than-or-equal-to ($\leq$) relationships in (3.2) and (3.9), we could omit the first two sets of $\leq$ restrictions in (3.10) and have them automatically hold true at optimality. Constraints (3.11), (3.12), (3.13), and (3.14) ascribe nonnegativity and binary restrictions on the decision variables, while the $\boldsymbol{v}$-variables will automatically turn out to be binary-valued even though declared to be continuous on [0, 1]. Note also that the nonnegativity on the $\boldsymbol{t}$-variables is implied by (3.2), (3.3), (3.12), and (3.13).

The value of $H^{(m,s)}_{(i,j,k,l)}$ used in (3.4) can be prescribed as follows. Note that, if $y^m_{(i,j,k,l)} = 0$, then this constraint reduces to

$$t^s_{(i,j)} + p^{(m,s)}_{(k,l)} + u^m_{(i,j,k,l)} - t^s_{(k,l)} \le H^{(m,s)}_{(i,j,k,l)}. \tag{3.15}$$

Hence, it is sufficient to assign to $H^{(m,s)}_{(i,j,k,l)}$ a valid upper bound on the left-hand side expression in (3.15). Given conservative bounds for $t^s_{(i,j)}$ such that $\left(t^s_{(i,j)}\right)_{\min} \le t^s_{(i,j)} \le \left(t^s_{(i,j)}\right)_{\max}$, we can set

$$H^{(m,s)}_{(i,j,k,l)} = \left(t^s_{(i,j)}\right)_{\max} + p^{(m,s)}_{(k,l)} + u^m_{(i,j,k,l)} - \left(t^s_{(k,l)}\right)_{\min}. \tag{3.16}$$

With respect to the bounds for $t^s_{(i,j)}$, we take

$$\left(t^s_{(i,j)}\right)_{\min} = r_i + \sum_{j' \le j} \min_{m \in M(i,j')} \left\{p^{(m,s)}_{(i,j')}\right\} + \sum_{2 \le j' \le j} \min_{\substack{e \in M(i,j'-1)\\ f \in M(i,j')}} \left\{d_{(e,f)}\right\},$$

$$\left(t^s_{(i,j)}\right)_{\max} = \tau^s - \sum_{j' > j} \min_{m \in M(i,j')} \left\{p^{(m,s)}_{(i,j')}\right\} - \sum_{j < j' \le J_i} \min_{\substack{e \in M(i,j'-1)\\ f \in M(i,j')}} \left\{d_{(e,f)}\right\},$$

$$\forall i = 1, \ldots, N, j = 1, \ldots, J_i, s = 1, \ldots, S, \tag{3.17}$$

where τ^s is some conservative upper bound on the overall makespan of all the jobs under scenario s. We used the value of τ^s to be the sum of the processing times of all the operations of the jobs.

3.2.2 The L-Shaped Method for the NFJS Problem

Formulation NFJSP can be decomposed into the following Stage-I (master) and Stage-II (recourse) problems:

Stage-I: Master Problem
MP: Minimize

$$\sum_{s=1}^{S} \phi_s \sum_{i=1}^{N} \sum_{j=1}^{J_i} \sum_{m \in M_{(i,j)}} p^{(m,s)}_{(i,j)} x^m_{(i,j)} + \sum_{m \in M} \sum_{(i,j) \in Z_m} \sum_{\substack{(k,l) \in Z_m\\ (i,j) \ne (k,l)}} y^m_{(i,j,k,l)} u^m_{(i,j,k,l)}$$
$$+ \sum_{i=1}^{N} \sum_{j=1}^{J_i - 1} \sum_{e \in M_{(i,j)}} \sum_{f \in M_{(i,j+1)}} v^{(e,f)}_{(i,j,j+1)} d_{(e,f)} + \sum_{s=1}^{S} \phi_s \mathbf{Q}(\mathbf{x}, \mathbf{y}, \mathbf{v}, s) \tag{3.18}$$

subject to: (3.5), (3.6), (3.7), (3.8), (3.10), (3.12), (3.13), and (3.14), where $\mathbf{Q}(\mathbf{x}, \mathbf{y}, \mathbf{v}, s)$ is the recourse function corresponding to the optimal value of the subproblem that minimizes the penalized sum of job completion times and budget over-runs for a given assignment vector **x**, sequencing vector **y**, tracking vector **v**, and for a processing time scenario s. The linear recourse subproblem for scenario s is given by:

Stage-II: Recourse Problem

$$\mathbf{RP}: \quad \mathbf{Q}\left(\mathbf{x}, \mathbf{y}, \mathbf{v}, s\right) = \text{Min} \sum_{i=1}^{N} \alpha_i t^s_{(i,J_i)} + \sum_{i=1}^{N} \beta_i \Delta^s_i \tag{3.19}$$

subject to: (3.2), (3.3), (3.4), (3.9), and (3.11).

We note that, in the decomposition outlined above for formulation NFJSP, the master problem could generate an assignment and sequencing solution that might not be feasible to the subproblem. There are three possible causes for such infeasibility. First, the Stage-I formulation does not exclude "subtours" while sequencing operations assigned to a particular machine. For example, suppose that operations (a, b), (c, d), and (e, f) are assigned to machine m, so that

$$x^m_{(a,b)} = x^m_{(c,d)} = x^m_{(e,f)} = 1.$$

One can verify that the following values of direct precedence variables are feasible to the Stage-I formulation (in particular, satisfies Constraint (3.8)):

$$\begin{aligned} y^m_{(a,b,c,d)} &= 1;\ y^m_{(a,b,e,f)} = 0;\\ y^m_{(c,d,a,b)} &= 1;\ y^m_{(c,d,e,f)} = 0;\\ y^m_{(e,f,a,b)} &= 0;\ y^m_{(e,f,c,d)} = 0. \end{aligned}$$

However, due to the subtour between (a, b) and (c, d), this solution does not represent a valid processing sequence. In the NFJSP formulation, this kind of subtour is eliminated by Constraints (3.4), which are not included in the master problem. Second, note that Constraints (3.2) in NFJSP, upon decomposition, become part of the subproblem and capture the fact that the completion time of a lower indexed operation of a job must be less than or equal to that for any higher indexed operations of the same job. In NFJSP, Constraints (3.2) in conjunction with other constraints that determine the value of the $\boldsymbol{y}$-variables (Constraints (3.6), (3.7), and (3.8)) ensure that, in the case of re-entrant flow, where a job visits a machine for multiple operations, the lower indexed operations of a job are sequenced before a higher indexed operation of the same job. However, since Constraints (3.2) are

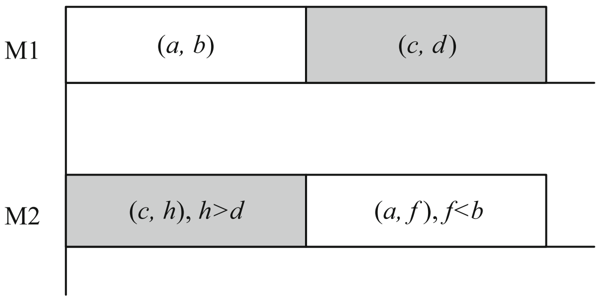


Fig. 3.1 A deadlock configuration involving two machines

no longer a part of the master problem, its absence may result in an assignment and sequencing vector that does not honor the re-entrant flow conditions. Third, the assignment and sequencing vectors from the master problem may cause a *deadlock*. This occurs in the face of a certain configuration of assignment and sequencing decisions that result in a circuit or a cycle wherein each operation in the cycle waits for another operation within the cycle to complete processing. This is illustrated in Fig. 3.1. Note that on machine M1, (c, d) waits for (a, b) to finish processing according to the sequencing decision. Operation (a, b) on machine M1 must follow (a, f) on machine M2 owing to operating precedence constraints. However, on machine M2, operation (a, f) follows (c, h), which, in turn, can begin only after (c, d) on machine M1 has been completed. Thus, none of the four operations can begin processing, resulting in a deadlock.

