Toward Agile and Robust Supply Chains: A Lesson from Stochastic Job-Shop Scheduling

Mikhail A. Bragin  
Department of Electrical and Computer Engineering, University of Connecticut, mikhail.bragin@uconn.edu

Matthew E. Wilhelm  
Department of Chemical and Biomolecular Engineering, University of Connecticut, matthew.wilhelm@uconn.edu

Matthew D. Stuber  
Department of Chemical and Biomolecular Engineering, University of Connecticut, stuber@alum.mit.edu

Motivated by the presence of uncertainties as well as combinatorial complexity within the links of supply chains, this paper addresses the outstanding and timely challenge illustrated through a case study of stochastic job-shop scheduling problems arising within low-volume high-variety manufacturing. These problems have been classically formulated as integer linear programs (ILPs), which are known to be NP-hard, and are computationally intractable. Yet, optimal or near-optimal solutions must be obtained within strict computational time requirements. While the deterministic cases have been efficiently solved by state-of-the-art methods such as branch-and-cut (B&C), uncertainties may compromise the entire schedule thereby potentially affecting the entire supply chain downstream, thus, uncertainties must be explicitly captured to ensure the feasibility of operations. The stochastic nature of the resulting problem adds a layer of computational difficulty on top of an already intractable problem, as evidenced by the presented case studies with some cases taking hours without being able to find a “near-optimal” schedule. To efficiently solve the stochastic JSS problem, a recent Surrogate “Level-Based” Lagrangian Relaxation is used to reduce computational effort while efficiently exploiting geometric convergence potential inherent to Polyak’s step-sizing formula thereby leading to fast convergence. Computational results demonstrate that the new method is more than two orders of magnitude faster compared to B&C. Moreover, insights based on a small intuitive example are provided through simulations demonstrating an advantage of scholastic scheduling.

Key words: Supply Chain Debottlenecking; Advanced Manufacturing; Stochastic Job-Shop Scheduling; Lagrangian Relaxation; Markov Processes

1. Introduction

To stay competitive in the marketplace, the ultimate goal of any supply chain is customer satisfaction (Cheng et al. 2015, Choi 2015, Rahman et al. 2021), which primarily rests upon on-time delivery of high-quality products. The COVID-19 pandemic and ensuing lock-downs, however, caused major disruptions to the supply chain operations (Chopra et al. 2021), including insufficient resources such as a labor shortage (Nagurney 2021). In addition, safety measures preventing the spread of the virus, negatively affected the customers’ ability to obtain products of their choosing. Customer expectations have thus greatly risen, thereby forcing supply chains to swiftly readjust.
their operations toward the fulfillment of an increased number of customers’ orders at a higher pace while accounting for various uncertainties that can propagate across the entire supply chain, potentially leading to delays. As a result, a product’s value in the eyes of customers may become compromised if the product is delivered late. Therefore, agile and flexible tailored production accounts for a significant portion of the value created. The trend to improve flexibility is through the *make-to-order* production allowing the ordering of bespoke products such as machinery, automobiles, computers/electronics, or furniture (Chen et al. 2015, Rahman et al. 2017, Wang et al. 2017). After setting a due date, customers expect the order to arrive on or before the set date (Rahman et al. 2021). Therefore, the manufacturing operations must account for tardiness to avoid a tarnished image in the marketplace, and the loss of goodwill. The need to honor due dates has been widely recognized in general production-distribution operations (Leung and Chen 2013, Zhong 2015, Zhang et al. 2016, 2018, Wang 2021).

A very basic and simplified depiction of a supply chain is shown in Figure 1, illustrating the processes connecting raw materials and value-added products to the customer. An actual supply chain may be much more complicated. For example, automobile manufacturers are not equipped to manufacture every single part of the vehicle (e.g., electronic devices) and generally rely on several manufacturers as well as on logistic operations upstream. In chemical supply chains, the procurement and operations are also complicated in their structure, since raw material sources and precursor compounds are often geographically distributed and vary in purity/quality. Transportation processes are often hazardous and heavily regulated (e.g., due to explosive or toxic liquids and gases). Production processes are dynamic, involving complicated scheduling and may be responsible for producing several products. Lastly, the final products may be bulk or packaged to be delivered directly to customers on-demand (Voudouris 1996, Ferrio and Wassick 2008, Serdarasan 2013). Summarily, the above two examples are provided to demonstrate the intricate interdependency of
the supply chain links. With insufficient resources (e.g., labor shortages) and uncertainties within one or several links may become a bottleneck.

**Scope:** The purpose of this work is to pave the way for “debottlenecking” from a computational perspective. To this end, a specific type of scheduling problem—a tardiness-based job-shop scheduling (JSS) problem—will be considered to account for due dates to avoid late shipments to a greater extent possible. Within this context, job shops are a specific manufacturing environment designed for low-volume/high-variety production and are capable of producing bespoke products. The problem has been recognized as a notoriously difficult NP-hard problem because of the combinatorial complexity brought on by the presence of discrete decision variables (e.g., scheduling decisions) leading to difficulties in efficiently obtaining “optimal” or even “near-optimal” solutions. As a consequence, two possibilities can occur: 1. inefficient operations resulting from sub-optimal solutions, or 2. computational inefficiency leading to long solving times. However, “a good solution obtained late is bad.”

Within the JSS problem considered, a possibility for damaged parts/orders 1 (with some parts scrapped and with some parts reworked) is also considered, and the problem is modeled stochastically, leading to a probabilistic routing of jobs. The presence of uncertainties further complicates the JSS problem, although, scrapped orders are not specific to JSS problems only, and the implications of scrap extend well beyond manufacturing to other parts of supply chains as well. The implications of inaccurate decisions in the presence of uncertainty may be especially dire since scrapping one operation of one job may trigger remanufacturing of the entire part. This may affect and disrupt the overall schedule within the entire job shop, thereby leading to unexpectedly high delays in production. Delays are generally undesirable since they not only lead to the dissatisfaction of customers but may also potentially cause a domino effect and lead to downstream supply chain backups (i.e., bottlenecking). The intention of this work is to obtain a “good” (i.e., near-optimal) solution fast. 2 It will then be demonstrated that stochastic routing leads to lower tardiness in expectation.

The rest of this section is organized as follows. In subsection 1.1, JSS problems with the consideration of tardiness will be reviewed; in subsection 1.2, stochastic JSS will be reviewed; in subsection 1.3, recent formulation of JSS problems will be reviewed; and in subsection 1.4, a review of solution methodologies for solving JSS problems as well as for general discrete programming problems will be provided.

1 An “order” may consist of several “jobs,” while a “job” is to process a “part” such as a tractor gear; though, for the purposes of the paper, these terms are used interchangeably.

2 For the purposes of this paper, near-optimal solutions are defined as certifiably quantifiable solutions with a quality of less than an appropriate threshold, e.g. the gap of less than 5%. This threshold may heavily depend on the problem; for job-shop scheduling problems, gaps are inherently large - 10% or higher (Zhao et al. 1999).
1.1. Minimization of Tardiness: Toward On-Time Delivery Within Job Shops.

Summarily, the goal of a JSS problem is to assign a set of jobs, each consisting of a sequence of consecutive non-preemptive\(^3\) operations, to be processed on a set of machines. Generally, JSS problems have been recognized as one of the most difficult combinatorial optimization problems (Garey et al. 1976, Zhu et al. 2020).

As previously reviewed, on-time delivery has long been recognized as an important aspect of production; the minimization of tardiness has been used as an objective within JSS problems as well. The seminal works on “tardy problems” begin with the research of Sidney (1977) and Lakshminarayan et al. (1978) with subsequent research of, for example, Vepsalainen and Morton (1987), Chryssoulouris et al. (1988), Anderson and Nyirenda (1990), Du and Leung (1990), Chryssoulouris et al. (1991), Kanet and Zhou (1993), and Rohleder and Scudder (1993). The problem is known to be NP-hard (e.g., Du and Leung (1990)), even single-machine scheduling (Seidmann et al. 1981, Panwalkar et al. 1982, Ow and Morton 1989, Du and Leung 1990) belongs to a class of NP-hard problems. Yet, the job-shop scheduling problems are also subject to strict computational requirements; long solving times are not allowed since they may contribute to orders shipped late. For this reason, several rules have been developed in early implementations such as “first-come, first-serve”, “shortest processing time”, or “earliest operation due date” rules; these and several other rules are described and compared by Kanet and Zhou (1993). The minimization of tardiness, however, requires the coordination of multiple jobs and multiple machines, while dispatching rules typically ignore other resources. In fact, as pointed out by Barman and Laforge (1998), “no single priority rule performs well.” Therefore, most of the research has been heavily invested in the development of solution methodologies beyond the rule-based ones.

