A Novel Integer Linear Programming Formulation for Job-Shop Scheduling Problems

SUBMISSION DATE / POSTED DATE

15-05-2021 / 19-05-2021

CITATION

Liu, Anbang; Luh, Peter; Yan, Bing; Bragin, Mikhail (2021): A Novel Integer Linear Programming Formulation for Job-Shop Scheduling Problems. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.14601123.v1

DOI

10.36227/techrxiv.14601123.v1

A Novel Integer Linear Programming Formulation for Job-Shop Scheduling Problems

Anbang Liu, Peter B. Luh, *Life Fellow*, Bing Yan, *Member*, *IEEE*, Mikhail A. Bragin, *Member*, *IEEE*

***Abstract* – Job-shop scheduling is an important but difficult problem arising in low-volume high-variety manufacturing. It is usually solved at the beginning of each shift with strict computational time requirements. To obtain near-optimal solutions with quantifiable quality within strict time limits, a direction is to formulate them in an Integer Linear Programming (ILP) form so as to take advantages of widely available ILP methods such as Branch-and-Cut (B&C). Nevertheless, computational requirements for ILP methods on existing ILP formulations are high. In this paper, a novel ILP formulation for minimizing total weighted tardiness is presented. The new formulation has much fewer decision variables and constraints, and is proven to be tighter as compared to our previous formulation. For fast resolution of large problems, our recent decomposition-and-coordination method "Surrogate Absolute-Value Lagrangian Relaxation" (SAVLR) is enhanced by using a 3-segment piecewise linear penalty function, which more accurately approximates a quadratic penalty function as compared to an absolute-value function. Testing results demonstrate that our new formulation drastically reduces the computational requirements of B&C as compared to our previous formulation. For large problems where B&C has difficulties, near-optimal solutions are efficiently obtained by using the enhanced SAVLR under the new formulation.**

***Index terms*–Manufacturing, job-shop scheduling, integer linear programming, decomposition and coordination**

I. Introduction

Job shops are manufacturing systems designed for low-volume/high-variety production [1]. In a job shop, machines are grouped based on their functionalities, and each group has limited capacities. A part may need to go through a sequence of operations, each can be processed by one or a few machine groups. A schedule is usually generated with strict computational time requirements, e.g., 10 to 20 minutes, at the beginning of a shift. The scheduling problem is subject to four types of constraints: part-to-machine assignment constraints, processing time requirements, operation precedence constraints, and machine capacity constraints. To have on-time delivery – the ultimate goal of job-shop scheduling, the objective function should be due date related, e.g., weighted tardiness penalties. The problem is difficult because of its combinatorial nature.

To obtain near-optimal solutions within strict time limits, a direction is to formulate them in an Integer Linear Programming (ILP) form so as to take advantages of widely available ILP methods such as Branch-and-Cut (B&C) [2]–[4]. Nevertheless, the efficiency of ILP methods is significantly affected by problem formulations, and when a formulation contains large numbers of decision variables and constraints, these methods may experience difficulties [5]. Developing good formulations is thus of critical importance. This, however, is difficult in view of the many types of complicated constraints within job-shop scheduling problems.

As will be reviewed in Section II, an ILP formulation was presented in [6] based on our classic formulation [7], with very recent extensions reported in [8]. In [6], operation beginning times were selected as integer decision variables. Processing time requirements and operation precedence constraints were easily formulated based on them. To consider machine capacity constraints, an additional set of binary indicator variables was created to indicate the status of operations: if an operation is active on a machine group at a time slot (i.e., being processed), then the corresponding indicator variable equals one; and zero otherwise. These indicator variables depend on operation beginning times, and to describe such relationships, many constraints are needed. Consequently, the computational requirements of ILP methods for large problems are high.

To overcome the above-mentioned difficulties, a novel ILP formulation will be developed in Section III. In the formulation, the set of binary indicator variables indicating whether an operation begins at a time slot on a machine group is selected as decision variables. If an operation begins at a certain time slot on a machine group, then the corresponding indicator variable equals one; and zero otherwise. Based on these variables, all constraints, including machine capacity constraints, are innovatively formulated without introducing additional decision variables or constraints. The numbers of decision variables and constraints are thus significantly reduced as compared to those of [6]. To theoretically demonstrate the advantages of our formulation over that of [6] beyond counting the numbers of decision variables and constraints, the two formulations are compared in terms of "tightness" – a novel concept to examine MILP formulations [8]. A formulation is "tight" if its constraints directly delineate the convex hull of the feasible solution set. In this case, the problem can be directly solved by using Linear Programming (LP) methods. Suppose that the two formulations of a problem have the same convex hull. One formulation is "tighter" if the solution set of its LP-relaxed formulation (with integrality requirements relaxed) is a proper subset of the solution set of the other LP-relaxed formulation. We prove that our new formulation is tighter than the previous formulation. Since the new formulation has fewer

This work is supported in part by Chinese National Innovation Center of High Speed Train R&D project "Modeling and comprehensive intelligent optimization for new high efficiency urban rail transit system" under grant No. CX/KJ-2020-0006.

Anbang Liu is with the Center for Intelligent and Networked System (CFINS), Department of Automation, Tsinghua University, Beijing 10084, China (e-mail: liuab19@mails.tsinghua.edu.cn).

Peter B. Luh and Mikhail A. Bragin are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269-4157, USA (e-mail: peter.luh@uconn.edu and mikhail.bragin@uconn.edu.).

Bing Yan is with the Department of Electrical and Microelectronic Engineering at Rochester Institute of Technology, Rochester, NY 14623, USA (e-mail: bxyeee@rit.edu).

decision variables and constraints and is tighter as compared to the previous formulation, solution quality of B&C is much improved, while computational requirements are much smaller, as will be supported by numerical testing results in Section V.