As a result of this potential infeasibility, the above decomposition of NFJS problem does not possess the property of relatively complete recourse. In order to render the first-stage solution feasible to the second-stage, it is necessary to obviate the infeasibility due to subtours, re-entrant flows, and deadlocks. One way to achieve this is through the use of artificial variables as described by van Slyke and Wets (1969). These variables are inserted into the subproblems for every scenario and feasibility cuts are developed that become a part of the master problem, which in turn ultimately induce the master problem to generate solutions that are feasible to the subproblems. Accordingly, for a given output $(\mathbf{x}, \mathbf{y}, \mathbf{v})$ from the master problem, the following augmented recourse problem (ARP) is solved, one for each scenario s:

ARP: Minimize

$$\sum_{i=1}^{N}\sum_{j=1}^{J_i} a^{s}_{1(i,j)} + \sum_{m\in M}\sum_{(i,j)\in Z_m}\sum_{\substack{(k,l)\in Z_m\\(k,l)\neq(i,j)}} a^{(m,s)}_{2(i,j,k,l)} \tag{3.20}$$

subject to:

$$t_{(i,j)}^{s} + \sum_{m \in M_{(i,j+1)}} \left(p_{(i,j+1)}^{(m,s)} x_{(i,j+1)}^{m} \right) + \sum_{e \in M_{(i,j)}} \sum_{f \in M_{(i,j+1)}} d_{(e,f)} v_{(i,j,j+1)}^{(e,f)} - a_{1(i,j+1)}^{s} \leq t_{(i,j+1)}^{s},$$
$$\forall i = 1, \ldots, N, j = 1, \ldots, J_i - 1 \tag{3.21}$$

$$r_i + \sum_{m \in M_{(i,1)}} \left(p_{(i,1)}^{(m,s)} x_{(i,1)}^{m} \right) - a_{1(i,1)}^{s} \leq t_{(i,1)}^{s}, \quad \forall i = 1, \ldots, N \tag{3.22}$$

$$t_{(i,j)}^{s} + p_{(k,l)}^{(m,s)} + u_{(i,j,k,l)}^{m} - a_{2(i,j,k,l)}^{(m,s)} \leq t_{(k,l)}^{s} + \left(1 - y_{(i,j,k,l)}^{m}\right) H_{(i,j,k,l)}^{(m,s)},$$
$$\forall m \in M, \forall\ (i,j) \neq (k,l) \text{ in } Z_m \tag{3.23}$$

$$\sum_{j=1}^{J_i} \sum_{m \in M_{(i,j)}} c_{(i,j)}^{m} p_{(i,j)}^{(m,s)} x_{(i,j)}^{m} + \left(\sum_{j=1}^{J_i - 1} \sum_{e \in M_{(i,j)}} \sum_{f \in M_{(i,j+1)}} q_{(e,f)} v_{(i,j,j+1)}^{(e,f)} \right) w_i - \Delta_i^s \leq b_i,$$
$$\forall i = 1, \ldots, N \tag{3.24}$$

$$a_{1(i,j)}^{s} \geq 0, \quad \forall i = 1, \ldots, N, j = 1, \ldots, J_i \tag{3.25}$$

$$a_{2(i,j,k,l)}^{(m,s)} \geq 0, \quad \forall m \in M, \forall\ (i,j) \neq (k,l) \in Z_m \tag{3.26}$$

$$t_{(i,j)}^{s} \geq 0, \quad \forall i = 1, \ldots, N, j = 1, \ldots, J_i \tag{3.27}$$

$$\Delta_i^s \geq 0, \quad \forall i = 1, \ldots, N. \tag{3.28}$$

Note that artificial variables are included in Constraints (3.2), (3.3), and (3.4), which now become (3.21), (3.22), and (3.23), respectively. Constraints (3.9) do not require any artificial variables because they can always be satisfied by virtue of the budget over-run variables Δ_i^s, $i = 1, \ldots, N$. Whereas the corresponding restrictions are included in (3.24), they can be effectively omitted from Problem ARP.

If the value of the objective function (3.20) in ARP equals zero for all the subproblems, then it indicates that the solution from the master program (first-stage) is feasible to the recourse (second-stage) problem. However, if there exists a scenario, say $\bar{s}$, such that the subproblem corresponding to this scenario has a positive optimal objective value, then a feasibility cut is generated so as to eliminate the corresponding solution from the master program, as follows. Rewriting Constraints (3.21), (3.22), and (3.23) as "$\geq$" inequalities, we associate nonnegative dual variables $\eta_{(i,j)}^{\bar{s}}$, $\eta_i^{\bar{s}}$, $\eta_{(i,j,k,l)}^{(m,\bar{s})}$ with these respective constraints. Note that (3.24) has been dropped from Problem ARP. Then, we derive the following feasibility cut for scenario $\bar{s}$:

$$\sum_{i=1}^{N}\sum_{j=1}^{J_i-1}\eta_{(i,j)}^{\bar{s}}\left(\sum_{m\in M_{(i,j+1)}} p_{(i,j+1)}^{(m,\bar{s})}x_{(i,j+1)}^{m}+\sum_{e\in M_{(i,j)}}\sum_{f\in M_{(i,j+1)}} d_{(e,f)}v_{(i,j,j+1)}^{(e,f)}\right)$$
$$+\sum_{i=1}^{N}\eta_i^{\bar{s}}\sum_{m\in M_{(i,1)}} p_{(i,1)}^{(m,\bar{s})}x_{(i,1)}^{m}+\sum_{m\in M}\sum_{(i,j)\in Z_m}\sum_{\substack{(k,l)\in Z_m\\(k,l)\neq(i,j)}}\eta_{(i,j,k,l)}^{(m,\bar{s})}H_{(i,j,k,l)}^{(m,\bar{s})}y_{(i,j,k,l)}^{m}$$
$$\leq -\sum_{i=1}^{N}\eta_i^{\bar{s}}r_i+\sum_{m\in M}\sum_{(i,j)\in Z_m}\sum_{\substack{(k,l)\in Z_m\\(k,l)\neq(i,j)}}\eta_{(i,j,k,l)}^{(m,\bar{s})}\left(H_{(i,j,k,l)}^{(m,\bar{s})}-p_{(k,l)}^{(m,\bar{s})}-u_{(i,j,k,l)}^{m}\right). \tag{3.29}$$

This feasibility cut is appended to the master program. Whenever the objective function values for all the augmented subproblems equal zero, the Stage-I solution yields feasible Stage-II recourse problems, whence we either verify optimality or generate optimality cuts as described next.

3.2.3 *Optimality Cuts*

When a Stage-I solution $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{v}})$ is feasible for the separable Stage-II problems, the latter effectively determine optimal values for the completion time and budget over-run variables for each scenario. This yields the expected recourse value of the Stage-II objective function as given by $\mathbf{Q}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{v}}) \equiv \sum_{s=1}^{S}\phi_s\mathbf{Q}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{v}}, s)$. This value is then compared with the lower bound ($\bar{\theta}$, say) on the recourse value as previously obtained by solving the master problem. Note that (3.18) evaluated for $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{v}})$ provides an upper bound for the NFJS problem given the feasibility of $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{v}})$, and can be used to update the incumbent objective function value. If $\bar{\theta} \geq \mathbf{Q}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{v}})$, we have that $(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{v}})$ is an optimal solution to the NFJS problem. Otherwise, if $\bar{\theta} < \mathbf{Q}(\bar{\mathbf{x}}, \bar{\mathbf{y}}, \bar{\mathbf{v}})$, we generate an optimality cut to help close the gap between the two bounds. Letting $\xi_{(i,j)}^{s}$, ξ_i^{s}, $\xi_{(i,j,k,l)}^{(m,s)}$, and ω_i^{s} be the nonnegative dual variables associated with respect to Constraints (3.2), (3.3), (3.4), and (3.9) written as $\geq$ restrictions, the optimality cut is given as follows, where, as mentioned above, θ is used to represent the final term in the objective function (3.18) of the master program:

$$\theta \geq \sum_{s=1}^{S} \phi_s \left\{ \sum_{i=1}^{N} \sum_{j=1}^{J_i-1} \xi_{(i,j)}^{s} \left(\sum_{m \in M_{(i,j+1)}} p_{(i,j+1)}^{(m,s)} x_{(i,j+1)}^{m} + \sum_{e \in M_{(i,j)}} \sum_{f \in M_{(i,j+1)}} d_{(e,f)} v_{(i,j,j+1)}^{(e,f)} \right) \right.$$
$$+ \sum_{i=1}^{N} \xi_i^s \sum_{m \in M_{(i,1)}} p_{(i,1)}^{(m,s)} x_{(i,1)}^{m} + \sum_{m \in M} \sum_{(i,j) \in Z_m} \sum_{\substack{(k,l) \in Z_m \\ (k,l) \neq (i,j)}} \xi_{(i,j,k,l)}^{(m,s)} H_{(i,j,k,l)}^{(m,s)} y_{(i,j,k,l)}^{m}$$
$$+ \sum_{i=1}^{N} \omega_i^s \left[\sum_{j=1}^{J_i} \sum_{m \in M_{(i,j)}} c_{(i,j)}^{m} p_{(i,j)}^{(m,s)} x_{(i,j)}^{m} + \left(\sum_{j=1}^{J_i-1} \sum_{e \in M_{(i,j)}} \sum_{f \in M_{(i,j+1)}} q_{(e,f)} v_{(i,j,j+1)}^{(e,f)} \right) w_i \right]$$
$$+ \sum_{i=1}^{N} \xi_i^s r_i - \sum_{m \in M} \sum_{(i,j) \in Z_m} \sum_{\substack{(k,l) \in Z_m \\ (k,l) \neq (i,j)}}$$
$$\xi_{(i,j,k,l)}^{(m,s)} \left(H_{(i,j,k,l)}^{(m,s)} - p_{(k,l)}^{(m,s)} - u_{(i,j,k,l)}^{m} \right) - \sum_{i=1}^{N} \omega_i^s b_i. \tag{3.30}$$