1.2. Robustness to Uncertainty: Stochastic Job-Shop Scheduling.

Uncertainty is a concomitant process within any production system and must be accounted for. Unforeseen events such as natural (Chopra and Sodhi 2004) as well as human-made (Thomas and Helgeson 2021) disasters have been generally recognized as one of the causes for potential disruption for supply chains thereby prompting proactive management (Chopra and Sodhi 2004). Therefore, it is important to account for potential causes of disruptions to mitigate the consequences should such unforeseen events occur in actual operations. Without the appropriate incorporation of uncertainty into problem formulations, deterministic scheduling decisions may not be optimal during operations, if feasible at all. In terms of the JSS problem, and in terms of the supply chain in general, neglecting uncertainties within the optimization may result in unexpectedly high tardiness in actual operations, thereby compromising the on-time delivery of products.

\(^3\)Each operation must be processed on a definite machine without interruption.
In the extant research, uncertain processing times have been considered (Luh et al. 1999, Golenko-Ginzburg and Gonik 2002, Lei and Xiong 2007, Lei 2011, Horng et al. 2012, Zhang et al. 2013, Yang et al. 2014, Shen and Zhu 2016, Jamili 2019, Horng and Lin 2021).

Within these papers, while the processing time is uncertain, a job is assumed to be processed with certainty. However, in actual operations, a product may be damaged (scrapped) after being processed and must be remade anew thereby potentially leading to higher tardiness. This addition of tardiness cannot be captured through stochastic modeling of processing times, since a job may require several operations, all of which may need to be repeated. Another salient feature of the above existing methods is their meta-heuristic nature (with the exception of the work by Luh et al. (1999)). Heuristic-based algorithms generally do not provide a lower bound to quantify the solution quality, with a few notable exceptions, such as (Dawande et al. 2006).

In the following subsections 1.3 and 1.4, “separable” mixed-integer linear programming (MILP) JSS formulations will be reviewed. MILP formulations are amenable for certification of solution quality through lower bounds provided by using such methods as B&C. Moreover, the separability is amenable to the efficient use of a Lagrangian Relaxation (LR). A fast resolution of JSS problems rests upon two pillars: formulation and solution methodology, both will be reviewed ahead.

1.3. “Tight” Job-Shop Scheduling ILP/MILP Formulations.

As for the formulation, if a problem is formulated in ILP forms by using too many decision variables and complicated constraints, the methods might experience difficulties (Nemhauser and Wolsey 1999). The development of “good” formulations is thus of critical importance. Another related aspect of problem formulation that has a direct impact on the solution process is the formulation “tightening.” The gist of the tightening is to delineate facets of the convex hull; if successful, then the combinatorial problem reduces to a much easier-to-solve LP problem; even if a problem is “partially” tightened, the CPU time is greatly reduced (Yan et al. 2018, 2021). One of the most recent advancements to reduce the number of decision variables and constraints, compared to previous ILP formulation, is proven to be tighter compared to previous formulations (Liu et al. 2021): only the beginning times of operations are decision variables thereby reducing the number of decision variables and constraints compared to those within the previous formulations. Since the formulation has few decision variables and constraints, for a problem instance from (Hoitomt et al. 1993), the formulation leads to several orders of magnitude improvement over previously-used approaches in terms of CPU time and to optimality of solution obtained (from 3,600 s and 3.72% gap down to 3.31 s and 0% gap) when solved by using standard B&C (Liu et al. 2021).

---

4 The exceptions are, perhaps, Hasan et al. (2011) and Lin et al. (2019) whereby machine unavailability is considered, although, both papers consider the minimization of makespan, not tardiness.
1.4. Previous Solution Methodologies.

In this subsection, the methods used to solve general scheduling problems will be briefly reviewed. Then, the methods specifically developed for the JSS problem will be reviewed. Since the methodology of this paper will be based on Lagrangian Relaxation, the LR-based methods will be reviewed as related to JSS problems. Finally, the most recent advancements of LR will be presented.

**Methodologies for General Scheduling Problems.** To solve scheduling problems (including JSS problems minimizing makespan, and flow shop scheduling), the following methods have been used: taboo search (Taillard 1994), local search (Aarts et al. 1994, Vaessens et al. 1996, Beck et al. 2011), symbiotic evolutionary algorithm (Kim et al. 2003), heuristics (Dawande et al. 2006), ant colony optimization (Leung et al. 2010), constraint programming (Beck et al. 2011, Malapert et al. 2012), conflict-directed search (Grimes and Hebrard 2015), a hybrid of genetic and tabu search algorithms (Li and Gao 2016), Lagrangian Relaxation (Iloglu and Albert 2018), particle swarm optimization (Nouiri et al. 2018), hybrid genetic algorithm and variable neighborhood search (Li et al. 2018), and genetic programming-based evolutionary approach (Zhu et al. 2020). The area of the scheduling is very rich and quickly evolving, and the above list is by no means inclusive.

**Methodologies for Job-Shop Scheduling Problems with the Consideration of Tardiness.** Because of the complexity, JSS problems require a more sophisticated methodology beyond the simple rules reviewed in subsection 1.1. Toward this goal, several methods have been developed such as Lagrangian Relaxation (Hoitomt et al. 1993, Luh et al. 1999), genetic algorithms (Herrmann et al. 1995, Lei 2011), heuristics (Golenko-Ginzburg and Gonik 2002), neural networks with simulated annealing (Tavakkoli-Moghaddam et al. 2005), evolutionary algorithms (Lei and Xiong 2007, Horng et al. 2012, Zhang et al. 2013), ordinal optimization (Yang et al. 2014), and evolutionary learning-based simulation optimization (Ghasemi et al. 2021).

**Lagrangian Relaxation for Job-Shop Scheduling Problems.** To deal with the combinatorial complexity described above, Lagrangian Relaxation has been a successful method to “reverse” the combinatorial complexity, thereby exponentially reducing the effort required to solve sub-problems. In terms of JSS problems, after relaxing machine capacity constraints, sub-problems associated with individual jobs can be solved with significantly reduced complexity.\(^5\) To the best of the authors’ knowledge, the application of standard Lagrangian Relaxation starts with the work of Hoitomt et al. (1993) whereby Lagrange multipliers are updated by using the sub-gradient method. Supported by the work “Surrogate” Lagrangian Relaxation (SLR) (Bragin et al. 2015) whereby the difficulties of the sub-gradient method (such as zigzagging of multipliers and high computational effort) were

---

\(^5\) Relaxation of other constraints other than the constraints that couple subsystems, such as jobs in job-shop scheduling problem, is not necessary since the decomposition into job sub-problems already leads to a drastic reduction of complexity while relaxing more constraints would lead to a larger number of multipliers to adjust and slow convergence.
overcome, the JSS problem was formulated as an MILP problem and efficiently solved by using the SLR method (Yan et al. 2018). To further improve computational efficiency, supported by the work of Bragin et al. (2019) whereby convergence of SLR was improved by using two-segment absolute-value penalties for constraint violations, the JSS problem (as mentioned before) was formulated as a tighter MILP problem and the resulting problem was solved by using a version of the SLR method with three-segment penalties for faster convergence (Liu et al. 2021).

Lagrangian Relaxation-based methods have been instrumental in achieving success in solving difficult combinatorial problems and several relevant methods will be briefly reviewed next.

**Surrogate Sub-Gradient Method.** Within the Surrogate Sub-Gradient Method (Zhao et al. 1999), multipliers are updated after solving one sub-problem at a time rather than solving all the sub-problems as within the standard sub-gradient method. This significantly reduces the computational effort. The step-sizes are updated by using the following variation of the Polyak’s formula Polyak (1969):

\[
0 < s^k < \gamma \cdot \frac{q(\lambda^*) - L_k}{\|\tilde{g}^k\|_2^2}, \gamma < 1,
\]

where \(L_k\) is the Lagrangian value and \(\tilde{g}^k\) are the levels of constraint violations (also referred to as “surrrogate” sub-gradients). These values are used in place of dual values \(q(\lambda^k)\) and sub-gradients \(g^k\). The convergence to \(\lambda^*\) is guaranteed (Zhao et al. 1999). The concomitant reduction of multiplier zigzagging has been also observed. Even through the geometric convergence rate is expected, e.g., multipliers strictly move closer to \(\lambda^*\), the implementation of the method is problematic because of the unavailability of the knowledge about the optimal dual value \(q(\lambda^*)\).