Since our new formulation has low computational requirements, B&C can solve small- or medium-sized problems efficiently. For large problems, B&C may still suffer from poor performance. Our recent decomposition-and-coordination method "Surrogate Absolute-Value Lagrangian Relaxation" (SAVLR) is enhanced in Section IV. SAVLR exploits exponential reduction of complexity upon problem decomposition, and effectively coordinates subproblem solutions with accelerated convergence [9]. In the method, machine capacity constraints, which couple various parts assigned to a machine group, are "relaxed." Instead of using powerful quadratic penalty functions as in Augmented Lagrangian Relaxation [10], violations of these constraints are penalized by using an absolute-value function. This absolute-value function is piece-wise linear with two segments, and is exactly linearized by introducing additional decision variables and constraints so as to use ILP solvers. The absolute-value function, however, is a poor approximation to a quadratic function, and the quadratic growth of penalties cannot be well captured. When the level of constraint violation is large, the absolute-value penalty function may not impose a sufficiently large penalty, especially at the early stage of optimization. To overcome this difficulty, the absolute-value penalty function is enhanced by using a symmetric, 3-segment piecewise linear function, where the middle segment is a constant. This function better captures the quadratic growth of penalties as compared to the absolute-value function. In addition, it can be exactly linearized, and the numbers of additional decision variables and constraints needed for linearization are the same as those required by the absolute-value function. Extension to multiple-segment piecewise linear penalty functions is possible.

To demonstrate the performance of our new formulation and the enhanced SAVLR (or SAVLRE for short), three examples are presented in Section V. The first and second are of small and medium sizes, respectively, and are solved by using B&C. The results demonstrate that our new formulation drastically reduces the computational requirements of B&C as compared with the original formulation of [6], and optimal solutions are efficiently obtained. The third example with five large instances is solved by using SAVLRE. Testing results demonstrate that the convergence of multipliers is improved by using the new 3-segment penalty function as compared to using the absolute-value penalty function, and near-optimal solutions are therefore obtained in a computationally efficient manner.

II. Literature Review

In this section, existing job-shop scheduling formulations are reviewed in subsection A. Solution methodologies are then discussed in subsection B.

A. Problem formulations

With complicated constraints, formulating a job-shop scheduling problem is difficult [1]. ILP formulations were reported in [11]–[14]. In these formulations, the objective is usually to minimize the makespan, i.e., the time span from the very beginning to the end when all the parts are processed. The makespan, however, is a questionable objective function. The reason is that for practical job shops, the paramount objective is on-time deliveries of parts, and part due dates need to be considered in the objective function. The makespan, however, does not even include due dates in the formulation, and cannot capture the on-time delivery performance. The total weighted tardiness is a much practical objective function, and has been used in most of our previous work [6], [7]. As related to constraints, operations are usually assigned to individual machines, and overlapping of two operations on a machine is prohibited. In view that there are many possible sequences among the operations assigned to a machine, the number of such constraints are large, limiting the size of problems that can be solved by optimization. Practically, there are multiple machines of the same functionality in a job shop. The consideration of groups of machines with the same functionality as opposed to individual machines is therefore a better modeling approach.

In [7], one of our earlier papers, a separable nonlinear formulation was presented. Operation beginning times were selected as integer decision variables, indicating when each operation is to begin on an eligible machine group (as opposed to begin on an eligible machine). The objective is to minimize the total weighted tardiness. Machine capacity constraints limit the number of active operations (i.e., operations that are being processed) on a group. Consider for example a machine group with five machines available at a particular time slot. Then at most five operations can be active at that time slot. These machine capacity constraints couple parts together and are additive. The objective function is also part-wise additive. Therefore, after machine capacity constraints are relaxed, the problem can be decomposed into part subproblems, each with much-reduced complexity.

In our recent work [6], an ILP formulation was developed based on [7]. In the formulation, operation beginning times were selected as integer decision variables. From these variables, processing time requirements and operation precedence constraints were easily formulated. The difficulty is the modeling of machine capacity constraints. An additional set of binary indicator variables was created to indicate if an operation is active on an eligible machine group at each time slot following the idea of [7]. Then the sum of these indicator variables over all relevant operations for a machine group should be less than or equal to the number of machines available in that group at each time slot. This set of indicator variables is related to operation beginning times. To describe such relationships, a significant number of additional constraints are needed. Therefore, although this formulation is linear, B&C might suffer from poor performance for large problems. Very recently, the formulation in [6] was tightened through a systematic approach by transforming the constraints to directly delineate the convex hull of the feasible solution set in [8], and the computational efficiency of B&C has thus been improved.

B. Solution methodologies

In this subsection, heuristics, branch-and cut, standard Lagrangian Relaxation, Surrogate Lagrangian Relaxation and Surrogate Absolute-Value Lagrangian Relaxation are briefly reviewed to solve job shop scheduling problems.

Heuristics. Heuristics such as shift bottleneck method [15], [16] and meta-heuristics such as Tabu search [17] are frequently used to solve job-shop scheduling problems. These methods have the advantage of low computational requirements. However, convergence is difficult to guarantee, and the quality of solutions is difficult to quantify.

Branch-and-Cut (B&C). Branch-and-Cut (B&C) has been used to solve ILP job-shop scheduling problems, e.g., [6], [12], [13]. The key idea of B&C is to find the convex hull of the feasible solution set through adding "valid cuts." If the convex hull is found, then the optimal solution can be obtained by using LP methods. If the method fails to obtain the convex hull, or valid cuts are difficult to obtain or are ineffective, then time-consuming Branch-and-Bound is used. Since finding convex hulls of feasible solution sets itself is NP-hard, B&C may suffer from poor performance for large problems.

Standard Lagrangian Relaxation. Lagrangian Relaxation (LR) methods were frequently used to solve ILP problems [7], [18]. In [7], after relaxing coupling machine capacity constraints, the "relaxed problem" is decomposed into smaller subproblems, one for each part. Since subproblems have much-reduced sizes, their complexity is drastically reduced as compared to that of the original problem. Subproblem solutions are then coordinated through iterative updating of multipliers based on subgradients. However, since all subproblems need to be solved to obtain a subgradiant and the stepsizes require the knowledge of the unknown optimal dual value, the performance of traditional LR is generally poor. To accelerate convergence, the violations of coupling constraints can be penalized by quadratic penalty terms which can be approximated by using piecewise linear functions as in [18]. Nevertheless, [18] did not provide the equation for the piecewise linear functions, and did not examine how many segments should be used and why.