The optimality cut is appended to the MP and the revised MP is re-solved. The iterations continue in this fashion until the lower and upper bounds converge (or come within a desired optimality tolerance).

Note that the master problem and the linear programs corresponding to the subproblems need to be re-solved every time a new feasibility or optimality cut is added. This can lead to a lengthy process in case a large number of feasibility cuts are required to generate a feasible solution. Therefore, it is helpful to a priori include suitable valid inequalities in the master problem to induce second-stage feasibility. We present such inequalities next.

3.2.4 Alternative Valid Inequalities for Inducing Stage-II Feasibility That Also Provide a Stage-I Lower Bound

The alternative set of valid inequalities derived in this section relies on the fact that for any fixed value of (**x**, **y**, **v**), the feasibility of the Stage-II problem does not depend on a particular scenario. In other words, if the routing and sequencing decisions are feasible for a given scenario, then they are also feasible for any other scenario, because changes in job processing times can be accommodated by adjusting completion times while maintaining the feasibility of the subproblem constraints. Consequently, variables and constraints of the subproblem for a given scenario can be included in the master problem to induce feasibility of the subproblems for all the scenarios. The next result indicates that the particular scenario that use the expected values of processing times provides a lower bound on $\theta \equiv \mathbf{Q}(\mathbf{x}, \mathbf{y}, \mathbf{v})$.

Proposition 1 The optimal objective value of the subproblem with expected processing times yields a lower bound on $\mathbf{Q}(\mathbf{x}, \mathbf{y}, \mathbf{v}) \equiv \sum_{s=1}^{S} \phi_s \mathbf{Q}(\mathbf{x}, \mathbf{y}, \mathbf{v}, s)$.

Proof For any fixed values of $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{v}$, the recourse function $\mathbf{Q}(\mathbf{x}, \mathbf{y}, \mathbf{v}, s)$ is only a function of s. We rewrite this as $\mathbf{Q}(\boldsymbol{p}^s)$, where $\boldsymbol{p}^s$ is the vector of operation processing times.

Let $\boldsymbol{p}^E = \sum_{s=1}^{S} \phi_s \boldsymbol{p}^s$. The fact that $\mathbf{Q}(\boldsymbol{p}^E) = \mathbf{Q}\left(\sum_{s=1}^{S} \phi_s \boldsymbol{p}^s\right) \leq \sum_{s=1}^{S} \phi_s \mathbf{Q}(\boldsymbol{p}^s)$ is easily established because $\mathbf{Q}(\boldsymbol{p}^s)$ is a convex function of $\boldsymbol{p}^s$, due to fixed recourse (see Theorem 5 in Birge and Louveaux 2000, p. 89.) □

Accordingly, we define the following variables:

$t_{(i,j)}^E$ = completion time of operation j of job i under expected processing times.
Δ_i^E = budget over-run for job i under expected processing times.

Then, by Proposition 1 and Constraints (3.2), (3.3), (3.4), and (3.9), we include the following set of restrictions in the Stage-I master program:

$$\theta \geq \sum_{i=1}^{N} \alpha_i t_{(i,J_i)}^E + \sum_{i=1}^{N} \beta_i \Delta_i^E \tag{3.31}$$

$$t_{(i,j)}^E + \sum_{m \in M_{(i,j+1)}} \left(p_{(i,j+1)}^{(m,E)} x_{(i,j+1)}^m \right) + \sum_{e \in M_{(i,j)}} \sum_{f \in M_{(i,j+1)}} d_{(e,f)} v_{(i,j,j+1)}^{(e,f)} \leq t_{(i,j+1)}^E,$$
$$\forall i = 1, \ldots, N, \;\; j = 1, \ldots, J_i - 1 \tag{3.32}$$

$$r_i + \sum_{m \in M_{(i,1)}} \left(p_{(i,1)}^{(m,E)} x_{(i,1)}^m \right) \leq t_{(i,1)}^E, \quad \forall i = 1, \ldots, N \tag{3.33}$$

$$t_{(i,j)}^E + p_{(k,l)}^{(m,E)} + u_{(i,j,k,l)}^m \leq t_{(k,l)}^E + \left(1 - y_{(i,j,k,l)}^m\right) H_{(i,j,k,l)}^{(m,E)},$$
$$\forall m \in M, \forall\, (i,j) \neq (k,l) \in Z_m \tag{3.34}$$

$$\sum_{j=1}^{J_i} \sum_{m \in M_{(i,j)}} c_{(i,j)}^m p_{(i,j)}^{(m,E)} x_{(i,j)}^m + \left(\sum_{j=1}^{J_i-1} \sum_{e \in M_{(i,j)}} \sum_{f \in M_{(i,j+1)}} q_{(e,f)} v_{(i,j,j+1)}^{(e,f)} \right) w_i - \Delta_i^E \leq b_i,$$
$$\forall i = 1, \ldots, N \tag{3.35}$$

$$t_{(i,j)}^E \geq 0, t_{(i,j)}^E \geq 0, \quad \forall i = 1, \ldots, N, j = 1, \ldots, J_i \tag{3.36}$$

$$\Delta_i^E \geq 0, \quad \forall i = 1, \ldots, N. \tag{3.37}$$

Note that $p_{(i,j)}^{(m,E)} \equiv \sum_{s=1}^{S} \phi_s p_{(i,j)}^{(m,s)}$, and that $H_{(i,j,k,l)}^{(m,E)}$ is pre-calculated similar to that in (3.16) and (3.17), with $p_{(i,j)}^{(m,E)}$ replacing $p_{(i,j)}^{(m,s)}$ in the expressions.

Also note that the infeasibility caused by re-entrant flow will be eliminated by Constraints (3.32), (3.33), and (3.34), since they enforce a proper ordering (via the $t_{(i,j)}^{E}$ -variables) among the operations of the same job. These constraints also prevent the occurrence of a deadlock as follows. Consider the situation depicted in Fig. 3.1, where we have $x_{(a,b)}^{M1} = x_{(c,d)}^{M1} = x_{(c,h)}^{M2} = x_{(a,f)}^{M2} = 1$ and $y_{(a,b,c,d)}^{M1} = y_{(c,h,a,f)}^{M2} = 1$. Since $h > d$ and $f < b$, the constraint set (3.32) asserts that $t_{(c,h)}^{E} > t_{(c,d)}^{E}$ and $t_{(a,f)}^{E} < t_{(a,b)}^{E}$. On the other hand, the constraint set (3.34) enforces $t_{(a,b)}^{E} < t_{(c,d)}^{E}$ and $t_{(c,h)}^{E} < t_{(a,f)}^{E}$. Clearly, these four inequalities lead to a contradiction, and consequently, the corresponding values of the **x**- and **y**-variables would be infeasible to the master problem augmented with Constraints (3.32) and (3.34).

Note that (3.34) also serves to eliminate subtours among the operations processed on a machine. These are essentially the MTZ-type of subtour elimination constraints (Miller et al. 1960), and they can be weak in the sense that they lead to loose LP relaxations. However, they can potentially be strengthened through the use of flow-based valid inequalities as shown by Sherali et al. (2006).

