An Innovative Formulation Tightening Approach for Job-Shop Scheduling

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

Abstract— Job shops are an important production environment for low-volume high-variety manufacturing. Its scheduling has recently been formulated as an integer linear programming (ILP) problem to take advantages of popular mixed-integer linear programming (MILP) methods, e.g., branch-and-cut. When considering a large number of parts, MILP methods may experience difficulties. To address this, a critical but much overlooked issue is formulation tightening. The idea is that if problem constraints can be transformed to directly delineate the problem convex hull in the data preprocessing stage, then a solution can be obtained by using linear programming (LP) methods without combinatorial difficulties. The tightening process, however, is fundamentally challenging because of the existence of integer variables. In this article, an innovative and systematic approach is established for the first time to tighten the formulations of individual parts, each with multiple operations, in the data preprocessing stage. It is a major advancement of our previous work on problems with binary and continuous variables to integer variables. The idea is to first link integer variables to binary variables by innovatively combining constraints so that the integer variables are uniquely determined by the binary variables. With binary and continuous variables only, it is proved that the vertices of the convex hull can be obtained based on vertices of the LP problem after relaxing binary requirements. These vertices are then converted to tightened constraints for general use. This approach significantly improves our previous results on tightening individual operations. Numerical results demonstrate significant benefits on solution quality and computational efficiency. This approach also applies to other complex ILP and MILP problems with similar characteristics and fundamentally changes the way how such problems are formulated and solved.

Note to Practitioners—Scheduling is an important but difficult problem in planning and operation of job shops. The problem has been recently formulated in an integer linear programming (ILP) form to take advantage of popular mixed-integer linear programming methods. Given an ILP problem, there must exist a linear programming (LP) formulation so that all of its vertices are also the vertices to the ILP problem. If such an LP problem can be found in the data preprocessing stage, then the corresponding ILP problem is tight and can be solved by using an LP method

without difficulties. In this article, an innovative and systematic approach is established to tighten the formulations of individual parts, each with one or multiple operations. It is a major advancement of our previous work on problems with binary and continuous variables by novel exploitation of the relationship between integer and binary variables in job-shop scheduling. The resulting tightened constraints are characterized by part parameters and the length of the scheduling horizon and can be easily adjusted for other data sets. Results demonstrate significant benefits on solution quality and computational efficiency. This approach also applies to other complex ILP and MILP problems with similar characteristics and fundamentally changes the way how such problems are formulated and solved.

Index Terms— Formulation tightening, job-shop scheduling, manufacturing, mixed-integer linear programming (MILP).

I. Introduction

JOB shops are an important production environment for low-volume high-variety manufacturing. In a job shop, machines are usually categorized into different types based on their functions. With these machines, multiple parts with different due dates are processed, and each part needs a sequence of operations to be completed [1]. To meet on-time deliveries, scheduling of parts is critical. The problem is to minimize the required objective, e.g., the total weighted tardiness and the total cycle time, by assigning parts to machines while satisfying part processing time requirements, and operation precedence and machine capacity constraints. It is one of the hardest scheduling problems. Only a few special cases are polynomially solvable, such as two machines or two parts, and slightly generalized versions of these problems are NP-hard [2]. For practical-sized job-shop problems, the optimal solution is difficult to get and the goal is usually to obtain near-optimal solutions with quantifiable quality.

As reviewed in Section II, some nonlinear job-shop scheduling formulations were established and efficiently exploited by decomposition and coordination methods. To take advantage of popular mixed-integer linear programming (MILP) methods, e.g., branch-and-cut, the problem is recently formulated in an integer linear programming (ILP) form and is not convex[1] with the existence of integer variables. Branch-and-cut first solves the linear programming (LP) problem without integrality requirements. If the solution is feasible to the original MILP problem, then it is optimal. If not, valid cuts are performed around the optimal solution of the LP problem on the fly to get

[1]A set is convex if the line segment between any two points in it lies in it. A function defined on a convex set is convex if the line segment in-between any two points on the graph of the function lies above the graph between the two points.

solutions to the MILP problem. If such solutions are obtained, the problem is directly solved. If not, the method relies on time-consuming branching operations.

When considering a large number of parts, the state-of-the-art and practice MILP methods may experience convergence and quality difficulties. To obtain near-optimal job-shop schedules fast, a critical but much overlooked issue is formulation transformation. The idea is to transform problem constraints to directly delineate the convex hull (the smallest convex set that contains all feasible solutions [3]) in the data preprocessing stage. If this can be done (i.e., the formulation is "tight"), then a solution can be obtained by using an LP method without combinatorial difficulties. Theoretically, the tighter the formulation, the less the time to obtain solutions with the same quality. The tightening process, however, is fundamentally challenging without a systematic approach because of the existence of integer variables (e.g., beginning time).