Surrogate Lagrangian Relaxation (SLR) and Surrogate Absolute-Value Lagrangian Relaxation (SAVLR). By exploiting a contraction mapping concept, our recent SLR ([10]) was developed without requiring the knowledge of the optimal dual value to calculate stepsizes. Moreover, multipliers are updated without requiring all subproblems to be solved, thereby reducing the high computational requirements and the multiplier zigzagging issues. Most of the major difficulties of traditional LR have thus been overcome. Convergence is further improved in the SAVLR method in [9], where absolute-value penalty functions (2-segment piecewise linear functions) are used to penalize the levels of machine capacity violations in [6]. This penalty function can be exactly linearized with very few additional constraints. SAVLR is usually combined with B&C to efficiently solve ILP problems.

III. A Novel Integer Linear Formulation

In this section, a novel ILP formulation for job-shop scheduling problems based on that of [6, 8] is presented in subsection A. The advantages of the formulation are then analyzed in subsection B.

A. Problem formulations

Consider a job shop with M machine groups. The capacity of machine group m at time slot t is denoted as $M_{m,t}$ for $m \in [1, 2, .., M]$ and $t \in [1, 2, ..., T]$, where T is the total number of time slots and is assumed to be long enough to process all the parts. There are I parts, each with an arrival time a_i and a due date d_i. Part $i \in [1, 2, ..., I]$ needs to go through a sequence of J_i operations, and the j^{th} operation of part i is denoted as (i, j) – a "part-operation pair." Let the set of all part-operation pairs be denoted as S. An operation can be processed by one of the eligible machine groups $U_{i,j}$, and the processing time of operation (i, j) on machine group m is denoted as $p_{i,j,m}$, which may be machine group dependent. It is assumed that processing cannot be interrupted, i.e., non-preemptive. In the following, decision variables are first introduced. Then four types of constraints, including part-to-machine assignment constraints, processing time requirements, operation precedence constraints, and machine capacity constraints, are formulated, followed by the objective function.

a) Decision variables

As reviewed in subsection II.A, operation beginning times are integer decision variables in the formulations of [6], [8], resulting in large numbers of additional variables and constraints. To overcome this difficulty, a set of binary indicator variables indicating whether an operation begins at a time slot on a machine group is selected as decision variables. If operation (i, j) is to begin on machine group m at time slot t, then $b_{i,j,m,t}$ equals one; and it equals zero otherwise.

b) Part-to-machine assignment constraints

Since each operation must be assigned to a unique machine group and to begin at a unique time slot, part-to-machine assignment constraints are modeled as follows:

$$\sum_{\forall m \in U_{i,j}} \sum_{t=l_{i,j}}^{u_{i,j}} b_{i,j,m,t} = 1, \forall (i,j) \in S. \tag{1}$$

In the above, the range $[l_{i,j}, u_{i,j}]$ contains all eligible beginning times for operation (i, j). The method to calculate $[l_{i,j}, u_{i,j}]$ will be described later in (9) and (10).

c) Processing time requirements

From the operation beginning indicator variables, integer operation beginning times are obtained as

$$b_{i,j} = \sum_{\forall m \in U_{i,j}} \sum_{t=l_{i,j}}^{u_{i,j}} t \cdot b_{i,j,m,t}, \forall (i,j) \in S, \tag{2}$$

where the integer variable $b_{i,j}$ is the beginning time of operation (i, j). Since processing is non-preemptive, the completion time equals its beginning time plus the required processing time following equation (2) in [6], i.e.,

$$c_{i,j} = \sum_{\forall m \in U_{i,j}} \sum_{t=l_{i,j}}^{u_{i,j}} (t + p_{i,j,m}) b_{i,j,m,t} - 1, \forall (i,j) \in S, \tag{3}$$

where $c_{i,j}$ is the completion time of operation (i, j). Note that $\{b_{i,j}\}$ and $\{c_{i,j}\}$ are introduced here for easy understanding and presentation. They are not decision variables in the solution process to be introduced in Section IV.

d) Operation precedence constraints

For each part, its operations need to be processed in a given sequence. Without loss of generality, operations for a part are numbered according to their precedence, and operation $(i, j+1)$ cannot begin until operation (i, j) is completed, i.e.,

$$b_{i,j+1} > c_{i,j}, \forall (i,j) \in \{(i,j) \mid (i,j) \in S, (i,j+1) \in S\}. \tag{4}$$

Equation (4) can be re-written without $\{b_{i,j}\}$ or $\{c_{i,j}\}$ as:

$$\sum_{\forall m\in U_{i,j}}\sum_{t=l_{i,j}}^{u_{i,j}} t\cdot b_{i,j+1,m,t} \ge \sum_{\forall m\in U_{i,j}}\sum_{t=l_{i,j}}^{u_{i,j}} (t+p_{i,j,m})b_{i,j,m,t}, \quad (5)$$

$$\forall (i,j)\in\{(i,j)\,|\,(i,j)\in S,(i,j+1)\in S\}.$$

e) Machine capacity constraints

To formulate machine capacity constraints, it is noticed that if operation (i, j) is active on machine group m at time t, then its beginning time must be within the interval $[t-p_{i,j,m}+1,\ t]$. Therefore, the status (active or not) of operation (i, j) on machine group m at time t, represented by $\delta_{i,j,m,t}$, can be obtained by summing up the operation beginning indicator variables over $[t-p_{i,j,m}+1,\ t]$:

$$\delta_{i,j,m,t} = \sum_{\forall k\in[t-p_{i,j,m}+1,t]} b_{i,j,m,k}, \forall (i,j)\in S, m\in U_{i,j}, \forall t. \quad (6)$$

Again, variables $\{\delta_{i,j,m,t}\}$ are introduced for easy understanding and presentation. They are not decision variables.