3.3 Valid Inequalities for Further Tightening the Model Formulation

In this section, we develop three classes of valid inequalities by exploiting the inherent structure of the NFJS problem. These inequalities are added to the MP to tighten its continuous relaxation and provide better lower bounds for use in the branch-and-bound algorithm for solving NFJSP. The first type of inequalities arises from the flow balance-type constraints that capture the movement of operations among the machines. The other two types of inequalities are formulated to obviate infeasibility caused by re-entrant flow and deadlock, respectively. They are based on a new formulation for the asymmetric travelling salesman problem (ATSP) presented in Sarin et al. (2005).

3.3.1 Flow-Balance Constraints

In the NFJSP formulation, we used the variable $v_{(i,j,j+1)}^{(e,f)}$ to represent the transfer of job i from machine e to machine f when performing the respective operations j and $j+1$. This definitional role of $v_{(i,j,j+1)}^{(e,f)}$ is enforced by (3.10). We can further tighten the continuous relaxation of the model by introducing the following flow-balance constraints (FBC):

$$\sum_{f \in M_{(i,2)}} v_{(i,1,2)}^{(e,f)} = x_{(i,1)}^{e}, \quad \forall i = 1, \ldots, N, e \in M_{(i,1)} \tag{3.38}$$

$$\sum_{e \in M_{(i,j-1)}} v_{(i,j-1,j)}^{(e,f)} = \sum_{g \in M_{(i,j+1)}} v_{(i,j,j+1)}^{(f,g)}, \quad \forall i = 1, \ldots, N, j = 2, \ldots, J_i - 1, f \in M_{(i,j)} \tag{3.39}$$

$$\sum_{e \in M_{(i,J_i-1)}} v_{(i,J_i-1,J_i)}^{(e,f)} = x_{(i,J_i)}^{f}, \quad \forall i = 1, \ldots, N, f \in M_{(i,J_i)}. \tag{3.40}$$

Constraints (3.38) assert that if the first operation of job i is assigned to machine e, then job i must be transported from machine e to some machine f that is capable of processing the second operation of job i. Constraints (3.39) capture the fact that if machine f is chosen for processing the jth operation of job i, $1 < j < J_i$, then job i is necessarily transferred from some previous machine, e, and is transported to a succeeding machine, g, while performing the respective operations $j - 1$ and $j + 1$. Similarly, Constraints (3.40) require job i to be transferred from some previous machine e in case its last operation is processed on machine f.

3.3.2 *Re-Entrant Flow-Based Constraints*

In the case of re-entrant flows, the lower indexed operations of any job must precede the higher indexed operations of that job for the sequence to be feasible. For the sake of convenience, we designate an order $ord(i, j)$ for elements of Z_m, $\forall m \in M$, such that the ordering of operations from the same job is maintained. For instance, if $Z_m = \{(1, 1), (2, 2), (1, 2)\}$, we can assign $ord(1, 1) = 1$, $ord(2, 2) = 2$, and $ord(1, 2) = 3$. Based on this definition, we let

$$h_{(i,j,k,l)}^{m} \equiv x_{(i,j)}^{m} x_{(k,l)}^{m}, \forall m \in M; (i,j), (k,l) \in Z_m : ord(i,j) < ord(k,l)$$

We can linearize the foregoing relationship between the **h**- and the **x**-variables by using the following logical constraints:

$$h_{(i,j,k,l)}^{m} \le x_{(i,j)}^{m}, \quad h_{(i,j,k,l)}^{m} \le x_{(k,l)}^{m}, \quad h_{(i,j,k,l)}^{m} \ge x_{(i,j)}^{m} + x_{(k,l)}^{m} - 1,$$
$$\forall m \in M, (i,j), (k,l) \in Z_m : ord(i,j) < ord(k,l). \tag{3.41}$$

We also define certain *indirect precedence variables* as follows:

$$g_{(i,j,k,l)}^{m} = \begin{cases} 1, & \text{if operation } j \text{ of job } i \text{ is processed sometime before} \\ & \text{operation } l \text{ of job } k \text{ on machine } m, \\ 0, & \text{otherwise.} \end{cases}$$

Then, we have,

$$g^m_{(i,j,k,l)} + g^m_{(k,l,i,j)} = h^m_{(i,j,k,l)}, \quad \forall m \in M, (i,j), (k,l) \in Z_m : ord\,(i,j) < ord\,(k,l) \tag{3.42}$$

$$g^m_{(i,j,\gamma,\eta)} \geq g^m_{(i,j,k,l)} + g^m_{(k,l,\gamma,\eta)} - 1, \quad \forall m \in M, \forall \text{distinct } (i,j), (k,l), (\gamma,\eta) \in Z_m \tag{3.43}$$

$$g^m_{(i,j,i,l)} = 0, \quad \forall m \in M, (i,j), (i,l) \in Z_m : j > l. \tag{3.44}$$

Constraints (3.42) state that given two job-operations on a machine, one of them must either precede or succeed the other. Constraints (3.43) represent the transitivity property; that is, for any triplet of job-operations (i,j), (k,l), and (γ,η) on machine m, if operation (i,j) is scheduled somewhere before operation (k,l) and operation (k,l) is scheduled somewhere before operation (γ,η), then operation (i,j) must necessarily be scheduled before operation (γ,η). Finally, the re-entrant flow Constraints (3.44) ensure that if two operations of the same job are assigned to a machine, then the lower indexed job operation precedes the higher indexed operation.

Also, we have the following logical constraints connecting the indirect and the direct precedence variables:

$$g^m_{(i,j,k,l)} \geq y^m_{(i,j,k,l)}, \quad \forall m \in M, \forall\,(i,j) \neq (k,l) \text{ in } Z_m. \tag{3.45}$$

Hence, the re-entrant flow constraints that can be accommodated into the master (Stage-I) program are given by (3.41), (3.42), (3.43), (3.44), and (3.45). Note that by introducing the $g^m_{(i,j,k,l)}$ -variables, we also eliminate infeasibility caused by subtours in operation sequencing, since the indirect precedence enforced by the $g^m_{(i,j,k,l)}$ -variables precludes the occurrence of subtours.

3.3.3 Deadlock Prevention Constraints

Next, we develop valid inequalities to prevent the occurrence of a deadlock. For the sake of brevity, we only present inequalities for the prevention of 2-machine deadlocks and establish their validity. For a detailed development of the corresponding results for the general case of m-machine deadlocks, see Varadarajan (2006).

Consider the following situation in a job shop environment: operations (a, b) and (c, d) are assigned to machine m; and operations (a, f) and (c, h), where $f < b$ and $h > d$, are assigned to machine n. If (a, b) precedes (c, d), then (a, f) must necessarily precede (c, h) to avoid a deadlock (see Fig. 3.1, where machines m and n are denoted by M1 and M2, respectively). Then, we have the following result:

Proposition 2 Two-machine deadlocks are prevented by including the following additional inequalities:

$$\begin{aligned} & g^n_{(a,f,c,h)} \geq g^m_{(a,b,c,d)} + h^n_{(a,f,c,h)} - 1, \\ & \quad \forall\, m, n \in M, (a, b), (c, d) \in Z_m, (a,f), (c, h) \in Z_n, \\ & \quad f < b \text{ and } h > d. \end{aligned} \tag{3.46}$$

Proof If $h^n_{(a,f,c,h)} = 0$, then $g^n_{(a,f,c,h)} = 0$ by (3.42), and $g^m_{(a,b,c,d)}$ can be 1 without causing any deadlock. If $h^n_{(a,f,c,h)} = 1$ and $g^m_{(a,b,c,d)} = 0$, then $g^n_{(a,f,c,h)} \geq 0$, and the schedule is deadlock free. On the other hand, if both $g^m_{(a,b,c,d)}$ and $h^n_{(a,f,c,h)}$ are equal to 1, then (a, b) is processed sometime before (c, d) on machine m, and (a, f) and (c, h) are processed on the same machine n. To yield a deadlock-free schedule under this situation, (a, f) must be processed sometime before (c, h) on machine n, which is enforced by (3.46). □

Note that to apply the above deadlock prevention constraints to the master problem, we need to also include Constraints (3.41), (3.42), (3.43), and (3.44), so that $\boldsymbol{g}$- and $\boldsymbol{h}$-variables take their definitional roles in the model.