In the literature, a few tightened single-part formulations were reported and shown computationally efficient for the overall problems. In our previous work [4], [5], a systematic formulation tightening approach was developed for mixed-binary linear programming (MBLP) for the first time, and the method was realized based on power system unit comment problems. The idea is to derive vertices of the convex hull of a unit (a generator) without binary requirements. From them, vertices of the original convex hull are innovatively obtained. These vertices are converted to tightened constraints, which are then parameterized based on unit characteristics for general use in unit commitment problems. The method was then extended to job-shop scheduling problems with integer variables in [6]. Our idea is to obtain the vertices of the convex hull of a part without integrality requirements and then approximate noninteger values in vertices as nearest feasible integers. For each operation, two sets of tightened constraints related to processing time were obtained. They were then parameterized by analyzing their patterns for general use in job-shop scheduling problems. Precedence constraints were modeled but not tightened. The results were then extended to consider energy costs as a component of the objective function in energy-efficient job-shop scheduling problems in [7]. Results in [4]–[7] demonstrate computational efficiency and solution quality benefits of formulation tightening.

In this article, the job-shop scheduling problem is first formulated in an integer programming form in Section III following [6], and the objective is to minimize the total weighted tardiness and the total cycle time. Since tardiness is a nonlinear function of part completion times, it is linearized by introducing new binary and continuous variables and the corresponding constraints to make effective use of MILP methods. Then, the problem becomes an MILP problem. Since the complexity of tightening increases with problem sizes, the focus is on individual parts.

To tighten the formulations of parts with multiple operations in the data preprocessing stage, an innovative and systematic approach is established for the first time in Section IV. This is a major advancement of our previous work as it is generalized from MBLP to MILP problems with special structures. The idea is to first link integer variables (e.g., beginning time) to binary variables (e.g., part statues) by innovatively combining constraints so that the integer variables are uniquely determined by the binary variables. With binary and continuous variables only, it is proved that the vertices of the convex hull can be obtained based on vertices of the LP problem after relaxing binary requirements [4], [5]. Thus, there is no approximation in the tightening process as those in [6] and [7]. These vertices are then converted to tightened constraints. The numbers of resulting tightened constraints and variables involved, and the constraint coefficients depend on part parameters. Since all parts must be processed within the scheduling horizon, the above also depends on the length of the horizon. For general use purposes, these tightened constraints are characterized by analyzing constraint structures and the relationship between coefficients and part parameters as well as the scheduling horizon.

Since it is difficult to directly tighten the formulation with multiple operations, the idea is to first tighten the formulation of a single operation to explore relations among part status and beginning/completion time. The resulting processing time and beginning/completion time-related tightened constraints can be applied to every operation of parts with multiple operations. Then, the same method is used to tighten the formulation of two successive operations to explore their interactions. The resulting precedence-related tightened constraints can be used for every two consecutive operations of parts with multiple operations. The process can be repeated for the formulation with three and more operations. The tightening process only needs to be performed once, and the resulting tightened constraints can be easily adjusted for other data sets after parameterization and can be directly applied in the data preprocessing stage, tremendously reducing online computational requirements. This approach significantly improves our previous results on tightening individual operations [6].

Three examples are considered in Section V. The first is to tighten the formulations for single parts to illustrate the tightening idea and present insights. Robustness of formulation tightening is shown in the second example. The last example is to demonstrate the performance of tightened single-part formulations when solving the overall job-shop scheduling problems. Results demonstrate significant benefits on solution quality and computational efficiency.

Beyond MILP job-shop scheduling problems under consideration, this approach also applies to the other complex ILP and MILP problems with similar characteristics between integer and binary variables. It fundamentally changes the way how such problems are formulated and solved. This approach goes naturally with decomposition and coordination approaches, a subject worthy of further exploration.

II. Literature Review

Existing job-shop formulations and solution methodologies are reviewed in Section II-A. Tightened constraints are reviewed in Section II-B.

A. Problem Formulations and Solution Methodologies

With large numbers of decision variables and constraints in job-shop scheduling, developing efficient formulations is complex [8]. "Separable" and nonlinear formulations were established and efficiently exploited by decomposition and coordination-based Lagrangian relaxation methods in [9]–[14]. ILP models were also developed in [15]–[25]. Considering sequence-dependent setups, an ILP model was established in [15]. With additional variables, job successors and predecessors were modeled. In our previous work on high-volume and low-variety manufacturing [25], an ILP model was developed. However, large-scale problems cannot be effectively solved by using those models.