Based on (6), the number of active operations on machine group m at time slot t is obtained by summing up the statuses of all relevant operations that can be processed by the machine group. Machine capacity constraints can thus be formulated as:

$$\sum_{(i,j)\in O_m}\ \sum_{\forall k\in[t-p_{i,j,m}+1,t]} b_{i,j,m,k} \le M_{m,t}, \forall t, \forall m, \quad (7)$$

where O_m denotes the set of operations that can be processed by machine group m.

f) The Objective function

Following [6], [7], the objective function to be minimized is the total weighted tardiness. The tardiness of part i is the number of time slots being late, i.e., the number of time slots that the completion time of the last operation of part i exceeds the due date d_i. It is thus described by $\max(c_{i,J_i}-d_i,0)$, where (i, J_i) is the last operation of part i. The total weighted tardiness is thus the sum of the weighted tardiness of all the parts:

$$f(c) \equiv \sum_i w_i\cdot\max(c_{i,J_i}-d_i,0), \quad (8)$$

where w_i is the weight or the importance of part i. Equation (8) can be easily re-written without $\{b_{i,j}\}$ or $\{c_{i,j}\}$.

g) Ranges of operation beginning times

To reduce the decision space, possible beginning time slots of operation (i, j), i.e., $[l_{i,j}, u_{i,j}]$, need to be delineated. The lower limits $\{l_{i,j}\}$ and the upper limits $\{u_{i,j}\}$ are derived at the data-preprocessing stage as follows. The earliest possible beginning time slot of the first operation of part i is the part arrival time, i.e., $l_{i,1} = a_i$. A subsequent operation cannot begin until there is enough time to complete all the proceeding operations, i.e.,

$$l_{i,j} = a_i + \sum_{k=1}^{j-1}\min_{m\in U_{i,j}}(p_{i,k,m}), \forall j\ge 2. \quad (9)$$

There is a minimization in (9) since processing times may depend on the selections of machine groups, which cannot be predetermined. The smallest processing times are used to give the maximal flexibility in selecting the beginning time slot. Similarly, an operation cannot begin too late so that there is not enough time to complete it and its subsequent operations. We therefore have:

$$u_{i,j} = T - \sum_{k=j}^{J_i}\min_{m\in U_{i,j}}(p_{i,k,m}), \forall j. \quad (10)$$

Here, the smallest processing times are also used.

Based on the above, the job-shop scheduling problem can be described as:

$$\min_b\{\sum_i w_i\cdot\max(\sum_{\forall m\in U_{i,J_i}}\sum_{t=l_{i,J_i}}^{u_{i,J_i}}(t+p_{i,J_i,m})b_{i,J_i,m,t}-1-d_i,0)\}, \quad (11)$$

$$s.t.(1),(5),(7), b_{i,j,m,t}\in\{0,1\}, \forall i,j,m,t.$$

The above objective function with the nonlinear "max" within it can be linearized by introducing additional non-negative integer decision variables $\{z_i\}$ following page 150 of [19] as:

$$\min_{b,z}\{\sum_i w_i\cdot z_i\},$$

subject to constraints (1), (5), (7) and additional inequalities

$$\sum_{\forall m\in U_{i,J_i}}\sum_{t=l_{i,J_i}}^{u_{i,J_i}}(t+p_{i,J_i,m})b_{i,J_i,m,t}-1-d_i\le z_i, \forall i. \quad (12)$$

In view that part-to-machine assignment constraints (1) and operation precedence constrains (5) are only associated with individual parts, and machine capacity constraints (7) that couple operations together and the objective function (8) are part-wise additive, the above formulation is separable.

B. The advantages of our new formulation

It is difficult to compare our formulation directly with that of [6] since for the latter, each operation can only be processed by one machine group rather than by multiple machine groups. This limitation has been removed in [8] before the formulation is tightened – the theme of [8]. In view that tightening is a new and dedicated research topic and has not been addressed for the new formulation, the new formulation is thus compared with the untightened formulation of [8], i.e.,

$$\begin{aligned}
&\min_{\delta,b,c,x}\{\sum_i w_i\cdot\max(c_{i,J_i}-d_i,0)\},\\
&s.t.\ \sum_{m\in U_{i,j}} x_{i,j,m}=1, \forall (i,j)\in S, (13a)\\
&c_{i,j} = b_{i,j} + \sum_{m\in U_{i,j}} x_{i,j,m}p_{i,j,m} - 1, \forall (i,j)\in S, (13b)\\
&t\ge b_{i,j} - M(1-\sum_{m\in U_{i,j}}\delta_{i,j,m,t}), \forall (i,j)\in S, t, (13c)\\
&t\le c_{i,j} + M(1-\sum_{m\in U_{i,j}}\delta_{i,j,m,t}), \forall (i,j)\in S, t, (13d)\\
&\sum_t \delta_{i,j,m,t} = x_{i,j,m}p_{i,j,m}, \forall (i,j)\in S, m, (13e)\\
&b_{i,1}\ge a_i, b_{i,j+1}\ge c_{i,j}+1, \forall (i,j)\in S, (13f)\\
&\sum_{\forall (i,j)\in O_m}\delta_{i,j,m,t}\le M_{m,t}, \forall m,t. (13g)
\end{aligned} \quad (13)$$

To start with, our new formulation has significantly fewer decision variables and constraints as compared to the previous formulation [8] (now as (13)). To demonstrate this, consider a problem with M machine groups, T time slots and a total of N operations. In the new formulation, the number of decision variables is $N{\cdot}M{\cdot}T$, and the total number of constraints is $M{\cdot}T+2{\cdot}N$. In the previous formulation (13), the total number of decision variables is $N{\cdot}M{\cdot}T+2N+M{\cdot}N$, and the total number of constraints is at least $M{\cdot}T+2T{\cdot}N$. Since the number of time slots considered is usually much larger than 1, e.g., T = 300 in [8], our new formulation has much fewer decision variables and constraints.