**Surrogate “Level-Based” Lagrangian Relaxation.** To overcome the above issue, the Surrogate “Level-Based” Lagrangian Relaxation (SLBLR) method has been recently developed by Bragin (2022) with adaptive adjustment of the “level” estimate of the optimal dual value within the step-sizing formula (1) to exploit the geometric convergence potential of the Polyak’s step-sizing. The main idea is to obtain “level values” - overestimates of the optimal dual value to set step-sizes per (1). This has been achieved in (Bragin 2022, p. 12, Corollary 2):

**Corollary 1.** If the following auxiliary “multiplier-convergence-feasibility” problem (with \(\lambda\) being a continuous decision variable; \(\lambda \in \mathbb{R}^m\))

\[
\begin{align*}
    \|\lambda - \lambda^{k_j+2}\|_2 & \leq \|\lambda - \lambda^{k_j+1}\|_2, \\
    \|\lambda - \lambda^{k_j+4}\|_2 & \leq \|\lambda - \lambda^{k_j+3}\|_2, \\
    \ldots \\
    \|\lambda - \lambda^{k_j+n_j}\|_2 & \leq \|\lambda - \lambda^{k_j+n_j-1}\|_2,
\end{align*}
\]

\(6\) Unlike that in Polyak’s formula (Polyak 1969), parameter \(\gamma\) is less than 1 rather than 2.
admits no feasible solution with respect to $\lambda$ for some $k_j$ and $n_j$, then $\exists i \in [k_j, k_j + n_j]$ such that

$$s^i \geq \gamma \cdot \frac{q(\lambda^*) - L^k}{\|\tilde{g}^k\|_2^2}. \quad (3)$$

In simple terms, if (2) is infeasible, then multipliers do not approach any common limit, including $\lambda^*$, which is only possible if stepsizes violate the Polyak’s step-sizing formula (1). In a sense, the Corollary helps detect when step-sizes are “too large” thereby triggering the need for their subsequent reduction. Firstly, from (3) it follows that there exists an overestimate of $q(\lambda^*)$, and the overestimate (the “level value”) has been derived to be:

$$\bar{q}_j = \max_{i \in [k_j, k_j + n_j]} \left( \frac{1}{\gamma} \cdot s^i \cdot \|\tilde{g}^k\|_2^2 + L^k \right) > q(\lambda^*). \quad (4)$$

As a result, the step-sizes are set by using a series of decreasing (not necessarily monotonically) overestimates $\{\bar{q}_k\}$ as:

$$s^k = \zeta \cdot \gamma \cdot \frac{\bar{q}_k - L^k}{\|\tilde{g}^k\|_2^2}, \gamma < 1, \zeta < 1. \quad (5)$$

The factor of $\gamma$ in the above formula is inherited from the formula (1), and the extra factor of $\zeta$ is introduced to counter the fact that the stepsizes are set by using overestimates of the optimal dual value $q(\lambda^*)$. The method was able to obtain globally optimal solutions for the largest instances of the Generalized Assignment Problem (Bragin 2022).

The rest of the paper is organized as follows. After presenting the stochastic JSS problem formulation in Section 2, the SLBLR method will be used to solve the problem in Section 3. The numerical case studies and solution results are presented in Section 4 that demonstrate the advantages of the SLBLR method. The conclusion will be provided in Section 5.

2. Job-Shop Scheduling Problem Description and Formulation

Within a typical job shop, machines are grouped by their functionality (e.g., milling, lathing, drilling, etc.), and the number of such groups is denoted by $M$. Within each group, the number of machines, referred to as the “capacity” is denoted as $C_m$, where $m = 1, \ldots, M$. For every order $i = 1, \ldots, I$ (e.g., a part to be processed) arriving at a job shop with priority $w_i$, a specific sequence of $J_i$ operations, processing times $p_{i,j,m}$ (for each part $i$, operation $j^7$ and machine type $m$), as well as an integer-valued due date $d_i$ is specified. Processing times are assumed to be integer and each operation can occupy several contiguous time blocks, that is, every operation is assumed to be non-preemptive. Accordingly, the time horizon is assumed to consist of $T$ discrete equidistant time blocks. A part $i$ may need to go through a sequence of operations, each is eligible to be performed
by one or a few machine groups. Let $E_{i,j}$ be a group of machines eligible to process operation $(i, j)$, and $O_m$ be a set of operations that can be processed on machine $m$.

Figure 2 illustrates possible, albeit simplified, job flow within a job shop omitting the time aspect. As shown in Figure 2, Job 1 requires 3 operations to be performed by Machine Groups 1, 6, and 3, sequentially. Likewise, Job 2 requires 3 operations to be performed by Machine Groups 2, 3, and 5, sequentially; and Job 3 requires 4 operations to be performed by Machine Groups 1, 2, 5, and 4, sequentially. If Machine Group 1 contains only one machine available and Jobs 1 and 3 arrive at the same time, only one job can be processed during the next time period — this is to be decided through optimization. A similar argument holds for other machines groups, except Machine Groups 4 and 6, which only operates on Job 3 and Job 1, respectively. In terms of the job shop shown in Figure 2, $E_{1,1} = \{1\}$, $E_{1,2} = \{6\}$, etc., and $O_1 = \{(1,1), (3,1)\}, O_2 = \{(2,1), (3,2)\}$, etc.

There is a nonzero probability of scrap—with probability $p^s_{(i,j)}$ a part is scrapped after operation $j$ of the part $i$ and its production must start from Operation 1. There is also a nonzero probability of rework—with probability $p^r_{(i,j)}$ a scrapped part $i$ can be recovered and its production must start from an operation $j$. Transitions among the states of a part follow a Markov process schematically

---

7 Operation $j$ will be denoted as $(i, j)$ whenever appropriate to indicate specific part $i$ requiring operation $j$. 
illustrated in Figure 3. For simplicity of exposition, probabilities of scrap and rework are chosen to be 20% and 50%, respectively. Transitions from one operation to the next thus happen with a probability of 80%, and in a case of a defective part, with equal probabilities of 10%, a part will either be reworked or completely discarded, triggering the restart of processing from Operation 1.

![Figure 3](image)

**Figure 3** A Markov process is illustrated and used in this work to capture transitions among states (operations) of a part. In this illustration, with a probability of 10% a part is either scrapped or reworked; at operation 1, scrap and rework are lumped together since a defective part needs to begin getting processed at operation 1.

In this paper, the following assumptions are made:

1. Each machine can process at most one operation at a time;
2. Preemption is not allowed, so no operation can be interrupted during processing;
3. No two operations of a given job can be processed at the same time, and moreover, the predefined sequence of the operations should be kept: no two operations can be swapped;
4. All jobs and machines are available at Time 1;
5. All the non-processing related times such as setup times and transportation times, are included into the processing time.

The job-shop scheduler’s task is to *proactively* assign each operation of each order to specifically designated machines in anticipation of scrap and rework while minimizing the overall *expected* tardiness. Accordingly, the overall layout of the *proactive* scheduling consists of two attempts — if scrap happens on a first attempt, the second attempt is initiated. These two attempts will be captured probabilistically and will be modeled within one problem formulation. The formulation developed by *Liu et al. (2021)* will be used as a basis, and stochastic elements due to scrap and
rework will be added, as explained in the following sections. Once a feasible schedule is obtained, the schedule corresponding to the first attempt is generated at the beginning of a shift: which job is to be assigned to what machine group and at what time. The integer-valued beginning and completion times for all operations within both attempts ((1) and (2)) will be collectively denoted as vectors \( b = \{(b^1), (b^2)\} \) and \( c = \{(c^1), (c^2)\} \). After a schedule is obtained, a sequence of operations to be processed by each machine group is inferred from beginning times of the “first attempt” \( b^1 \) and the resulting sequence is passed on to the job-shop floor. After each shift, if there is no scrap, job-shop operations follow the aforementioned sequence. Otherwise, a rescheduling is triggered and a new sequence of operations is inferred. The short solving times and high-quality schedules are thus especially important in the presence of uncertainties.

2.1. Objective Function

To avoid the delays mentioned in the previous section, the on-time shipment of processed orders/jobs is of paramount importance. Accordingly, weighted tardiness is minimized. The objective is on-time delivery; accordingly, the expected total weighted tardiness is minimized:

\[
o(c) = \sum_{i=1}^{l} w_i \cdot \left\{ \prod_{j=1}^{J_i} (1 - p^s_{i,j}) \right\} \cdot \max(c^1_{i,J_i} - d_i, 0) +
\]

\[
\sum_{i=1}^{l} w_i \cdot \sum_{j'=1}^{J_i} \left\{ \sum_{j=1}^{j'-1} \left( \prod_{j=1}^{j'} (1 - p^s_{i,j}) - \prod_{j=1}^{j'} (1 - p^s_{i,j}) \right) \cdot (1 - p^r_{i,j'}) \cdot \max(c^2_{i,J_i,j'} - d_i, 0) \right\} +
\]

\[
\sum_{i=1}^{l} w_i \cdot \sum_{j'=1}^{J_i} \left\{ \sum_{j=1}^{j'-1} \left( \prod_{j=1}^{j'} (1 - p^s_{i,j}) - \prod_{j=1}^{j'} (1 - p^s_{i,j}) \right) \cdot (p^r_{i,j'}) \cdot \max(c^2_{i,J_i,j'} - d_i, 0) \right\}.
\]

Here, \( c = \{(c^1_{i,j}), (c^2_{i,J_i,j'})\} \) collectively denotes an array of completion times within first (superscript 1) and second (superscript 2) attempts. The first term of the expectation involves probabilities \( \prod_{j=1}^{J_i} (1 - p^s_{i,j}) \) that a part is completed on a first attempt with a completion time of \( c^1_{i,J_i} \). For each part \( i \), scrap after each operation \( j \) leads to \( J_i \) scenarios, each corresponding to processing a part from scratch with certain probabilities, and the expectation is taken with respect to these second-attempt scenarios. Accordingly, the second and the third terms in the expectation involve the probability \( \prod_{j=1}^{j'-1} (1 - p^s_{i,j}) - \prod_{j=1}^{j'} (1 - p^s_{i,j}) \) that a part is scrapped at operation \( j' \).\(^8\) \(^9\) Within the second term, \( (1 - p^r_{i,j'}) \) is the probability that a part cannot be reworked and the associated

---

\(^8\) Consider a small example involving 3 operations and the probability of scrap of 20%. Within the first term, the probability that a part will be completed on a first attempt is 0.8. The probability that a part is scrapped after a first operation is 0.8, the probability that a part is scrapped after a second operation is 0.8, and the probability that a part is scrapped after the third operation is 0.8 - 0.8 = 0.16. These probabilities add up to one. To avoid excessive complexity, the probability that a part is scrapped for the second time is assumed to be zero, although, if a part is scrapped, the scheduling problem is to be solved again at the beginning of the next shift.