To solve job-shop scheduling problems, branch-and-cut has been widely used. The method solves the LP problem without integral requirements by using an LP method first. If the solution has integer values for all integer decision variables, it is optimal with respect to the original problem. If not, the method tries to obtain the convex hull by adding valid cuts to cut off regions outside the convex hull without cutting off feasible solutions. If successful, the problem is directly solved. If not, time-consuming branching operations are performed, resulting in very slow convergence. In [15] and [17]–[25], the problems were solved by branch-and-cut implemented in commercial software CPLEX or Gurobi. When considering a large number of parts, branch-and-cut may experience convergence and quality difficulties.

B. Tightened Constraints

Obtaining a tight formulation is fundamentally difficult and NP-hard without clear ways. In the literature, few tightening studies exist on general problems. For traveling salesman problems, a tightened formulation was obtained based on subtour elimination [26]. For knapsack problems, tight formulations were obtained through the use of "structural" disjunctive cuts based on the problem structure [27].

For manufacturing scheduling, a few tightened constraints were presented for single parts without explaining how they were generated. For traditional job-shop scheduling, a few valid cuts were developed by analyzing problem structures in [17]. The major idea is to find a ceiling for inventory shortage and the longest working procedure sequence till completion for parts. Testing results based on randomly generated data for 325 instances with 3–5 machines and 4–6 parts demonstrate the computational efficiency of the cuts. For flow-shop scheduling, subtour elimination constraints and lower/upper bound mixed-integer inequalities were developed by analyzing formulation structures in [15], and some of them are facet-defining cuts. Testing results based on randomly generated data for problems with 2–6 machines and 7–10 parts show that the computational time is much reduced with these tightened constraints. For both studies, testing results demonstrate the computational efficiency of these tightened constraints.

In our previous work [4], [5], a systematic formulation tightening approach was developed for MBLP problems for the first time, and the method was realized based on power system unit comment problems. It is based on novel integration of "constraint-and-vertex conversion," "vertex elimination," and "parameterization" processes to tighten single-unit formulations. Our idea is to derive vertices of the convex hull of a unit (a generator) without binary requirements. From them, vertices of the original convex hull are innovatively obtained by eliminating vertices with fractional values for binary variables with rigorous proof. These vertices are converted to tightened constraints, which are then parameterized based on unit characteristics for general use in unit commitment problems. The method was then extended to job-shop scheduling problems with integer variables in [6]. Our idea is to obtain the vertices of the convex hull of a part without integrality requirements and then approximate noninteger values in vertices as nearest feasible integers. For each operation, two sets of tightened constraints related to processing time were obtained. They were then parameterized by analyzing their patterns for general use in job-shop scheduling problems. Precedence constraints were modeled but not tightened. The results were then extended to consider energy costs as a component of the objective function in energy-efficient job-shop scheduling problems in [7]. Results in [4]–[7] demonstrate computational efficiency and solution quality benefits of formulation tightening.

III. Job-Shop Scheduling Formulation

Consider a job shop with multiple machines categorized into M types based on their functionalities. By using these machines, I parts with different due dates need to be processed, and the part index is i. Part i requires J_i operations, and the operation index is j. It is assumed that the scheduling horizon is long enough so that all parts can be processed. The horizon is discretized into K time slots and let k denote the time index. Assuming that system-level machine capacity constraints are relaxed, a single-part scheduling problem is formulated based on our previous work [6] in Section III-A (part index i is omitted for brevity). Machine capacity constraints and the objective function are briefly described in Section III-B.

A. Single-Part Formulation

For a part with J operations, the main decision variables are beginning time b_j and completion time c_j for each operation j, both integer variables. To capture the status of operation j at time k, i.e., active (processed) or not, binary variables δ_{jmk} with operation j, machine group m, and time k indices are considered

$$\delta_{jmk} = \begin{cases} 1, & \text{if operation } j \text{ is active on machine type } m \text{ at time } k \\ 0, & \text{otherwise.} \end{cases}$$

Here, the machine group concept is considered instead of individual machines since machines are usually categorized into different types based on their functions in a job shop. With machine groups, the decision space is much reduced, and the problem complexity is thus much reduced. The choice of machines within the same group will be based on heuristics after optimization and is not captured in the formulation.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

4   IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING

Since the processing time p_{jm} is machine-type-dependent, a new set of binary decision variables is defined to identify the assignment of operation j to machine type m as follows:

$$x_{jm} = \begin{cases} 1, & \text{if operation } j \text{ is assigned to machine type } m \\ 0, & \text{otherwise.} \end{cases}$$

Part-level constraints are processing time requirements, operation precedence constraints, and machine-type assignment constraints. Modeling of linearized tardiness is also included.