To theoretically demonstrate the advantages of our formulation over that of [8] beyond counting the numbers of decision variables and constraints, it is proved that our new formulation is tighter. This is done in two steps. First, the two formulations are shown to have the same convex hull of the feasible solution set after rewriting the new formulation in terms of the operation status variables $\{\delta_{i,j,m,t}\}$ of the previous formulation (13). It is then shown by a counter example that the feasible solution set of LP-relaxed formulation of the new formulation is a proper subset of that of (13).

a) Equivalent formulation

In view that (6) determines a one-to-one relationship between operation beginning times $\{b_{i,j,m,t}\}$ and operation statuses $\{\delta_{i,j,m,t}\}$, our new formulation (11) can be rewritten as an equivalent problem with $\{\delta_{i,j,m,t}\}$ as decision variables:

$$\min_{\delta,b}\{\sum_i w_i \cdot \max(\sum_{\forall m\in U_{i,J_i}}\sum_{t=l_{i,J_i}}^{u_{i,J_i}}(t+p_{i,J_i,m})b_{i,J_i,m,t}-1-d_i,0)\}, \quad (14)$$

$$s.t.(1),(5),(6),(7),\delta_{i,j,m,t},b_{i,j,m,t}\in\{0,1\},\forall i,j,m,t.$$

Since the formulation (14) and the previous formulation (13) have the same set of feasible $\{\delta_{i,j,m,t}\}$, they have the same convex hull of the solution set.

b) Comparing solution sets of LP-relaxed formulations

After relaxing integrality requirements of (14), the LP-relaxed problem of our new formulation can be formed as

$$\min_{\delta,b}\{\sum_i w_i \cdot \max(\sum_{\forall m\in U_{i,J_i}}\sum_{t=l_{i,J_i}}^{u_{i,J_i}}(t+p_{i,J_i,m})b_{i,J_i,m,t}-1-d_i,0)\}, \quad (15)$$

$$s.t.(1),(5),(6),(7),0\le\delta_{i,j,m,t}\le 1,0\le b_{i,j,m,t}\le 1,\forall i,j,m,t.$$

To demonstrate that the solution set of (15) is a proper subset of that of the previous formulation (13), two propositions are presented. In Proposition 1, it is demonstrated that any continuous solution $\{\delta_{i,j,m,t}\}$ feasible to (15) is also feasible to the LP-relaxed problem of the previous formulation (13). In Proposition 2, a continuous solution $\{\delta_{i,j,m,t}\}$ is given that is feasible to the LP-relaxed problem of the previous formulation (13) but not feasible to (15).

Proposition 1. Any continuous solution $\{\delta_{i,j,m,t}\}$ feasible to (15) of the new formulation is also feasible to the LP-relaxed problem of the previous formulation (13).

Proof. Consider a solution $\{\delta_{i,j,m,t}\}$ that is feasible to (15). The set of beginning times $\{b_{i,j,m,t}\}$ can be determined by (6). In the following, solution $\{\delta_{i,j,m,t}, b_{i,j}, c_{i,j}, x_{i,j,m}\}$ with $\{b_{i,j}\}$ satisfying (2), $\{c_{i,j}\}$ satisfying (3), and $\{x_{i,j,m}\}$ satisfying

$$x_{i,j,m}=\sum_t b_{i,j,\mathrm{m},t},\quad \forall i,j,m, \quad (16)$$

is proven to be feasible to the LP-relaxed problem of the previous formulation (13) as follows.

Satisfaction of the part-to-machine assignment constraints (13a). Since (1) and (16) hold, (13a) are clearly satisfied.

Satisfaction of the processing time requirements (constraints (13b), (13c), (13d), (13e)). In [8], the processing time requirements are described by two equality constraints (13b) and (13e) and two "big-M" inequality constraints (13c) and (13d). Since (13b) can be easily derived from (13e), the satisfaction of (13e) is first proved. By using the relationship between $\{\delta_{i,j,m,t}\}$ and $\{b_{i,j,m,t}\}$ as described by (6), the left-hand side of (13e) can be written as a sum of $\{b_{i,j,m,t}\}$, and it can then be shown that (13e) are satisfied. Based on this, the satisfaction of constraints (13b) can be easily proved.

The satisfaction of (13c) and (13d) can be proved as follows. When operation (i, j) is active at time slot t, it can be shown that its beginning time (i.e., the right-hand side of constraints (13c)) is always smaller than or equal to t, and therefore constraints (13c) are satisfied. For other cases, constraints (13c) are always satisfied since M (as in the "big-M") is a large number. It can be similarly shown that constraints (13d) are satisfied.

Satisfaction of the operation precedence constraints and machine capacity constraints (constraints (13f) and (13g)). By using (2), (3) and (6), constraints (13f) and (13g) of [8] can be rewritten in the form of (5) and (7) of our new formulation, and it can be shown that they are satisfied. ■

In the following, a continuous solution $\{\delta_{i,j,m,t}\}$ is given that is feasible to the LP-relaxed problem of the previous formulation (13) but not feasible to (15). To guarantee that the machine capacity constraints (13g) are not violated, the total number of time slots T and the machine capacity $\{M_{m,t}\}$ should not be too small. Specifically, suppose that they satisfy the following inequality:

$$\max_m(\sum_{(i,j)\in O_m}\frac{1}{\lfloor T/p_{i,j,m}\rfloor\left|U_{i,j}\right|}-\min_{\mathrm{t}}(M_{m,t}))\le 0. \quad (17)$$

This condition is generally easy to satisfy, and is satisfied for all the instances tested in Section V.

Proposition 2. Under condition (17), the solution

$$\delta_{i,j,m,t}=\begin{cases}1/Y_{i,j,m}\left|U_{i,j}\right|, & \forall i,j,m,t\in\{1,...,Y_{i,j,m}p_{i,j,m}\},\\ 0, & \forall i,j,m,t\in\{Y_{i,j,m}p_{i,j,m}+1,...,T\},\end{cases} \quad (18)$$

with $Y_{i,j,m}=\lfloor T/p_{i,j,m}\rfloor$,

is feasible to the LP-relaxed problem of the previous formulation (13), but not feasible to (15).

Proof. Under (17), it can be checked that solution (18) satisfies all the constraints of the LP-relaxed formulation of previous formulation (13). However, (18) is not feasible to (15), and this can be demonstrated as follows. When the arrival times of all the parts are 1, then it can be shown that (5) and (6) cannot be satisfied at the same time. When there exists at least one part whose arrival time is greater than 1, then it can be shown that constraints (5) and (1) cannot be satisfied at the same time.