\(^9\) Note that if \( j' = 1 \), there are no multiplicands within the product \( \prod_{j=1}^{j'-1} (1 - p^s_{i,j}) \) and it is defined to be equal to 1.
completion time is $c_{i,j}^2$ with zero ($r = 0$) in the subscript indicating on “off” status for the rework. Similarly, within the third term, $p_{(i,j')}^r$ is the probability that a part can be reworked and the associated completion time is $c_{i,j'}^2$ with one ($r = 1$) in the subscript indicating an “on” status for the rework. The max operators indicate that there is no penalty for parts finished before the due date $d_i$. The linearization of these operators follows the standard special ordered set procedure employed by Yan et al. (2018, 2021). Despite the complex appearance of the objective function within (6), the important salient feature of it is additivity in terms of jobs $i: o(c) = \sum_{i=1}^t o(c_i)$. This feature will be efficiently exploited in Section 3.

2.2. Constraints

To ensure the feasibility of the schedule, processing-time-requirement constraints (how long an operation needs to be processed), operation precedence constraints (to keep the order of operations as specified within a job), and machine capacity constraints (the number of operations to be processed on machines of a certain type cannot exceed the number of machines) are imposed at a modeling stage.

**Beginning and Completion Time Constraints:** During Attempt 1, the completion time $c_{i,j}^1$ of operation $(i,j)$ is expressed as:

$$c_{i,j}^1 = \sum_{m \in E_{i,j}} \sum_{t=1}^T (t + p_{i,j,m}) \cdot x_{i,j,m,t}^1 - 1, \forall (i,j),$$

(7)

where $x_{i,j,m,t}^1$ is an indicator: $x_{i,j,m,t}^1 = 1$, if an operation $(i,j)$ is to begin at time $t$ \(^{11}\) on a machine group $m$, and 0, otherwise; the above indicator variable can take the value of 1 for only one combination of $\{i,j,m,t\}$, therefore, the following constraint is imposed to satisfy Assumption 1 and 3:

$$\sum_{m \in E_{i,j}} \sum_{t=1}^T x_{i,j,m,t}^1 = 1, \forall (i,j).$$

(8)

Therefore, all the terms within (7) with the exception of the $t^{th}$ term are zero, and the part is completed at a time $t + p_{i,j,m}$. Accordingly, during Attempt 1, the beginning time $b_{i,j}^1$ of operation $(i,j)$ is expressed as:

$$b_{i,j}^1 = \sum_{m \in E_{i,j}} \sum_{t=1}^T t \cdot x_{i,j,m,t}^1, \forall (i,j).$$

(9)

\(^{10}\) Here one unit of time is subtracted because of the discrete nature of the time. For example, if the beginning time is 4 (e.g., at the top of the hour at 4:00 pm) and the processing time is 2, then the completion time is 5 (e.g., at the close of the hour at 5:59:59 pm). This way, if the next operation needs to start after the completion of the previous one, the beginning of the operation will be at time 6; in this scenario, 2 units of time are between the beginning times of subsequent operations.

\(^{11}\) Here $t$ is used to distinguish a specific beginning time from $t$, which is a dummy variable within summations.
The completion time within the second attempt is modeled in a similar way:

\[ c_{i,j,j',r}^2 = \sum_{m \in E_{i,j}} \sum_{t=1}^{T} (t + p_{i,j,m}) \cdot x_{i,j,j',m,t,r}^2 - 1, \forall (i,j,j',r). \] (10)

The additional subscripts \( j' \) and \( r \) indicate that the completion time is computed for scenarios whereby a part is scrapped after operation \( j' \) within the first attempt, and then the part is either reworked \((r = 1)\) or completely discarded \((r = 0)\). Analogously to (8) and (9), the beginning times of the second attempt are formulated as follows:

\[ \sum_{m \in E_{i,j}} \sum_{t=1}^{T} x_{i,j,j',m,t,r}^2 = 1, \forall (i,j,j',r), \] (11)

\[ b_{i,j,j',r}^2 = \sum_{m \in E_{i,j}} \sum_{t=1}^{T} t \cdot x_{i,j,j',m,t,r}^2, \forall (i,j,j',r). \] (12)

**Operation Precedence**: Within each attempt, operation precedence constraints ensure that subsequent operation \( j + 1 \) can only begin after the previous operation \( j \) is completed:

\[ b_{i,j+1}^1 \geq c_{i,j+1}^1 + 1, \quad b_{i,j+1,j',r}^2 \geq c_{i,j,j',r}^2 + 1, \forall (i,j,j',r). \] (13)

**Scrap/Rework Constraints**: Scrap/rework constraints ensure that the second attempt begins at Operation 1 but only after operation \((i,j)\) leads to scrap \((r = 0)\) of part \(i\):

\[ b_{i,1,j',0}^2 \geq c_{i,j'}^1 + 1, \forall (i,j'). \] (14)

If a part can be reworked \((r = 1)\), then the operation \( j \) (second subscript) is repeated:

\[ b_{i,j,j',1}^2 \geq c_{i,j'}^1 + 1, \forall (i,j'). \] (15)

**Beginning of Shift Constraints**: It is a common practice to reschedule operations after the end of a shift to generate the schedule for the upcoming shifts. After part \(i\) is scrapped after operation \(j'\), the second attempt needs to be initiated at or after the beginning of future shifts. This is captured through the following constraints:

\[ S \cdot \left\lceil \frac{c_{i,j'}^1}{S} \right\rceil + 1 \leq b_{i,j',j',0}^2, \quad S \cdot \left\lfloor \frac{c_{i,j'}^1}{S} \right\rfloor + 1 \leq b_{i,1,j',1}^2, \forall (i,j'), \] (16)

where \(S\) is the length of a shift. The ceiling operator is linearized after introducing integer variables \( y_{i,j'} \) as:

\[ \frac{c_{i,j'}^1}{S} \leq y_{i,j'} \leq \frac{c_{i,j'}^1}{S} + 1 - \varepsilon, \] (17)

where \(\varepsilon\) is a small positive number.
Expected Machine Capacity Constraints: Machine capacity constraints ensure that at any point in time, the expected number of operations processed does not exceed the total number of machines within a certain group eligible to perform the corresponding operation:

\[
\sum_{(i,j) \in O_m} \prod_{j'=1}^{j-1} (1 - p'_{i,j'}) \cdot \left\{ \sum_{t'=t-p_{i,j,m,1}, t' \geq 1}^{t} x^1_{i,j,m,t'} \right\} + \\
\sum_{j', (i,j) \in O_m} \left( \prod_{j=1}^{j'-1} (1 - p'_{i,j'}) - \prod_{j=1}^{j'} (1 - p'_{i,j'}) \right) \cdot (1 - p'_{i,j'}) \cdot \left\{ \sum_{t'=t-p_{i,j,m,1}, t' \geq 1}^{t} x^2_{i,j',j',m,t',0} \right\} + \\
\sum_{j', (i,j) \in O_m} \left( \prod_{j=1}^{j'-1} (1 - p'_{i,j'}) - \prod_{j=1}^{j'} (1 - p'_{i,j'}) \right) \cdot (p'_{i,j'}) \cdot \left\{ \sum_{t'=t-p_{i,j,m,1}, t' \geq 1}^{t} x^2_{i,j',j',m,t',1} \right\} \leq C_m, \forall (m,t).
\]

Equation (18) ensures that at any time \( t \) and for any machine group \( m \), the expected number of part processed does not exceed the number of machines \( C_m \), and the expectation within (18) follows the similar logic as within the objective function (7). The exception is the first term, thereby the upper limit of the summation is \( j - 1 \). For example, the probability of a part surviving up to the last operation is \( \prod_{j=1}^{j'-1} (1 - p'_{i,j'}) \), while the probability of a part surviving after the last operation is complete is \( \prod_{j=1}^{j'} (1 - p'_{i,j'}) \), which is appropriate for calculating the tardiness as in (6).