1) Processing Time Requirements: Because of the "non-preemption," a contiguous time period with length of p_{jm} is needed to process operation j, i.e.,

$$c_j = b_j + \sum_{m \in M_j} x_{jm} p_{jm} - 1, \quad \forall j \tag{1}$$

where M_j denotes the set of machine types that can process operation j of part i (part index i is omitted). Since processing time p_{jm} is generally machine-type-dependent, the actual processing time depends on the assignment of machine types.

Since δ_{jmk} represents the status of the part, δ_{jmk} must be 1 within $[b_j, c_j]$ if machine type m is assigned to process operation j and 0 otherwise, i.e.,

$$\delta_{jmk} = \begin{cases} 1, & \text{if } b_j \le k \le c_j \text{ and } x_{jm} = 1 \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Logical (2) is linearized as follows:

$$k \le c_j + N\left(1 - \sum_{m \in M_j} \delta_{jmk}\right), \quad \forall j, \forall k \tag{3}$$

$$k \ge b_j - N\left(1 - \sum_{m \in M_j} \delta_{jmk}\right), \quad \forall j, \forall k \tag{4}$$

$$\sum_k \delta_{jmk} = x_{jm} p_{jm}, \quad \forall j, \forall m \tag{5}$$

where N is a big number. It can be seen that (3)–(5) guarantee that $\delta_{jmk} = 1$ iff $b_j \le k \le c_j$ and $x_{jm} = 1$, and $\delta_{jmk} = 0$ otherwise.

2) Operation Precedence Constraints: It is assumed that the operation sequence of a part is fixed, and operation $j + 1$ cannot start until j is finished, i.e.,

$$b_{j+1} \ge c_j + 1, \quad \forall j. \tag{6}$$

Also, the part cannot start the process of the first operation until it arrives at time a, i.e.,

$$b_1 \ge a. \tag{7}$$

3) Machine-Type Assignment Constraints: Operation j can only be assigned to one machine type, i.e.,

$$\sum_{m \in M_j} x_{jm} = 1, \quad \forall j. \tag{8}$$

Also, if a machine type cannot process operation j of part i, then the assignment variable should be 0, i.e.,

$$x_{jm} = 0, \quad \forall j, m \notin M_j. \tag{9}$$

4) Linearized Tardiness: Tardiness T is formulated as follows:

$$\max(c_J - d, 0) \tag{10}$$

where d is the due date. To represent this, a piecewise linear function is used, as shown in Fig. 1.

As shown in the figure, the lower and upper bounds of $c_J - d$ are $\sum_j \min_{m \in M_j} p_{jm} - d$ and $K - d$, and the corresponding tardiness is 0 and $K - d$. The three break points of this function on the x-axis are $\sum_j \min_{m \in M_j} p_{jm} - d$, 0, and $K - d$ (if $\sum_j \min_{m \in M_j} p_{jm} - d < 0 < K - d$), and the corresponding values at the y-axis are 0, 0, and $K - d$. This piecewise linear function is linearized by using special ordered set techniques [28]. Three continuous variables w^1, w^2, and w^3 ($0 \le w^1, w^2, w^3 \le 1$) are considered to represent the weights of the three points. In addition, three binary variables α^1, α^2, and α^3 are used to set up upper bounds for these weights. The constraints are as follows:

$$c_J - d = \left(\sum_j \min_{m \in M_j} p_{jm} - d\right) w^1 + 0 w^2 + (K - d) w^3 \tag{11}$$

$$T = o w^1 + 0 w^2 + (K - d) w^3 \tag{12}$$

$$\alpha^l \ge w^l, \quad 1 \le l \le 3 \tag{13}$$

$$\alpha^1 + \alpha^3 \le 1 \tag{14}$$

$$\sum_l w^l = 1 \tag{15}$$

$$\sum_l \alpha^l = 2. \tag{16}$$

For simplicity, instead of $\sum_j \min_{m \in M_j} p_{jm} - d$ and $K - d$, two break points $-K$ and $2K$ are used for all parts (d could be negative). Here and later in the article, the max function is kept for compactness of notation.

B. Machine Capacity Constraints and Objective Function

For completeness, machine capacity constraints and the objective function are briefly described in this section.