IV. Solution Methodology

As will be demonstrated in Section V, B&C can solve small- or medium-sized problems based on the new formulation. For large problems, B&C may still suffer from difficulties because of the combinatorial nature of the problem. For fast resolution of such problems, SAVLR [9] is enhanced in this section.

In SAVLR, the problem is decomposed into subproblems by relaxing the coupling machine capacity constraints (7), and subproblem solutions are coordinated by iteratively updating multipliers. After the machine capacity constraints (7) are relaxed by using Lagrangian multipliers $\{\lambda_{t,m}\}$, the relaxed problem at iteration k is formed as:

$$\min_{z,b,s} \left\{ \begin{array}{l} \sum_i w_i \cdot z_i + \sum_t \sum_m \lambda_{t,m}^k \left(g_{t,m}(b) + s_{t,m} \right) + \\ c^k \sum_t \sum_m \left| g_{t,m}(b) + s_{t,m} \right| \end{array} \right\}, \quad (19)$$

subject to (1), (5), (13), where

$$g_{t,m}(b) \equiv \sum_{(i,j) \in O_m} \sum_{\forall k \in [t - p_{i,j,m} + 1, t]} b_{i,j,m,k} - M_{m,t}, \quad (20)$$

and $\{s_{t,m}\}$ are non-negative slack variables introduced to convert inequality constraints (7) to equality constraints. In (19), the absolute-value penalty function with the positive penalty coefficient c is used instead of a quadratic penalty function as in the Augmented Lagrangian Relaxation method ([10]) to facilitate the use of B&C. It is piecewise linear with two segments, and can be exactly linearized by introducing additional decision variables and constraints. However, since the absolute-value function is a poor approximation to a quadratic function, the quadratic growth characteristics of quadratic penalty functions cannot be well captured. When multipliers are far away from their optimal values and the levels of constraint violations are large, especially at the early stage of optimization, the absolute-value penalty function might not be able to impose a sufficiently large penalty.

To overcome the above-mentioned difficulty, our idea is to use a convex piecewise linear function with three segments to approximate the quadratic function. In Figure 1, a quadratic function is depicted in blue, and the absolute-value function is depicted in black. It can be seen that the absolute-value function is not a good approximation to the quadratic function. To improve approximation accuracy, take the following convex piecewise linear function with three segments as an example:

$$p(x) \equiv \max(0, 4x - 3, -4x - 3), \quad (21)$$

which is depicted in red in the figure. As can be seen, with three segments, the approximation accuracy is improved as compared to that of the absolute-value function, in particular for the interval [-4, 4]. For problem instances considered in Section V, the violation of the capacity for a machine group at a time slot is generally less than 4 according to our testing. Therefore, function (21) provides a good approximation to the quadratic function, and imposes sufficiently large penalties when the level of violation is 2 or above as compared to those imposed by the absolute-value function.

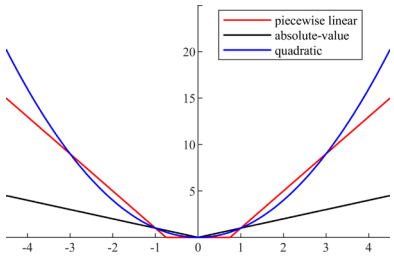

Fig. 1 Illustration of the 3-segment piece-wise linear function, the absolute-value function, and the quadratic function

A convex piecewise linear function such as (21) can be exactly linearized through introducing additional decision variables and constraints following the standard way as discussed on page 150 of [19]. To linearize such a function, one additional decision variable is required, and the number of additional constraints equals the number of non-zero segments. For our 3-segment function (21), the number of non-zero segments is two. Therefore, the numbers of additional decision variables and constraints required are equal to those required by the absolute-value function. In general, the number of segments, the vertices, and the slopes of the function can be adjusted. Nevertheless, more segments do not mean better performance, since more additional constraints are required by linearization.

With the new penalty function (21) replacing the absolute-value penalty function, the solution process follows exactly that of [9]. The relaxed problem becomes:

$$\min_{z,b,s} \left\{ \begin{array}{l} \sum_i w_i \cdot z_i + \sum_t \sum_m \lambda_{t,m}^k \left(g_{t,m}(b) + s_{t,m} \right) + \\ c^k \sum_t \sum_m p\left(g_{t,m}(b) + s_{t,m} \right) \end{array} \right\}, \quad (22)$$

$$s.t. \quad (1), (5), (13).$$

After linearizing the penalty function following page 150 of [19], a subproblem for part i can be formed by optimizing with respect to decision variables associated with that part while fixing decision variables associated with other parts at their previously obtained values.

After solving a subproblem consisting one or a few parts at iteration k by using B&C, multipliers are updated based on surrogate subgradients to coordinate subproblem solutions as:

$$\lambda_{t,m}^{k+1} = \lambda_{t,m}^k + s^k \left(g_{t,m}(b) + s_{t,m} \right), \forall t, m. \quad (23)$$

To guarantee the convergence of multipliers, the stepsize s^k in (23) is updated following (18) and (19) of [9]. If the penalty coefficient c^k is too large, the surrogate optimality condition ((14) in [9]) may not be satisfied, and solutions may get trapped at a local minimum. In this case, c^k is decreased following (21) in [9]. Finally, when the stepsize s^k reduces below a certain threshold or when the CPU time (consists of data and model loading, solving, and solution outputting time) reaches a pre-specified limit, the iterative multiplier updating process stops.

Since machine capacity constraints (7) are relaxed, subproblem solutions, when put together, generally do not satisfy (7). To obtain a solution feasible to the original problem (11), subproblem solutions are "repaired" when the norm squared of constraint violations is less than or equal to a threshold γ, i.e.,

$$\sum_{t,m} \left(g_{t,m}(b) + s_{t,m} \right)^2 \le \gamma. \quad (24)$$

Repairing is done by optimizing the decision variables associated with violated machine capacity constraints while fixing the remaining decision variables by using B&C. This will be done several times to obtain multiple feasible solutions. At the convergence of multipliers, surrogate dual value provides a lower bound to feasible costs, and subproblem solutions are repaired to obtain the last feasible solution. The feasible solution with the minimal cost is then the final solution.