For further convenience and compactness of notation, the objective function will be denoted as \( o(c) = \sum_{i=1}^{I} a_i(c_i) \) due to the additivity of (6) and the optimization problem becomes:

\[
\min_{b,c,x} \left\{ \sum_{i=1}^{I} a_i(c_i), \text{ s.t. } (6) - (18) \right\}.
\]

The constraints (18) are additive in terms of parts \( i \) as well, are converted to equality constraints through the use of nonnegative slack decision variables \( z_{m,t} \), and are expressed as:

\[
\sum_{i=1}^{I} g_i(x_i) + z_{m,t} - C_m = 0.
\]

Here, \( \sum_{i=1}^{I} g_i(x_i) \) compactly denotes the left-hand side of (18) with \( x_i \) denoting a vector that contains all decisions with respect to operations and attempts \( x_i = \{ x^1_{i,j,m,t}, x^2_{i,j',j',m,t,t',1} \} \). While all the variables within (6)-(18) are integer, because of the probabilities involved within (18), the slack variables are modeled as continuous variables. In the following section, the solution methodology is presented.

3. Solution Methodology

The success of Lagrangian Relaxation relies on the optimization of the nonconvex dual function, which, in a general form can be written as:

\[
\max_{\lambda} \{ q_{\rho}(\lambda) : \lambda \in \mathbb{R}^M \times \mathbb{R}^T \},
\]

\[(21)\]
with
\[
q_\rho(\lambda) = \min_{b,c,x,z} \left\{ L_\rho(c,x,z,\lambda), (b_i, c_i, x_i) \in \mathcal{F}_i, i = 1, \ldots, I, z \geq 0 \right\}, \tag{22}
\]

where \( L_\rho(c,x,z,\lambda) \equiv \sum_{i=1}^I o_i(c_i) + \lambda^T \left( \sum_{i=1}^I g_i(x_i) + z_{m,t} - C_m \right) + \rho \cdot \left\| \sum_{i=1}^I g_i(x_i) + z_{m,t} - C_m \right\|_1 \), is the “absolute-value” Lagrangian function (Bragin et al. 2019). The feasible set \( \mathcal{F}_i \) for each job \( i \) is delineated by constraints (7)-(15). The Lagrange multipliers \( \lambda \) (“dual” variables) are the decision variables with respect to the dual problem (20) and it is assumed that the set of optimal solutions \( \Omega = \{ \lambda \in \mathbb{R}^M \times \mathbb{R}^T \mid q(\lambda) = q(\lambda^*) \} \) is not empty. The minimization within (22) with respect to \( \{b,c,x,z\} \) is referred to as the “relaxed problem.” Due to the presence of integer variables \( \{b,c,x,z\} \), the dual function (20) is non-smooth with facets (each corresponding to a particular solution to the relaxed problem within (21)) intersecting at ridges whereby derivatives of \( q(\lambda) \) exhibit discontinuities; in particular, the dual function is not differentiable at \( \lambda^* \). As a result, sub-gradients may lead to lower dual values or exhibit zigzagging across ridges of the dual function.

To maximize the dual function, the recent “Surrogate Level-Based Lagrangian Relaxation” approach (Bragin 2022) is chosen. The dual function (20) is maximized by updating Lagrange multipliers \( \lambda \) by taking a series of steps \( s^k \) along “surrogate” sub-gradient directions \( \sum_{i=1}^I g_i(x_i^k) + \tilde{z}_{m,t}^k - C_m \) (levels of constraint violation) as
\[
\lambda^{k+1} = \left[ \lambda^k + s^k \cdot \left( \sum_{i=1}^I g_i(x_i^k) + \tilde{z}_{m,t}^k - C_m \right) \right]^+, \tag{23}
\]

where \([\cdot]^+\) is a projection operator onto a positive orthant \( \{\lambda \mid \lambda \geq 0\} \). Following (5), the step-sizes are set as:
\[
s^k = \zeta \cdot \gamma \cdot \frac{\bar{q}_k - L(\tilde{b}^k, \lambda^k)}{\left\| \sum_{i=1}^I g_i(\tilde{b}_i) + \tilde{z}_{m,t}^k - M_m \right\|_2}, \gamma < 1, \zeta < 1. \tag{24}
\]

Here the “tilde” is used to distinguish solutions \( \{\tilde{b}^k, x^k, z^k\} \) that are obtained by solving optimally the relaxed problem (minimization within (22)) from sub-problem solution \( \{\tilde{b}^k, x^k, z^k\} \). Sub-problems are formulated by forming a group of parts \( \mathcal{I} \) and optimizing the relaxed problem with respect to the associated variables while keeping decision variables not belonging to a subset \( \mathcal{I} \) fixed as:
\[
\min_{b_i, c_i, x_i, z_i \in \mathcal{I}} \left\{ \sum_{i \in \mathcal{I}} o_i(c_i) + \lambda^T \left( \sum_{i \in \mathcal{I}} g_i(x_i) + z_{m,t} \right) + \rho \cdot \left\| \sum_{i \in \mathcal{I}} g_i(x_i) + \sum_{i \notin \mathcal{I}} g_i(x_i^{k-1}) + z_{m,t} - C_m \right\|_1 \right\}. \tag{25}
\]

The above minimization involves piecewise linear penalties (\( l_1 \) norms), that efficiently penalize constraint violations and are exactly linearizable through the use of special ordered sets, thereby enabling the use of MILP solvers.
Heuristics. Heuristics are generally the last, yet, a necessary step required to obtain feasible solutions. Feasible solutions are obtained by repairing sub-problem solutions and the feasible solution search is initiated after the norm of constraint violations is “sufficiently small.” For the JSS problem, the following simple rule is used: adjust sub-problem beginning times of operations $(i,j)$ by no more than $\delta_j$ time periods. This rule is operationalized as follows:

$$\min_{b,c,x,z} (6), \text{ s.t. } (7) - (18), -\delta_j \leq b_{i,j} - b_{k,i,j} \leq \delta_j. \tag{26}$$

The above problem is then solved by an MILP solver. The steps of the SLBLR method are summarized in Algorithm 1.

**Algorithm 1. Surrogate “Level-Based” Lagrangian Relaxation**

**Step 1: Initialization.** Initialize $\lambda^0$, $s^0$, $c^0$.

**Step 2: Sub-problem Solving.** Solve a sub-problem (25).

**Step 3: Stepsize Update.** Use (24) to update stepsizes.

**Step 4: Multiplier Update.** Use (23) to update multipliers.

**Step 5: Penalty Coefficient Update.** Update $\rho^{k+1} = \rho^k + \beta, \beta > 0$.\(^{12}\)

**Step 6: Constraint Violation Check.** Check the Levels of Constraint Violation. If $\left|\sum_{i=1}^{I} g_i(x_i) + z_{m,t} - C_m\right| \leq \epsilon$, go to Step 7, otherwise, go to Step 2.

**Step 7: Feasible Solution Search.** Solve (26). If the CPU time limit is not reached, go to 2, otherwise, terminate with the best feasible solution obtained.

The efficiency of the overall approach will be discussed with respect to numerical case studies in the following section.

4. Numerical Testing Results

The solution methodology was implemented in an open-source Julia package, Jobshop.jl available at [https://github.com/PSORLab/Jobshop.jl](https://github.com/PSORLab/Jobshop.jl) (Wilhelm et al. 2022). Each subproblem in the numerical examples were solved using 64-bit CPLEX 12.10.0 optimizer and tested using a CPU (48 threads) of Intel(R) Xeon(R) E-2286M CPU @ 2.99 Hz, 192 GB of RAM, and Windows 10.

Example 1 is to demonstrate the computational efficiency of the method, it is an instance with 20 jobs, 5 operations per job, and 5 machines, each specializing in a specific operation. Example 2 is small enough (4 jobs, 4 operations, 4 machines) to intuitively provide insights into why stochastic scheduling leads to smaller tardiness in simulations compared to deterministic scheduling.\(^{13}\)

---

\(^{12}\)When solving continuous problems, penalty coefficients are typically updated by using a multiplicative constant $\beta$ as $\rho^{k+1} = \rho^k \cdot \beta, \beta > 1$, for example, within the Method of Multipliers (Bertsekas 2016). However, an additive constant $\beta$ is adopted here.

\(^{13}\)This paper is on the optimization aspect of stochastic scheduling; full-fledged simulations are out of scope. Nevertheless, we believe that the optimization developed opens the way to efficiently optimize stochastic JSS thereby enabling simulations in future work as well.
4.1. Example 1. Small Instance with 20 Jobs.

Within this example, an instance with 20 jobs, 5 operations per job, and 5 machines, each designated to perform one specific operation is considered. Processing times are generated randomly. The data used in this example are shown in the Table A1 in the Appendix.