3.4 Computational Results

We now present results of computational experimentation to demonstrate the effectiveness of our feasibility-inducing and model-tightening inequalities within the framework of the L-shaped method for the solution of NFJS problem. In this method, the Stage-I master problem is solved using a branch-and-bound algorithm. Whenever an integer solution is obtained for a node problem's LP relaxation, the Stage-II problem is solved to verify its feasibility and optimality. If any of these conditions are not met, a feasibility cut or an optimality cut is generated and added to the Stage-I problem, and the branch-and-bound process continues.

There are several ways in which the valid inequalities pertaining to the expected value scenario (EVS), the re-entrant flows (RF), and deadlock prevention (DP) (developed in Sects. 3.2.4, 3.3.2, and 3.3.3, respectively) can be applied. We can either use them separately, or we can apply the EVS inequalities in conjunction with selected members of the RF and DP inequalities in order to tighten the underlying relaxation. Our preliminary investigation has shown that the use of the EVS inequalities always leads to shorter CPU times. The question, then, is how (if at all) to apply the RF and DP inequalities in addition to the EVS inequalities. Note that, to achieve the full potential of the RF and DP inequalities, we need to consider re-entrant flows and deadlocks among all the machines, which would require a large number of extra variables and constraints that may overburden the master program and deteriorate its computational performance. Therefore, we choose to apply these inequalities to a proper subset of machines, as investigated in Sect. 3.4.2 below. Furthermore, we explore the optional addition of the flow-balance constraints FBC of Sect. 3.3.1.

3.4.1 Design of Test Problems

To represent routing flexibility, we group machines into work centers; operations assigned to a work center are allowed to be processed by any machine in that work center. Due to this feature, the manner in which workload is assigned to the machines within a work center is not determined until a solution is obtained. To indicate the potential workload on a machine, we define a *load factor* to be the total number of visits of all the jobs to that machine on average, assuming that all the machines within a work center equally share the workload. According to the load factor, we differentiate machines into two categories: low-number-of-visit (LNV) machines (with two potential visits on average), or high-number-of-visit (HNV) machines (with three potential visits on average). Consequently, three job-visiting patterns are considered, pertaining to different distributions of workload on the machines. These are: {"L^+•H^-," "L•H," "L^-•H^+"}. The letters "L" and "H" refer to the LNV and HNV machines, respectively; the plus/minus signs in the superscript indicate that, relatively, there are higher or lower number of machines in a category than those in the other. We consider test problems of various sizes, involving 6, 8, or 10 machines. Their basic specifications are listed in Table 3.1.

With respect to routing flexibility, three cases are considered. In Case ρ_1, all HNV machines are grouped into one work center, while no routing flexibility exists among the LNV machines. In Case ρ_2, all LNV machines are grouped into one work center; no routing flexibility exists among the HNV machines. In Case ρ_3, no routing flexibility exists.

Table 3.1 Specifications for various problem sizes

Number of machines	Job-visiting pattern	Number of LNV machines	Number of HNV machines	Number of jobs	Total number of operations
6	L^+•H^-	4	2	3	14
	L•H	3	3	3	15
	L^-•H^+	2	4	3	16
8	L^+•H^-	5	3	4	22
	L•H	4	4	4	24
	L^-•H^+	3	5	4	26
10	L^+•H^-	6	4	5	28
	L•H	5	5	5	30
	L^-•H^+	4	6	5	32

For each of the above $3 \times 3 \times 3 = 27$ combinations of numbers of machines, job-visiting patterns, and routing flexibility, we constructed 20 test problems with randomly generated job routings and processing times. For the 6-machine problems, we considered the following numbers of scenarios: $\{100, 200, 300, 400\}$. The larger-sized problems (involving 8 and 10 machines) were solved using 400 scenarios to reveal the effectiveness of the proposed strategy.

All experimental runs were implemented using AMPL-CPLEX 10.1 and performed on a Pentium D 3.2 GHz CPU computer with 2 GB memory.

3.4.2 Experimental Results

We first compare the performance of solving NFJSP directly by AMPL-CPLEX (designated as **Method I**) with that of our decomposition approach (designated as **Method II**), which includes the EVS inequalities but not the RF and DP inequalities. Results of the L-shaped method without any additional inequalities, i.e., only with the standard feasibility and optimality cuts (3.29) and (3.30) (designated as **Method III**) are also provided for comparison. In addition, we considered the option of either adding or not adding the FBC inequalities of Sect. 3.3.1 to these three methods. The 6-machine problems were solved to optimality; the 8- and 10-machine problems were run until an integer solution was obtained within an optimality gap of 5 %. Since the superiority of Method II was observed in our preliminary study, we adopted the following approach to avoid excessive run times: Method II was used to solve the test problems first. Since there are two options (with or without the FBC inequalities), we record the CPU time as t_2' and t_2'', respectively, for these options. Let $t_2 = \max\{t_2', t_2''\}$. For Methods I and III, we set an upper bound on the CPU time of $\max\{1.2 \times t_2, t_c\}$, where the value of t_c is 1500, 2000, and 2500 s, respectively, for the 6-, 8-, and 10-machine problems. The first term $(1.2 \times t_2)$ is used to provide a reasonable (20 %) margin to demonstrate the superiority of Method II over the other two methods. The second term (t_c) is included to ensure that the effectiveness of adding the FBC inequalities is not obscured by stopping prematurely.

The results obtained are presented in Table 3.2. Note that the inclusion of the FBC inequalities results in shorter CPU times for Methods I and II in most cases. Therefore, in the following discussion, we only provide results for the case where the FBC inequalities have been added to the NFJSP formulation. From these results, it is also evident that Method II is substantially faster than Methods I and III.

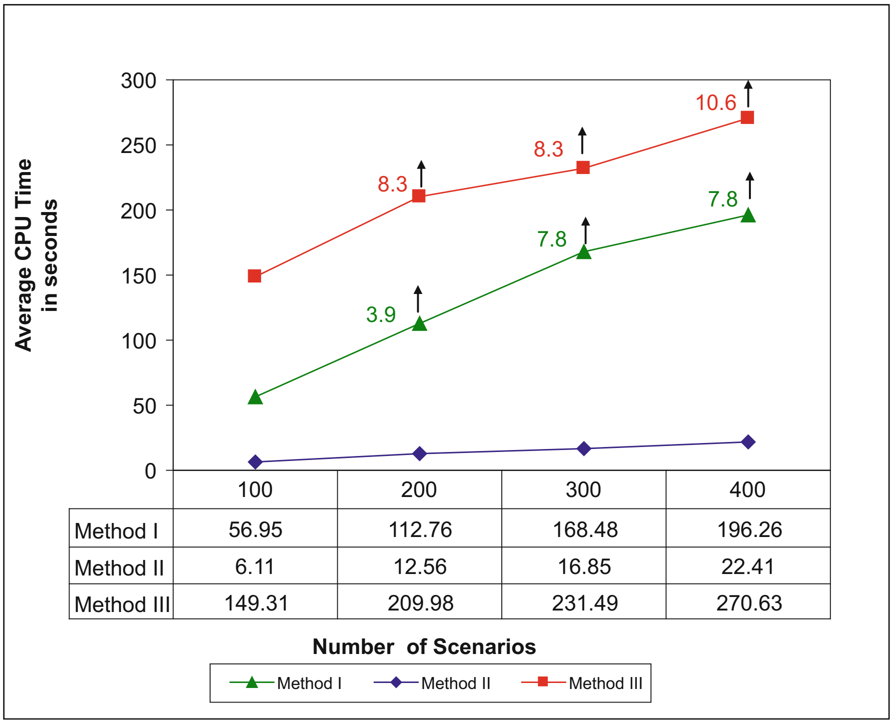
To further illustrate how this dominance varies with an increasing number of scenarios, we present, in Fig. 3.2, the results for 6-machine problems with four different numbers of scenarios, namely, 100, 200, 300, and 400. The arrow and the number next to a data point indicate the percentage of test problems that consume CPU times more than the limit of 1500 s. Note that Method II substantially dominates the other methods as the number of scenarios increases from 100 to 400. This pattern is observed not only for the average values that are depicted in

Table 3.2 Computational results for Methods I, II, and III for 400 scenarios

	Average CPU time (in seconds)					
	Method I		Method II		Method III	
Number of machines	w/o FBC	w/ FBC	w/o FBC	w/ FBC	w/o FBC	w/ FBC
6	207.53	196.26	21.38	22.41	266.82	270.63
8	742.29	707.98	27.42	25.09	1365.04	1379.96
10[a]	1315.04	1306.59	82.81	54.63	2244.51	2242.22
10[b]	–	–	123.55	109.87	2273.06	2271.03