V. Numerical Results

The new formulation and solution methodology are implemented by using MATLAB R2018a and CPLEX 12.8.0.0. Three examples are tested on a laptop with the Intel Xeon W-10855M processor at 4.3-Ghz, 64GB of RAM, and Windows

10. The first example is a small instance considered in [6]–[8], and the second example consists of several medium-sized problems. These two examples are solved by using B&C to numerically demonstrate the advantages of our new formulation. The third example consists of several large instances, and is solved by using our SAVLRE to demonstrate that near-optimal solutions can be efficiently obtained.

Example 1: A small problem

This example consists of an instance with 127 parts, and was taken from Pratt & Whitney's Development Operation shop solved in [6]–[8] to test problem formulations. There are 19 machine groups, each with one to six machines, and for each machine group, the number of machines contained is a constant with respect to time slot t. The problem is solved by using B&C. The solving time, which equals total CPU time minus data and model loading and solution outputting time, was used as the stopping criteria for B&C. It was used in [6] and [8] as well. In this example, optimization stops when the solving time reaches 3,600 seconds, or when the optimal solution is found. The feasible cost, the MIP gap, and the solving time of both the new formulation and the previous formulation (13) are presented in Table I. It can be seen that the new formulation enjoys better solution quality with much reduced computational requirements.

Table I Comparison of formulations: small size

New formulation (B&C)			Formulation [8] (B&C)		
Cost[1]	GAP[2]	Solving time (s)	Cost[1]	GAP[2]	Solving time (s)
14,872	0	3.31	15,117	3.72%	3600

1: Total weighted tardiness
2: MIP gap reported by the CPLEX solver

As reviewed in Section II, the formulation (13) has been improved by using a systematic formulation tightening approach. As reported in Table 2 of [8], after tightening, a feasible cost with 0.01% MIP gap was obtained by using B&C after 14.7s. The results thus fall within the range obtained by using our new formulation.

Example 2: Medium-sized problems

This example consists of six instances with 200 to 400 parts, and 10 to 30 machine types. Each part requires one to seven operations, and each operation can be processed by one to three machine groups. The arrival time of each part is generated by using a uniform distribution $U[1, 300]$, and the due date of each part equals its possible earliest completion time (i.e., its arrival time plus its smallest total processing time). As for tardiness weights, 30% of parts have a weight of 1, 65% of parts have a weight of 10, and 5% of parts have a weight of 100. The total number of time slots is 500, which is large enough to complete all the parts, and satisfies condition (17). The stopping criteria are the same as in Example 1.

Scenarios without machine breakdowns are considered first, where capacities of machine groups are constant over the planning horizon. The feasible costs, the MIP gaps, and the solving times by using B&C are presented in Table II. It can be seen that the new formulation significantly improves the solution quality and drastically reduces computational requirements of B&C as compared to the original formulation

To test the robustness of our new formulation with respect to non-constant capacities of machine groups, each machine is assumed to have a probability of 0.3% to break down at each time slot, and repairing takes four consecutive time slots. Based on this, ten scenarios are randomly generated, and other data are the same as those for the 300-part instance. The optimal solutions of all ten scenarios are efficiently obtained by using B&C. The average solving time is 319s, with the minimal 166s, the maximal 890s, and the standard deviation 214s. The results demonstrate the robustness of our new formulation with respect to non-constant machine capacities.

Table II Comparison of formulations: medium size

Instance	New formulation (B&C)				Formulation [8] (B&C)	
	Cost	GAP	Solving time (s)	LP Cost[3]	Cost	LP Cost[3]
200*20[1]	377	0	12.77	374.41	Fail[2]	0
250*20	593	0	33.91	572.17	Fail	0
300*20	609	0	179.27	558.55	Fail	0
300*10	621	0	156.14	538.47	Fail	0
400*10	1147	0	1395.13	1010.23	Fail	0
300*30	1129	0	188.16	832.01	Fail	0

1: 200*20 means the instance with 200 parts and 20 machine groups
2: No feasible solution is found after 3600s
3: The optimal cost of the corresponding LP-relaxed problem

The optimal costs of the corresponding LP-relaxed problems are also presented in Table II. As can be seen, the optimal costs of the LP-relaxed formulations of our new model are much larger than those of the previous formulation of [8]. This is because the solution set of the LP-relaxed problem of the new formulation is a proper subset of that of the previous formulations (13). Here, the LP cost of the previous formulation (13) is zero. This is because after relaxing integrality constraints, operation statuses $\{\delta_{i,j,m,t}\}$ of (13) can no longer be correctly determined by the "big-M" inequalities (13c) and (13d). For the data set tested, there exist solutions with operation beginning times equal to the earliest possible beginning times while satisfying machine capacity constraints, leading to zero total weighted tardiness.

Example 3: Large problems

This example consists of five instances with 400 to 600 parts, and 10 to 30 machine types. The parts are grouped to form 10 subproblems. Other data are generated following the method of Example 2 without machine breakdowns. The problems are solved by using SAVLRE with the new formulation. When the norm squared of machine capacity violations is less than or equal to 40, i.e., γ in (24) is 40, heuristics are used to find feasible solutions to the original problem. The algorithm stops when the CPU time reaches 1,800s or the stepsize reduces below 0.05. The feasible cost, the gaps, and the CPU times of both SAVLRE and B&C are presented in Table III.

The first instance is with 500 parts and 10 machine groups. By using SAVLRE, a solution with a cost of 4,210 is obtained after 813s, and 3,942 after 1100s. The duality gap is 13.1% and 7.1%, respectively. The results demonstrate that high-quality solutions are efficiently obtained by using SAVLRE. For comparison purposes, B&C is also tested. Since the problem instance is large, no feasible solution is obtained after 3,600s.

To examine the effects of the new penalty function, the norm squared of machine capacity violations at each minor iteration (i.e., after solving one subproblem) is depicted in red in Figure 2. For comparison purposes, those of SAVLR are depicted in blue. As can be seen, the new method enjoys a faster reduction

of the machine capacity violations than SAVLR. This is because our 3-segmnent function (21) provides a good approximation to the quadratic function and sufficiently large penalties. Moreover, since the numbers of decision variables and constraints required to linearize (21) are the same as those required by the absolute-value function, the computational requirements are similar to those of the absolute-value function. The average CPU time per minor iteration is 3.91s for SAVLRE, and 3.52s for SAVLR. Testing of 5- and 7-segment piecewise linear penalty functions has also been conducted, and the results were not satisfactory. This is because linearizing these functions requires additional constraints, resulting in significant increases of subproblem solving times.