The due dates are \( d = \{15, 25, 32, 36, 21, 27, 26, 13, 29, 12, 35, 31, 19, 24, 33, 23, 18, 21, 17, 22\} \). The machine capacities are \( M = \{2, 3, 2, 2, 3\} \). The scrap rate is 5% and the rework rate is 20%. The priorities are all chosen to be equal \( w_i = 1, i = 1, \ldots, I \). The results of the SLBLR method and standard B&C are shown in Figure 4. As illustrated in Figure 4, the SLBLR method is more than two orders of magnitude faster than standard B&C.

![Figure 4](image-url)

Figure 4: The results for Example 1 are illustrated. The SLBLR method performs approximately 250 times faster than the B&C algorithm.

4.2. Example 2. Simulations Based on a Small Problem Instance.

The purpose of this example is to provide intuitive insights into the advantages of the stochastic schedule based on an example small enough to visualize the scheduling decisions. For this example, only scrap is considered with the rate of 10% without considering rework for simplicity of exposition.
(rework rate = 0%). For this purpose, an instance with 4 jobs with 4 operations each is generated and the data are shown in Table 1. Machine capacities are: \( M = \{3, 2, 1, 2\} \).

| \( i \) | \( j \) | \( p_{i,j} \) | \( m \) | \( i \) | \( j \) | \( p_{i,j} \) | \( m \) |
|-------|-------|--------|-----|-------|-------|--------|-----|
| 1     | 1     | 2      | 1   | 3     | 1     | 2      | 1   |
| 1     | 2     | 5      | 2   | 3     | 2     | 4      | 2   |
| 1     | 3     | 4      | 3   | 3     | 3     | 6      | 3   |
| 1     | 4     | 3      | 4   | 3     | 4     | 3      | 4   |
| 2     | 1     | 2      | 1   | 4     | 1     | 5      | 1   |
| 2     | 2     | 5      | 2   | 4     | 2     | 4      | 2   |
| 2     | 3     | 4      | 3   | 4     | 3     | 4      | 3   |
| 2     | 4     | 4      | 4   | 4     | 3     | 4      | 4   |

For comparison, a deterministic schedule was also generated (scrap rate = 0%). In the following Figure 5 two schedules (the first is based on the stochastic scheduling, and the second is based on the deterministic scheduling) are shown. To visualize times slots for part scheduling as well as due dates the usual Gantt chart is shown in Figure 5. Scheduling of the first attempt is shown for stochastic scheduling. Within the first attempt, four operations are denoted by using the format \((i, j)\), for example, \((1,1)\) means Part 1, Operation 1, and so on. For enhanced visualization, operations are color-coded: Operation 1 is yellow, Operation 2 is green, Operation 3 is brown, and Operation 4 is black. In a black-and-white version of the paper, the operations are color-coded using grayscale: the lightest gray corresponds to the first operation, while the darkest (black) corresponds to the last operation. Due dates are shown by inverted triangles. For example, for Part 1 of the stochastic scheduling, the due date is 15. Since operation \((1,3)\) is scheduled to begin at time 17 and operation \((1,4)\) is scheduled to begin at time 21 and complete at 23, the tardiness for Part 1 is 8. The tardiness of other parts is similarly computed.

As shown in Figure 5, the stochastic scheduling leads to the total tardiness of 16 (within the first attempt), the same as within the deterministic scheduling. The purpose of this example is to provide several insights into why the stochastic schedule leads to lower expected tardiness in simulations.\(^{14}\) For this purpose, one specific case will be analyzed - scrap happens once after Operation 2.

Since there are 4 parts, there are 4 scenarios in which parts can be scrapped. The resulting scenarios based on the stochastic scheduling are illustrated in Figure 6. The deterministic counterpart is shown in Figure 7.

\(^{14}\)While the example is rather small, the analysis of all the possible scenarios is complicated; while there are 16 operations in total, the number of possible combinations of operations after which scrap can happen is large (e.g., the number of possibilities in which 2 parts can be scrapped is \(4C_2 \cdot 2^4 = 96\)). The full-fledged simulations are deferred to future work.
Figure 5  Schedules are obtained by solving stochastic and deterministic problems. Schedules, as well as due dates, are shown for each part.

![Figure 5](image)

Figure 6  The stochastic scheduling results for Example 3: the schedule for Shift 1 is the same as that within Figure 5; because scrap is assumed to occur after operation 2 of each part, the rescheduling is triggered at the beginning of Shift 2 for parts 1, 3 and 4, and at the beginning of Shift 3 for Part 2.

![Figure 6](image)

Within the stochastic scheduling, only one of the Operation 2: (2,2) is completed during Shift 2 (Scenario 2 in Figure 6), while within the deterministic scheduling two operations 2: (2,3) and (2,4) (Scenario 3 and 4 in Figure 7) are completed during shift 2. Since the rescheduling of these operations is performed at the beginning of Shift 3, the impact of Scenarios 3 and 4 (Figure 7) on the total tardiness is more pronounced. As a result, the expected tardiness (assuming that scrap happens one after the second operation) within the stochastic case is 21.4, while in the deterministic case, the expected tardiness is 24. Since in the scrap-free situation, the tardiness is 16 for both cases, the excess tardiness introduced by scrap is 5.4 and 8, respectively.
5. Conclusion

The major contributions and takeaways of this study are the following:

1. The unavoidable defects in manufacturing production are captured probabilistically by using Markov processes, the optimization problem to minimize expected tardiness is formulated, and the resulting problems are solved proactively in anticipation of scrap and rework.

2. The solution methodology is computationally efficient allowing to solve quickly the associated discrete optimization problem, which has the following major implications:

   (a) The solution methodology is based on decomposition and coordination principles. Therefore, besides being capable of drastically reducing the complexity of the problem, the method is easily adaptable for the handling of unexpected/urgent orders. In the presence of such orders, which could have otherwise affected the entire schedule, or needed to wait until the end of the shift, the method can run online and handle such orders “on-the-fly”: all that is needed is to readjust the Lagrange multipliers to accommodate new urgent/unexpected orders. While the formulation is proactive in nature, the methodology possesses an intrinsically reactive feature allowing to further improve manufacturing operations.

   (b) For the above reason, the method is extendable for rescheduling orders whenever scrap happens without waiting for the next shift. Therefore, the method may lead to more resilient operations without disruptions.

Figure 7   The deterministic scheduling results for Example 3: the schedule for Shift 1 is the same as that within Figure 5; because scrap is assumed to occur after operation 2 of each part, the rescheduling is triggered at the beginning of Shift 2 for parts 1 and 2, and at the beginning of Shift 3 for Parts 3 and 4.
3. The much-reduced tardiness is beneficial for the supply chain downstream operations management since the reduction of the wait from the upstream resources will reduce the propagation of the delay in the downstream direction.

4. The SLBLR method does not require any specific structure of the problem with the exception of separability. Therefore, the method can be used to solve other stochastic discrete optimization problems. More generally, even separability of the original problem is not required since optimization with respect to a subset of decision variables is sufficient thereby leading to a drastic reduction of complexity; moreover, the process of obtaining of “level” values to set step-sizes does not require separability. The “level” values are obtained through the auxiliary “multiplier-convergence-feasibility” problem rather than through the adjustments of hyper-parameters.

5. The new method can also be used as a decision support methodology to quickly analyze “what-if” scenarios to quickly provide managerial insights about the value of incoming orders as well as possible liabilities that new orders can bring (in case the entire schedule result is negatively tardiness-wise impacted by a new order).

6. Several rudimentary simulation scenarios indicate that the stochastic schedule is beneficial for reducing the expected tardiness in production.

Learning from test cases of the JSS problems, notoriously difficult and intractable through the use of modern methods, the authors are optimistic that the solution methodology could provide a beacon of hope to address computational challenges as well as pave the way to the decision support system within the entire supply chain and beyond.

**Acknowledgments**

This material is based upon work supported by the U.S. Department of Energy’s Office of Energy Efficiency and Renewable Energy (EERE) under the Advanced Manufacturing Office Award No. DE-EE0007613. We also gratefully acknowledge the Air Force Research Laboratory, Materials and Manufacturing Directorate (AFRL/RXMS) for support via Contract No. FA8650-20-C-5206.

**Disclaimer:** This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views, opinions, and/or findings contained in this paper are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of the United States Government or any agency thereof, including the Air Force Research Laboratory, the United States Air Force, and the Department of Defense.
References

Aarts EH, van Laarhoven PJ, Lenstra JK, Ulder NL (1994) A computational study of local search algorithms for job shop scheduling. ORSA Journal on Computing 6(2):118–125.

Anderson EJ, Nyirenda JC (1990) Two new rules to minimize tardiness in a job shop. The International Journal of Production Research 28(12):2277–2292, URL https://doi.org/10.1080/00207549008942866.

Barman S, Laforge RL (1998) The impact of simple priority rule combinations on delivery speed and delivery reliability in a hybrid shop. Production and Operations Management 7(4):402–416, URL https://doi.org/10.1111/j.1937-5956.1998.tb00132.x.