[a]Twenty-one (out of 180) 10-machine problems could not be solved by Method I due to excessive memory requirements. The average values in this row are calculated based on the remaining 159 test problems

[b]One (out of the 180) 10-machine problem could not be solved by Method III due to excessive memory requirements. The average values in this row are calculated based on the other 179 test problems



	100	200	300	400
Method I	56.95	112.76	168.48	196.26
Method II	6.11	12.56	16.85	22.41
Method III	149.31	209.98	231.49	270.63

Fig. 3.2 Average CPU times for 6-machine problems for various numbers of scenarios

Fig. 3.2, but also for all combinations of job-visiting patterns and cases of routing flexibility. For the sake of brevity, we illustrate this behavior in Fig. 3.3 by using the combination "L•H" × "ρ_2". Clearly, Method II dominates the other two methods for all numbers of scenarios considered. Moreover, as the number of scenarios

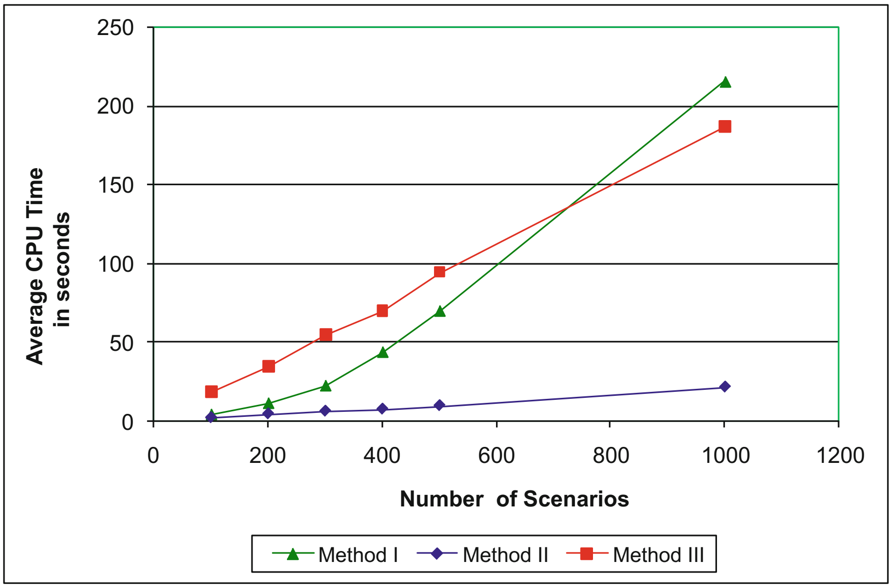


Fig. 3.3 Average CPU times required by the combination "L•H" × "(*ii*)" for 6-machine problems for various numbers of scenarios

increases, the CPU time required by Method II increases almost linearly, while that for Method I increases superlinearly. Although the CPU time for Method III also increases almost linearly, it does so at a much higher rate than that for Method II. Hence, the dominance of Method II becomes more prominent with an increase in the number of scenarios. Also, note that Method III begins to dominate Method I for larger number of scenarios.

Next, we investigated the impact of adding the RF and DP inequalities to Method II. There are several options that we can consider. The default option is Method II with the FBC inequalities. We can either add to this default option the RF and DP inequalities separately, or we can add them both at the same time. Another aspect to consider is the set of machines to which these inequalities are applied. We considered two options in this respect, namely, their application to only the HNV machines, or to only the LNV machines. Our preliminary investigation showed that the application of the RF and DP inequalities to the HNV machines results in the addition of a large number of extra variables and constraints to the Stage-I master problem, which leads to greater CPU times in comparison with the default option. Therefore, in the following discussion, we only consider the addition of the RF and DP inequalities to the LNV machines. To compare the performances of different model configurations, we fixed the number of scenarios at 400. The average CPU times and the ratio between the values of the LP solution and the 1. best-found solution, under different job-visiting patterns and cases of routing flexibility, are summarized in Tables 3.3 and 3.4.

Table 3.3 Average CPU times and percentage improvements over Model A

Model			A (default)	B (default + RF)	C (default + DP)	D (default + RF+ DP)
Number of machines	Job-visiting pattern	Routing flexibility	Average CPU time (in seconds), percentage improvement			
6	$L^+ \bullet H^-$	ρ_1	7.48	7.7	7.64	7.7
		ρ_2	17.48	22.12	25.52	23.6
		ρ_3	1.74	1.77	**1.74, 0.1 %**	1.77
	L•H	ρ_1	27.05	28.31	**26.4, 2.4 %**	28.32
		ρ_2	7.75	9.73	9.36	9.85
		ρ_3	3.09	3.09	3.11	3.13
	$L^- \bullet H^+$	ρ_1	130.12	**123.66, 5.0 %**	**124.15, 4.6 %**	**123.95, 4.7 %**
		ρ_2	4.16	4.49	4.25	4.49
		ρ_3	2.8	**2.77, 0.8 %**	2.81	**2.78, 0.8 %**
8	$L^+ \bullet H^-$	ρ_1	3.81	4.08	3.84	3.95
		ρ_2	5.78	8.97	6.85	9.44
		ρ_3	0.93	0.94	0.95	0.94
	L•H	ρ_1	29	**22.86, 21.2 %**	33.37	**22.94, 20.9 %**
		ρ_2	4.67	5.74	6.92	6.6
		ρ_3	0.97	**0.94, 3.0 %**	**0.97, 0.1 %**	**0.94, 3.1 %**
	$L^- \bullet H^+$	ρ_1	176.92	196.35	**160.16, 9.5 %**	195.94
		ρ_2	2.93	**2.81, 4.0 %**	**2.78, 5.1 %**	3.03
		ρ_3	0.76	0.77	0.78	0.77
10	$L^+ \bullet H^-$	ρ_1	178.09	**74.69, 58.1 %**	252.17	**76.01, 57.3 %**
		ρ_2	18.91	29.81	51.57	42.7
		ρ_3	1.21	1.23	1.29	1.23
	L•H	ρ_1	2227.97	**1066.2, 52.1 %**	**2170.97, 2.6 %**	**1086.73, 51.2 %**
		ρ_2	29.2	47.37	41.36	36.74
		ρ_3	1.6	1.62	1.85	1.62
	$L^- \bullet H^+$	ρ_1	549.05	**476.28, 13.3 %**	662.12	**536.86, 2.2 %**
		ρ_2	11.21	**10.89, 2.9 %**	15.28	15.37
		ρ_3	1.55	**1.51, 2.4 %**	**1.54, 1.1 %**	**1.51, 2.6 %**
Total average	127.64	79.88	134.06	83.29		

We highlight in bold, in Table 3.3, the CPU times that turn out to be shorter than that for the default model (A) for a given combination of job-visiting pattern and routing flexibility. For each such case, the percentage improvement in CPU time over the default model is also presented along with the CPU time. In view of these results, we can make the following observations: Simlarly a higher ratio between the values of the LP solution and the best-found solution obtained for a method over default is highlighted in Table 3.4