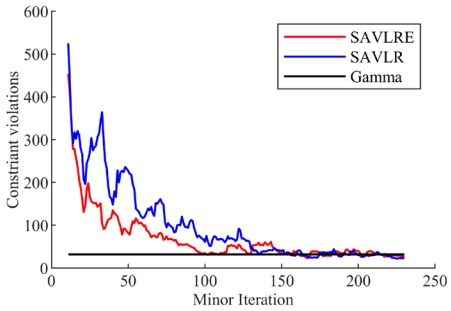

Fig. 2 the norm squared of constraint violations in each minor iteration

For the instance with 600 jobs and 10 machine types, a cost of 7448 with a gap of 5.9% is obtained by SAVLRE after 1261s. A cost of 7503 with a gap of 6.8% is obtained slower by SAVLR after 1403s. To demonstrate the scalability of SAVLRE, three more instances are tested, and the results are shown in Table III.

It is difficult to compare our results with many of the results obtained by using heuristic methods in the literature since their codes and data are generally not available. Nevertheless, let us comment on the results of the "shift bottleneck" methods of [15], [16]. The largest instance considered there has 200 parts and 30 machines, and the corresponding "ratio" (the average gap between feasible costs and their lower bounds) is 10.2%. These results fall within the range obtained by using our method, demonstrating the scalability and quality of our optimization-based method.

Table III Comparison of methodologies: large problems

Instance	New formulation (SAVLRE+B&C)				New formulation (B&C)	
	Cost	Lower Bound	GAP	CPU time (s)	Cost	Solving time (s)
500*10	3942	3659	7.1%	1100	Fail	
600*10	7448	7008	5.9%	1261	Fail	
400*20	2019	1907	5.5%	404	2773 (30.91%)	3600
500*20	3967	3554	10.4%	872	Fail	
500*30	6387	5835	8.6%	843	Fail	

VI. Conclusion

In this paper, a novel ILP formulation is developed for job-shop scheduling. As compared to the previous formulation of [6],[8], the new formulation has much reduced numbers of decision variables and constraints. Moreover, a brand new way to compare the tightness of formulations is developed, and our new formulation is proven to be tighter. Therefore, solution quality is much improved, while the computational requirements are much reduced. SAVLR is also enhanced so that near-optimal solutions can be efficiently obtained for large problems. These advancements will have major implications on formulating and resolution of other manufacturing scheduling problems and beyond. To further improve solution quality and computation efficiency, tightening of the new formulation will be further investigated.

References

[1] M. Pinedo, *Scheduling*, vol. 29. Springer, 2012.

[2] M. Padberg and G. Rinaldi, "A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems," *SIAM review*, vol. 33, no. 1, pp. 60–100, 1991.

[3] R. Gomory, "An algorithm for the mixed integer problem," RAND CORP SANTA MONICA CA, 1960.

[4] R. E. Gomory, "Solving linear programming problems in integers," *Combinatorial Analysis*, vol. 10, pp. 211–215, 1960.

[5] L. A. Wolsey and G. L. Nemhauser, *Integer and combinatorial optimization*, vol. 55. John Wiley & Sons, 1999.

[6] B. Yan, M. A. Bragin, and P. B. Luh, "Novel formulation and resolution of job-shop scheduling problems," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3387–3393, 2018.

[7] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job-shop scheduling problems," *IEEE transactions on Robotics and Automation*, vol. 9, no. 1, pp. 1–13, 1993.

[8] B. Yan, M. Bragin, and P. Luh, "An Innovative Formulation Tightening Approach for Job-Shop Scheduling," 2021, TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.12783893.v2.

[9] M. A. Bragin, P. B. Luh, B. Yan, and X. Sun, "A scalable solution methodology for mixed-integer linear programming problems arising in automation," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 2, pp. 531–541, 2018.

[10] D. P. Bertsekas, *Nonlinear programming*, Third Edit. Athena Scientific, Belmont, MA, 2016.

[11] V. Roshanaei, A. Azab, and H. ElMaraghy, "Mathematical modelling and a meta-heuristic for flexible job shop scheduling," *International Journal of Production Research*, vol. 51, no. 20, pp. 6247–6274, 2013.

[12] M. Karimi-Nasab and M. Modarres, "Lot sizing and job shop scheduling with compressible process times: A cut and branch approach," *Computers & Industrial Engineering*, vol. 85, pp. 196–205, 2015.

[13] S. Chansombat, P. Pongcharoen, and C. Hicks, "A mixed-integer linear programming model for integrated production and preventive maintenance scheduling in the capital goods industry," *International Journal of Production Research*, vol. 57, no. 1, pp. 61–82, 2019.

[14] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem," *Computers & Industrial Engineering*, vol. 142, p. 106347, 2020.

[15] Z. Yan, G. Hanyu, and X. Yugeng, "Modified bottleneck-based heuristic for large-scale job-shop scheduling problems with a single bottleneck," *Journal of Systems Engineering and Electronics*, vol. 18, no. 3, pp. 556–565, 2007.

[16] R. Uzsoy and C.-S. Wang, "Performance of decomposition procedures for job shop scheduling problems with bottleneck machines," *International Journal of Production Research*, vol. 38, no. 6, pp. 1271–1286, 2000.

[17] M. Saidi-Mehrabad and P. Fattahi, "Flexible job shop scheduling with tabu search algorithms," *The international journal of Advanced Manufacturing technology*, vol. 32, no. 5–6, pp. 563–570, 2007.

[18] C. Liu, M. Shahidehpour, and J. Wang, "Application of augmented Lagrangian relaxation to coordinated scheduling of interdependent hydrothermal power and natural gas systems," *IET generation, transmission & distribution*, vol. 4, no. 12, pp. 1314–1325, 2010.

[19] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.