Beck JC, Feng T, Watson JP (2011) Combining constraint programming and local search for job-shop scheduling. INFORMS Journal on Computing 23(1):1–14.

Bertsekas DP (2016) Nonlinear Programming: Third Edition (Belmont, MA: Athena Scientific), ISBN 978-1-886529-05-2.

Bragin MA (2022) Surrogate “level-based” lagrangian relaxation for mixed-integer linear programming. arXiv preprint arXiv:2203.04514 URL https://arxiv.org/abs/2203.04514.

Bragin MA, Luh PB, Yan B, Sun X (2019) A scalable solution methodology for mixed-integer linear programming problems arising in automation. IEEE Transactions on Automation Science and Engineering 16(2):531–541, URL http://dx.doi.org/10.1109/tase.2018.2835298.

Bragin MA, Luh PB, Yan JH, Yu N, Stern GA (2015) Convergence of the surrogate lagrangian relaxation method. Journal of Optimization Theory and applications 164(1):173–201, URL https://doi.org/10.1007/s10957-014-0561-3.

Chen J, Huang GQ, Luo H, Wang J (2015) Synchronisation of production scheduling and shipment in an assembly flowshop. International Journal of Production Research 53(9):2787–2802, URL https://doi.org/10.1080/00207543.2014.994075.

Cheng BY, Leung JYT, Li K (2015) Integrated scheduling of production and distribution to minimize total cost using an improved ant colony optimization method. Computers & Industrial Engineering 83:217–225, URL https://doi.org/10.1016/j.cie.2015.02.017.

Choi TM (2015) Supply chain systems coordination with multiple risk sensitive retail buyers. IEEE Transactions on Systems, Man, and Cybernetics: Systems 46(5):636–645, URL https://doi.org/10.1109/TSMC.2015.2452894.

Chopra S, Sodhi M, Läcker F (2021) Achieving supply chain efficiency and resilience by using multi-level commons. Decision Sciences URL https://doi.org/10.1111/deci.12526.

Chopra S, Sodhi MS (2004) Managing risk to avoid supply-chain breakdown. MIT Sloan Management Review 46(1):53–61, URL https://sloanreview.mit.edu/article/managing-risk-to-avoid-supplychain-breakdown/.
Chryssolouris G, Pierce J, Dicke K (1991) An approach for allocating manufacturing resources to production tasks. *Journal of Manufacturing Systems* 10(5):368–382, URL https://doi.org/10.1016/0278-6125(91)90055-7.

Chryssolouris G, Wright K, Pierce J, Cobb W (1988) Manufacturing systems operation: dispatch rules versus intelligent control. *Robotics and Computer-Integrated Manufacturing* 4(3-4):531–544, URL https://doi.org/10.1016/0736-5845(88)90026-9.

Dawande M, Gavirneni S, Rachamadugu R (2006) Scheduling a two-stage flowshop under makespan constraint. *Mathematical and computer modelling* 44(1-2):73–84, URL https://doi.org/10.1016/j.mcm.2004.12.016.

Du J, Leung JYT (1990) Minimizing total tardiness on one machine is np-hard. *Mathematics of Operations Research* 15(3):483–495, URL https://doi.org/10.1287/moor.15.3.483.

Ferrio J, Wassick J (2008) Chemical supply chain network optimization. *Computers & Chemical Engineering* 32(11):2481–2504, URL http://dx.doi.org/10.1016/j.compchemeng.2007.09.002.

Garey MR, Johnson DS, Sethi R (1976) The complexity of flowshop and jobshop scheduling. *Mathematics of operations research* 1(2):117–129, URL https://doi.org/10.1287/moor.1.2.117.

Ghasemi A, Ashoori A, Heavey C (2021) Evolutionary learning based simulation optimization for stochastic job shop scheduling problems. *Applied Soft Computing* 106:107309, ISSN 1568-4946, URL http://dx.doi.org/10.1016/j.asoc.2021.107309.

Golenko-Ginzburg D, Gonik A (2002) Optimal job-shop scheduling with random operations and cost objectives. *International Journal of Production Economics* 76(2):147–157, ISSN 0925-5273, URL http://dx.doi.org/10.1016/S0925-5273(01)00140-2.

Grimes D, Hebrard E (2015) Solving variants of the job shop scheduling problem through conflict-directed search. *INFORMS Journal on Computing* 27(2):268–284.

Hasan SK, Sarker R, Essam D (2011) Genetic algorithm for job-shop scheduling with machine unavailability and breakdowns. *International Journal of Production Research* 49(16):4999–5015, URL http://dx.doi.org/10.1080/00207543.2010.495088.

Herrmann JW, Lee CY, Hinchman J (1995) Global job shop scheduling with a genetic algorithm. *Production and Operations Management* 4(1):30–45, URL https://doi.org/10.1111/j.1937-5956.1995.tb0039.x.

Hoitonit DJ, Luh PB, Pattipati KR (1993) A practical approach to job-shop scheduling problems. *IEEE Transactions on Robotics and Automation* 9(1):1–13, URL http://dx.doi.org/10.1109/70.210791.

Horng SC, Lin SS (2021) Apply ordinal optimization to optimize the job-shop scheduling under uncertain processing times. *Arabian Journal for Science and Engineering* 1–13, URL https://doi.org/10.1007/s13369-021-06317-9.
Horng SC, Lin SS, Yang FY (2012) Evolutionary algorithm for stochastic job shop scheduling with random processing time. Expert Systems with Applications 39(3):3603–3610, ISSN 0957-4174, URL http://dx.doi.org/10.1016/j.eswa.2011.09.050.

Iloglu S, Albert LA (2018) An integrated network design and scheduling problem for network recovery and emergency response. Operations Research Perspectives 5:218–231, ISSN 2214-7160, URL http://dx.doi.org/https://doi.org/10.1016/j.orp.2018.08.001.

Jamili A (2019) Job shop scheduling with consideration of floating breaking times under uncertainty. Engineering applications of artificial intelligence 78:28–36, URL https://doi.org/10.1016/j.engappai.2018.10.007.

Kanet JJ, Zhou Z (1993) A decision theory approach to priority dispatching for job shop scheduling. Production and Operations Management 2(1):2–14, URL https://doi.org/10.1111/j.1937-5956.1993.tb00035.x.

Kim YK, Park K, Ko J (2003) A symbiotic evolutionary algorithm for the integration of process planning and job shop scheduling. Computers & operations research 30(8):1151–1171, URL http://dx.doi.org/10.1016/S0305-0548(02)00063-1.

Lakshminarayan S, Lakshmanan R, Papineau RL, Rochette R (1978) Optimal single-machine scheduling with earliness and tardiness penalties. Operations Research 26(6):1079–1082, URL https://doi.org/10.1287/opre.26.6.1079.

Lei D (2011) Simplified multi-objective genetic algorithms for stochastic job shop scheduling. Applied Soft Computing 11(8):4991–4996, ISSN 1568-4946, URL http://dx.doi.org/10.1016/j.asoc.2011.06.001.

Lei DM, Xiong HJ (2007) An efficient evolutionary algorithm for multi-objective stochastic job shop scheduling. 2007 International Conference on Machine Learning and Cybernetics, volume 2, 867–872, URL http://dx.doi.org/10.1109/ICMLC.2007.4370264.

Leung CW, Wong TN, Mak KL, Fung RYK (2010) Integrated process planning and scheduling by an agent-based ant colony optimization. Computers & Industrial Engineering 59(1):166–180, URL https://doi.org/10.1016/j.cie.2009.09.003.

Leung JYT, Chen ZL (2013) Integrated production and distribution with fixed delivery departure dates. Operations Research Letters 41(3):290–293, URL https://doi.org/10.1016/j.orl.2013.02.006.

Li X, Gao L (2016) An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. International Journal of Production Economics 174:93–110, URL https://doi.org/10.1016/j.ijpe.2016.01.016.

Li X, Gao L, Pan Q, Wan L, Chao KM (2018) An effective hybrid genetic algorithm and variable neighborhood search for integrated process planning and scheduling in a packaging machine workshop. IEEE Transactions on Systems, Man, and Cybernetics: Systems 49(10):1933–1945, URL https://doi.org/10.1109/TSMC.2018.2881686.
Lin YK, Chang PC, Yeng LCL, Huang SF (2019) Bi-objective optimization for a multistate job-shop production network using nsga-ii and topsis. *Journal of Manufacturing Systems* 52:43–54, ISSN 0278-6125, URL http://dx.doi.org/10.1016/j.jmsy.2019.05.004.

Liu A, Luh P, Yan B, Bragin M (2021) A novel integer linear programming formulation for job-shop scheduling problems. *IEEE Robotics and Automation Letters* URL https://doi.org/10.1109/LRA.2021.3086422.