Table 3.4 Comparing quality of LP relaxation

Model			A (default)	B (default + RF)	C (default + DP)	D (default + RF+ DP)
Number of machines	Job-visiting pattern	Routing flexibility	LP relaxation value/Best solution found			
6	$L^{+}\bullet H^{-}$	ρ_1	92.52 %	92.52 %	92.52 %	92.52 %
		ρ_2	93.52 %	**94.13 %**	**93.85 %**	**94.13 %**
		ρ_3	95.00 %	95.00 %	95.00 %	95.00 %
	L•H	ρ_1	94.00 %	94.00 %	94.00 %	94.00 %
		ρ_2	92.62 %	**92.96 %**	**92.80 %**	**92.96 %**
		ρ_3	91.58 %	91.58 %	91.58 %	91.58 %
	$L^{-}\bullet H^{+}$	ρ_1	94.45 %	94.45 %	94.45 %	94.45 %
		ρ_2	92.59 %	**92.78 %**	**92.75 %**	**92.78 %**
		ρ_3	92.46 %	92.46 %	92.46 %	92.46 %
8	$L^{+}\bullet H^{-}$	ρ_1	93.92 %	93.92 %	93.92 %	93.92 %
		ρ_2	92.88 %	**93.25 %**	**93.10 %**	**93.25 %**
		ρ_3	93.95 %	93.95 %	93.95 %	93.95 %
	L•H	ρ_1	94.83 %	94.83 %	94.83 %	94.83 %
		ρ_2	92.74 %	**93.00 %**	**92.91 %**	**93.00 %**
		ρ_3	94.48 %	94.48 %	94.48 %	94.48 %
	$L^{-}\bullet H^{+}$	ρ_1	93.24 %	93.24 %	93.24 %	93.24 %
		ρ_2	93.70 %	**93.80 %**	**93.78 %**	**93.80 %**
		ρ_3	93.99 %	93.99 %	93.99 %	93.99 %
10	$L^{+}\bullet H^{-}$	ρ_1	94.19 %	94.19 %	94.19 %	94.19 %
		ρ_2	92.98 %	**93.29 %**	**93.20 %**	**93.29 %**
		ρ_3	94.31 %	94.31 %	94.31 %	94.31 %
	L•H	ρ_1	93.59 %	93.59 %	93.59 %	93.59 %
		ρ_2	93.78 %	**94.03 %**	**93.94 %**	**94.03 %**
		ρ_3	93.12 %	93.12 %	93.12 %	93.12 %
	$L^{-}\bullet H^{+}$	ρ_1	92.38 %	92.38 %	92.38 %	92.38 %
		ρ_2	92.69 %	**92.87 %**	**92.82 %**	**92.87 %**
		ρ_3	94.47 %	94.47 %	94.47 %	94.47 %

1. On average, the addition of the RF inequalities alone (Model B) and the addition of both types of inequalities (Model D) help in achieving a shorter CPU time. This is particularly true when routing flexibility occurs on the HNV machines (Case ρ_1) because it leads to fewer conflicts for a large number of operations. This phenomenon appears to become more prominent with an increase in the number of machines, where savings of up to 58.1 % are achieved by Model B and up to 57.3 % by Model D for the 10-machine problems.
2. The benefit of adding the RF and DP inequalities is more evident for the harder problems, i.e., when the default model (Model A) takes a longer time to solve a problem, the addition of the RF and DP inequalities is more likely to help reduce the CPU time.

3.5 Heuristic Methods for the Solution of the NFJS Problem

Several heuristic methods can be used for the solution of the NFJSP that rely on work presented above. Six such viable procedures are described and investigated below.

1. Expected Value Problem Heuristic (EVP)

1. Assume processing times to be deterministic and equal to their expected values, and solve the model to optimality record the solution.
2. Evaluate solution using all scenarios.

2. Mixed Heuristic (Mixed)

Note that, even if all the scenarios are considered in the determination of budget over-runs, the resulting model is still relatively easy to solve (as the number of relevant constraints/variables is equal to *NS*). Hence, we can consider all scenarios in the determination of budget over-runs, while using expected processing times for the determination of job completion times. We call the resulting model a mixed-type model. Since this is a closer approximation of the original problem, we expect it to yield better solutions than the expected value heuristic.

For large-sized problem instances, even the mixed-type model becomes very difficult to solve. Therefore, we further relax the sequencing variables to be continuous and solve the mixed-type model to obtain assignment decisions. Then, we determine job sequences by considering the outcomes of all scenarios.

1. Solve the mixed-type model with sequencing variables relaxed as continuous; (assignment step).
2. Determine the processing sequences using the assignment decisions fixed in Step 1; (sequencing step).

3. Mixed + Shortest Processing Time (SPT)

The second step of the mixed heuristic is still difficult to solve for large-sized problems, hence we can apply the SPT dispatching rule to determine the job sequence. That is, whenever a machine is released by a previous operation, we choose the operation that has the shortest processing time on that machine from among the waiting operations. Note that, if an operation can be processed on multiple machines, it is considered to be waiting on all the compatible machines. Its assignment is determined based on which machine first chooses it as the next operation to be processed. Note that the SPT rule is based on the expected processing times.

4. Mixed + Least Work Remaining (LWKR)

This approach is similar to "Mixed + SPT," except that we use the least-work-remaining-first rule to dispatch operations.

5 and 6. Mixed + Shifting Bottleneck Heuristic (SBN)

As a first step, we use Step 1 of the Mixed Heuristic to determine the assignment of operations to the machines. The remaining sequencing problem is modeled as a disjunctive graph. In the beginning, all disjunctive arcs are relaxed and job completion times are recorded and regarded as due dates. The ready time and due date of each operation are determined by following the forward and backward passes along the critical paths (for the completion time problem, there are usually multiple critical paths). Next, each machine is considered individually to fix its operation sequence using a heuristic rule. The machine that yields the largest change in the total completion time is chosen, and the corresponding sequence (set of disjunctive arcs) is fixed. The procedure continues until all the machines are sequenced. Note that after the sequence of operations on each machine is fixed, a re-sequencing step is implemented by adjusting the sequences of operations on previously fixed machines.

We employ the following two heuristic rules to determine the sequence in which to process the jobs on a machine:

5. SBN_ATC Determine the following Apparent Tardiness Cost (ATC) priority index (Pinedo and Singer 1999):

$$I_{ij} = \sum_{k=1}^{N} \frac{1}{p_{ij}} \exp\left(-\frac{d_{ij}^k - p_{ij} + \left(r_{ij} - t\right)^+}{K\overline{p}}\right), \quad \forall i \in M, j = 1, \ldots, N,$$

where t is the scheduling time, $K = 2$, and $\overline{p}$ is the average of the processing times of the jobs assigned to machine i, and ready time r_{ij} and local due date d_{ij}^k of operation j of job k assigned to machine i are determined as explained in Pinedo and Singer (1999).

6. SBN_PRTT Choose the operation that has the lowest value of Chu and Portmann (1992):

$$PRTT_{ij} = \max\left(r_{ij},\ t\right) + \max\left\{\max\left\{r_{ij},\ t\right\} + p_{ij}, \quad \min_k \left\{d_{ij}^k\right\}\right\},$$

$$\forall i \in M, j = 1, \ldots, N.$$

This is a single machine sequencing rule, and we use the values of r_{ij} and d_{ij}^k as determined above. Additionally, insert the earliest available operation, if the operation chosen by the above rule leaves enough idle time before itself.

The relative performances of these heuristic methods are presented in Tables 3.5 and 3.6 for 6 machines (with three jobs) and 10 machines (with 15 jobs), respectively. Optimal gap is determined with respect to the solution of mixed-method, which serves as a lower bound. Note that, although EVP gives the best results with the least CPU time for the first set of problems, it becomes impractical for larger-sized problems (second set) due to its large number of binary sequencing variables. Mixed + LWKR heuristic gives the best results on larger-sized problem instances.

Table 3.5 Results on 6-machine problems (with 3 jobs)

Approach	Optimality gap	CPU time (seconds)
EVP	0.49 %	0.33
Mixed	–	0.27 (Step 1)
Mixed + SPT	1.46 %	1.63
Mixed + LWKR	1.42 %	1.64
Mixed + SBN_ATC	1.89 %	0.91
Mixed + SBN_PRTT	**1.27 %**	0.90

Table 3.6 Results on 10-machine problems (with 15 jobs)

Approach	Objective value	CPU time (seconds)
Mixed	2049.72	2.15 (Step 1)
Mixed + LWKR	**2796.85 (37.16 %)**	13.12
Mixed + SBN_ATC	2872.97 (40.79 %)	45.35
Mixed + SBN_PRTT	2804.47 (37.47 %)	47.32

3.6 Concluding Remarks

In this chapter, we have presented a stochastic programming approach for the NFJS (Network of Flexible Job Shops) problem. This problem arises in a distributed fabrication environment that has recently emerged to serve the evolving needs of the high investment, low volume MEMS industry. The problem is modeled as a two-stage stochastic program with recourse, where the uncertainty in processing times is captured using scenarios. The first-stage routing and sequencing variables are binary whereas the second-stage completion time and budget over-run variables are continuous. Since the NFJS problem lacks relatively complete recourse, the first-stage solution can be infeasible to the second-stage problem in that it might generate subtours, violate the re-entrant flow conditions, or create a deadlock. In the standard L-shaped method, feasibility cuts are iteratively added to the first-stage problem upon the discovery of these infeasibilities. As an alternative, we have provided certain expected-value-scenario-based inequalities to induce feasibility of the second-stage problem that greatly help reduce the effort required by the L-shaped method. To further tighten the first-stage problem formulation, we have also developed three types of valid inequalities: flow-balance constraints, re-entrant flow-based constraints, and deadlock prevention constraints. Our computational results reveal that: (a) our decomposition approach is substantially superior to the direct solution of the NFJSP using CPLEX; (b) the expected-value-scenario-based inequalities are significantly more effective than the use of standard feasibility cuts in the master problem; and (c) the judicious additional use of the re-entrant flow and deadlock prevention inequalities in conjunction with the expected-value-scenario-based inequalities further improves the overall algorithmic performance, particularly for more difficult problem instances. Furthermore, we have proposed heuristic methods for the solution relatively larger instances of NFJS and have presented results of their implementation.