Luh PB, Chen D, Thakur LS (1999) An effective approach for job-shop scheduling with uncertain processing requirements. *IEEE Transactions on Robotics and Automation* 15(2):328–339, URL http://dx.doi.org/10.1109/70.760354.

Malapert A, Cambazard H, Guéret C, Jussien N, Langevin A, Rousseau LM (2012) An optimal constraint programming approach to the open-shop problem. *INFORMS Journal on Computing* 24(2):228–244.

Nagurney A (2021) Supply chain game theory network modeling under labor constraints: Applications to the Covid-19 pandemic. *European journal of operational research* 293(3):880–891, URL https://doi.org/10.1016/j.ejor.2020.12.054.

Nemhauser G, Wolsey L (1999) *Integer and Combinatorial Optimization* (New York: John Wiley & Sons, Inc.), URL http://dx.doi.org/10.1002/9781118627372.

Nouiri M, Bekrar A, Jemai A, Niar S, Ammari AC (2018) An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. *Journal of Intelligent Manufacturing* 29(3):603–615, URL https://doi.org/10.1007/s10845-015-1039-3.

Ow PS, Morton TE (1989) The single machine early/tardy problem. *Management science* 35(2):177–191, URL https://doi.org/10.1287/mnsc.35.2.177.

Panwalkar SS, Smith ML, Seidmann A (1982) Common due date assignment to minimize total penalty for the one machine scheduling problem. *Operations Research* 30(2):391–399, URL https://doi.org/10.1287/opre.30.2.391.

Polyak BT (1969) Minimization of unsmooth functionals (in russian). *USSR Computational Mathematics and Mathematical Physics* 9(3):14–29, URL http://dx.doi.org/10.1016/0041-5553(69)90061-5.

Rahman HF, Janardhanan MN, Chuen LP, Ponnambalam SG (2021) Flowshop scheduling with sequence dependent setup times and batch delivery in supply chain. *Computers & Industrial Engineering* 158:107378, URL https://doi.org/10.1016/j.cie.2021.107378.

Rahman HF, Sarker R, Essam D (2017) A genetic algorithm for permutation flowshop scheduling under practical make-to-order production system. *AI EDAM* 31(1):87–103, URL https://doi.org/10.1016/j.cie.2015.08.006.

Rohleder TR, Scudder G (1993) A comparison of order-release and dispatch rules for the dynamic weighted early/tardy problem. *Production and Operations Management* 2(3):221–238, URL https://doi.org/10.1111/j.1937-5956.1993.tb00099.x.
Seidmann A, Panwalkar SS, Smith ML (1981) Optimal assignment of due-dates for a single processor scheduling problem. *The International Journal Of Production Research* 19(4):393–399, URL https://doi.org/10.1080/00207548108956667.

Serdarasan S (2013) A review of supply chain complexity drivers. *Computers & Industrial Engineering* 66(3):533–540, URL http://dx.doi.org/10.1016/j.cie.2012.12.008.

Shen J, Zhu Y (2016) Chance-constrained model for uncertain job shop scheduling problem. *Soft Computing* 20(6):2383–2391, URL https://doi.org/10.1007/s00500-015-1647-z.

Sidney JB (1977) Optimal single-machine scheduling with earliness and tardiness penalties. *Operations Research* 25(1):62–69, URL https://doi.org/10.1287/opre.25.1.62.

Taillard ED (1994) Parallel taboo search techniques for the job shop scheduling problem. *ORSA journal on Computing* 6(2):108–117.

Tavakkoli-Moghaddam R, Jolai F, Vaziri F, Ahmed P, Azaron A (2005) A hybrid method for solving stochastic job shop scheduling problems. *Applied Mathematics and Computation* 170(1):185–206, ISSN 0096-3003, URL http://dx.doi.org/10.1016/j.amc.2004.11.036.

Thomas D, Helgeson J (2021) The effect of natural/human-made hazards on business establishments and their supply chains. *International Journal of Disaster Risk Reduction* 59:102257, ISSN 2212-4209, URL http://dx.doi.org/https://doi.org/10.1016/j.ijdrr.2021.102257.

Vaessens RJM, Aarts EH, Lenstra JK (1996) Job shop scheduling by local search. *Informs Journal on computing* 8(3):302–317.

Vepsalainen APJ, Morton TE (1987) Priority rules for job shops with weighted tardiness costs. *Management Science* 33(8):1035–1047, URL https://doi.org/10.1287/mnsc.33.8.1035.

Voudouris VT (1996) Mathematical programming techniques to debottleneck the supply chain of fine chemical industries. *Computers & Chemical Engineering* 20:S1269–S1274, URL http://dx.doi.org/10.1016/0098-1354(96)00219-0.

Wang G (2021) Integrated supply chain scheduling of procurement, production, and distribution under spillover effects. *Computers & Operations Research* 126:105105, URL https://doi.org/10.1016/j.cor.2020.105105.

Wang K, Luo H, Liu F, Yue X (2017) Permutation flow shop scheduling with batch delivery to multiple customers in supply chains. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 48(10):1826–1837, URL https://doi.org/10.1109/TSMC.2017.2720178.

Wilhelm ME, Bragin M, Stuber MD (2022) Jobshop.jl. URL https://github.com/PSORLab/Jobshop.jl.

Yan B, Bragin MA, Luh PB (2018) Novel formulation and resolution of job-shop scheduling problems. *IEEE Robotics and Automation Letters* 3(4):3387–3393, URL https://doi.org/10.1109/LRA.2018.2850056.
Yan B, Bragin MA, Luh PB (2021) An innovative formulation tightening approach for job-shop scheduling. IEEE Transactions on Automation Science and Engineering URL https://doi.org/10.1109/TASE.2021.3088047.

Yang Ha, Lv Y, Xia C, Sun S, Wang H (2014) Optimal computing budget allocation for ordinal optimization in solving stochastic job shop scheduling problems. Mathematical Problems in Engineering 2014, URL https://doi.org/10.1155/2014/619254.

Zhang J, Wang X, Huang K (2016) Integrated on-line scheduling of order batching and delivery under b2c e-commerce. Computers & Industrial Engineering 94:280–289, URL https://doi.org/10.1016/j.cie.2016.02.001.

Zhang J, Wang X, Huang K (2018) On-line scheduling of order picking and delivery with multiple zones and limited vehicle capacity. Omega 79:104–115, URL https://doi.org/10.1016/j.omega.2017.08.004.

Zhang R, Song S, Wu C (2013) A hybrid differential evolution algorithm for job shop scheduling problems with expected total tardiness criterion. Applied Soft Computing 13(3):1448–1458, URL https://doi.org/10.1016/j.asoc.2012.02.024.

Zhao X, Luh PB, Wang J (1999) Surrogate gradient algorithm for lagrangian relaxation. Journal of optimization Theory and Applications 100(3):699–712, URL https://doi.org/10.1023/A:1022646725208.

Zhong W (2015) An improved algorithm for integrated production and distribution scheduling problem with committed delivery dates. Optimization Letters 9(3):537–567, URL https://doi.org/10.1007/s11590-014-0755-5.

Zhu X, Wang W, Guo X, Shi L (2020) A genetic programming-based evolutionary approach for flexible job shop scheduling with multiple process plans. 2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), 49–54 (IEEE), URL https://doi.org/10.1109/CASE48305.2020.9216783.

Appendix. Example Data
Table A1  Processing time $p_{i,j}$ data for job $i$ and operation $j$ for Example 1 are contained in this table.

| $i$ | $j$ | $p_{i,j}$ | $i$ | $j$ | $p_{i,j}$ | $i$ | $j$ | $p_{i,j}$ | $i$ | $j$ | $p_{i,j}$ |
|-----|-----|-----------|-----|-----|-----------|-----|-----|-----------|-----|-----|-----------|
| 1   | 1   | 1         | 3   | 1   | 3         | 5   | 1   | 3         | 7   | 1   | 3         |
| 1   | 2   | 2         | 3   | 2   | 2         | 5   | 2   | 2         | 7   | 2   | 2         |
| 1   | 3   | 3         | 3   | 3   | 3         | 5   | 3   | 3         | 7   | 3   | 3         |
| 1   | 4   | 4         | 4   | 4   | 4         | 5   | 4   | 4         | 7   | 4   | 4         |
| 1   | 5   | 5         | 5   | 5   | 5         | 5   | 5   | 5         | 7   | 5   | 5         |
| 2   | 1   | 1         | 3   | 1   | 3         | 5   | 1   | 3         | 7   | 1   | 3         |
| 2   | 2   | 2         | 3   | 2   | 2         | 5   | 2   | 2         | 7   | 2   | 2         |
| 2   | 3   | 3         | 3   | 3   | 3         | 5   | 3   | 3         | 7   | 3   | 3         |
| 2   | 4   | 4         | 4   | 4   | 4         | 5   | 4   | 4         | 7   | 4   | 4         |
| 2   | 5   | 5         | 5   | 5   | 5         | 5   | 5   | 5         | 7   | 5   | 5         |