Acknowledgements This work has been supported by the National Science Foundation under Grant CMMI-0856270.

References

Bagga PC (1970) N jobs, 2 machines sequencing problems with stochastic service times. Oper Res 7:184–197

Benders JF (1962) Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4:238–252

Birge JR, Louveaux F (2000) Introduction to stochastic programming. Springer, New York

Brandimarte P (1993) Routing and scheduling in a flexible job shop by tabu search. Ann Oper Res 41:157–183

Chu C, Portmann MP (1992) Some new efficient method to solve the $n \mid T \mid r_i \mid \Sigma_i w_i T_i$ problem. Eur J Oper Res 58:404–413

Cunningham AA, Dutta SK (1973) Scheduling jobs with exponentially distributed processing times on two machines of a flow shop. Nav Res Log Q 16:69–81

Dauzère-Pérès S, Paulli J (1997) An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search. Ann Oper Res 70:281–306

Elmaghraby SE, Thoney KA (1999) The two machine stochastic flowshop problem with arbitrary processing time distributions. IIE Trans 31:467–477

Foley RD, Suresh S (1984) Stochastically minimizing the makespan in flow shops. Nav Res Log Q 31:551–557

Forst FG (1983) Minimizing total expected costs in the two machine, stochastic flow shop. Oper Res Lett 2:58–61

Gascon A, Lefrancois P, Cloutier L (1998) Computer-assisted multi-item, multi-machine and multi-site scheduling in a hardwood flooring factory. Comput Ind 36:231–244

Gnoni MG, Iavagnilio R, Mossa G, Mummolo G, Di Leva A (2003) Production planning of multi-site manufacturing system by hybrid modelling: a case study from the automotive industry. Int J Prod Econ 85:251–262

Golenko-Ginzburg D, Gonik A (1997) Using "look-ahead" techniques in job-shop scheduling with random operations. Int J Prod Econ 50:13–22

Golenko-Ginzburg D, Gonik A (2002) Optimal job-shop scheduling with random operations and cost objectives. Int J Prod Econ 76:147–157

Golenko-Ginzburg D, Kesler S, Landsman Z (1995) Industrial job-shop scheduling with random operations and different priorities. Int J Prod Econ 40:185–195

Guinet A (2001) Multi-site planning: a transshipment problem. Int J Prod Econ 74:21–32

Hutchison J, Leong K, Snyder D, Ward P (1991) Scheduling approaches for random job shop flexible manufacturing systems. Int J Prod Res 29(11):1053–1067

Iwata K, Murotsu Y, Oba F, Okamura K (1980) Solution of large-scale scheduling problems for job-shop type machining systems with alternative machine tools. CIRP Ann Manuf Technol 29:335–338

Jia C (1998) Minimizing variation in a stochastic flowshop. Oper Res Lett 23:109–111

Jia HZ, Nee AYC, Fuh JYH, Zhang YF (2003) A modified genetic algorithm for distributed scheduling problems. J Intell Manuf 14(3–4):351–362

Kamburowski J (1999) Stochastically minimizing the makespan in two-machine flowshops without blocking. Eur J Oper Res 112:304–309

Kamburowski J (2000) On three machine flowshops with random job processing times. Eur J Oper Res 125:440–449

Kim Y-D (1990) A comparison of dispatching rules for job shops with multiple identical jobs and alternative routings. Int J Prod Res 28(5):953–962

Ku PS, Niu SC (1986) On Johnson's two-machine flowshop with random processing times. Oper Res 34:130–136

Kutanoglu E, Sabuncuoglu I (2001) Experimental investigation of iterative simulation-based scheduling in a dynamic and stochastic job shop. J Manuf Syst 20(4):264–279

Lai T-C, Sotskov YN, Sotskova N, Werner F (2004) Mean flow time minimization with given bounds of processing times. Eur J Oper Res 159:558–573

Lee C-Y, Chen Z-L (2001) Machine scheduling with transportation considerations. J Sched 4:3–24

Lin JT, Chen Y-Y (2006) A multi-site supply network planning problem considering variable time buckets–A TFT-LCD industry case. Int J Adv Manuf Technol 33:1031–1044

Luh PB, Chen D, Thakur LS (1999) An effective approach for job-shop scheduling with uncertain processing requirements. IEEE Trans Robot Autom 15(2):328–339

Makino T (1965) On a scheduling problem. J Oper Res Soc Jpn 8:32–44

Mastrolilli M, Gambardella LM (2000) Effective neighbourhood functions for the flexible job shop problem. J Sched 3:3–20

Miller C, Tucker A, Zemlin R (1960) Integer programming formulation of traveling salesman problems. J ACM 7:326–329

Mittal BS, Bagga PC (1977) A priority problem in sequencing with stochastic service times. Oper Res 14:19–28

Nasr N, Elsayed EA (1990) Job shop scheduling with alternative machines. Int J Prod Res 28(9):1595–1609

Pezzella F, Morganti G, Ciaschetti G (2008) A genetic algorithm for the flexible job-shop scheduling problem. Comput Oper Res 35:3202–3212

Pinedo M (1982) Minimizing the expected makespan in stochastic flow shops. Oper Res 30: 148–162

Pinedo M, Singer M (1999) A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. Nav Res Log 48:1–17

Prasad VR (1981) n $\times$ 2 flowshop sequencing problem with random processing times. Oper Res 18:1–14

Roux W, Dauzère-Pérès S, Lasserre JB (1999) Planning and scheduling in a multi-site environment. Prod Plan Control 10(1):19–28

Sarin SC, Sherali HD, Bhootra A (2005) New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. Oper Res Lett 33(1):62–70

Sauer J, Freese T, Teschke T (2000) Towards agent based multi-site scheduling. In: Proceedings of the ECAI 2000 workshop on new results in planning, scheduling, and design, pp 123–130

Sherali HD, Sarin SC, Tsai P (2006) A class of lifted path and flow-based formulations for the asymmetric traveling salesman problem with and without precedence constraints. Discret Optim 3(1):20–32

Singer M (2000) Forecasting policies for scheduling a stochastic due date job shop. Int J Prod Res 38(15):3623–3637

Subramaniam V, Lee GK, Ramesh T, Hong GS, Wong YS (2000) Machine selection rules in a dynamic job shop. Int J Adv Manuf Technol 16:902–908

Talwar TT (1967) A note on sequencing problems with uncertain job times. J Oper Res Soc Jpn 9:93–97

Tavakkoli-Moghaddam R, Jolai F, Vaziri F, Ahmed PK, Azaron A (2005) A hybrid method for solving stochastic job shop scheduling problems. Appl Math Comput 170:185–206

Timpe CH, Kallrath J (2000) Optimal planning in large multi-site production networks. Eur J Oper Res 126(2):422–435

van Slyke R, Wets RJ-B (1969) L-shaped linear programs with applications to optimal control and stochastic programming. SIAM J Appl Math 17:638–663

Varadarajan A (2006) Stochastic scheduling for a network of MEMS job shops. Ph.D. Dissertation, Grado Department of Industrial and Systems Engineering, Virginia Tech, Blacksburg, Virginia

Wang L, Zhang L, Zheng D-Z (2005) A class of hypothesis-test based genetic algorithms for flowshop scheduling with stochastic processing times. Int J Adv Technol 25(11–12):1157–1163

Weiss G (1982) Multiserver stochastic scheduling. In: Dempster M, Lenstra JK, Rinooy-Kan A (eds) Deterministic and stochastic scheduling. D. Reidel, Dordrecht, Holland

Xia W, Wu Z (2005) An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems. Comput Ind Eng 48:409–425

Yoshitomi Y (2002) A genetic algorithm approach to solving stochastic job-shop scheduling problems. Int Trans Oper Res 9:479–495