1) Machine Capacity Constraints: For each machine type m, the total number of active parts cannot exceed machine capacity M_m at any time slot, i.e.,

$$\sum_{\forall (i,j) \in O_m} \delta_{ijmk} \le M_m, \quad \forall m, \forall k. \tag{17}$$

In the above, (i, j) denotes operation j of part i, and O_m denotes the set of (i, j) that can be processed by machine type m.

2) Objective Function: The objective function is to minimize the weighted sum of total tardiness and total cycle time, i.e.,

$$\omega \sum_i \left(\omega_i^T \max(c_{iJ_i} - d_i, 0)\right) + (1 - \omega) \sum_i \left(c_{iJ_i} - a_{i,1}\right) \tag{18}$$

where ω is the weight for total tardiness and ω_i^T is for part i.

The job-shop scheduling problem with (1), (3)–(9), and (11)–(18) established above is an MILP problem. Most of the

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

YAN *et al.*: INNOVATIVE FORMULATION TIGHTENING APPROACH FOR JOB-SHOP SCHEDULING
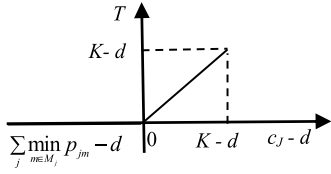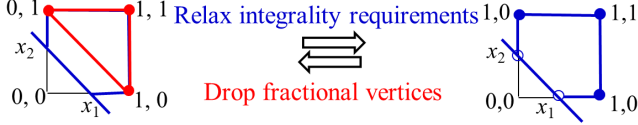
5

Fig. 1.   Tardiness function.

Fig. 2.   (a) Convex hull of a BLP problem with binary variables x_1 and x_2. (b) Convex hull of its integer-relaxed problem.

Fig. 3.   Flowchart of formulation tightening.

decision variables are binary (e.g., δ and x). There are also a few integers (e.g., b and c) and continuous variables (i.e., w). If every operation of each part can only be processed on one machine type, then there is no need to consider machine-type assignment variable x, and machine-type index m for part status variable δ can be deleted. The machine capacity constraints and objective function are linear but irrelevant for tightening.

IV. FORMULATION TIGHTENING

Building upon our previous work [4]–[7], an innovative and systematic method is established to tighten the above single-part formulation in Section IV-A. A numerical example is also presented to illustrate the tightening idea. Tightness is proved in Section IV-B.

A. Formulation Tightening

In our previous work on power system unit comment problems, a systematic approach is developed to tighten MBLP problems [4], [5]. To illustrate the idea, consider a simple binary linear programming (BLP) problem in Fig. 2 with two binary variables x_1 and x_2, and $x_1 + x_2 \geq 0.5$. After relaxing binary requirements, the vertices [blue dots in Fig. 2(b)] of the convex hull [blue lines in Fig. 2(b)] to the integer-relaxed problem are obtained. Then, the vertices [red dots in Fig. 2(a)] of the original convex hull [red lines in Fig. 2(a)] can be obtained by simply eliminating the vertices with factional values [open blue dots in Fig. 2(b)] [4], [5]. These vertices are then converted to tightened constraints for general use. The idea to tighten MBLP problems is the same.

For ease of presentation, the following terms are defined.

Definition 1: For an MBLP problem, if the integrality requirements are relaxed, the resulting convex hull is defined as the "integer-relaxed convex hull." In terms of the simple example described above, the integer-relaxed convex hulls are defined by blue lines in Fig. 2(b).

Definition 2: For an integer-relaxed convex hull of an MBLP problem, a vertex consists of integral and real components. If all integral components have integer values, then it is called an "integral vertex." Otherwise, it is called "fractional vertex." In terms of the simple example above, integral and fractional vertices are denoted by solid and open blue dots, respectively, in Fig. 2(b).

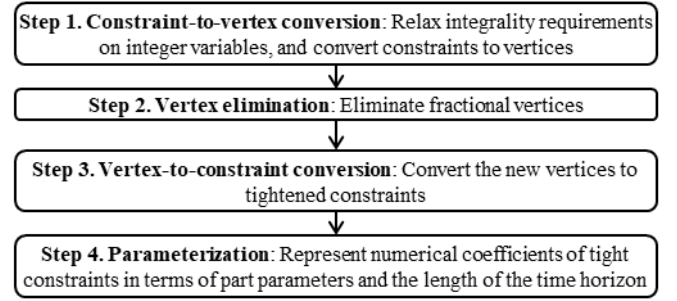The above definitions can apply to an MILP problem.

To apply the MBLP tightening idea to tighten the MILP problem under consideration, the relationship that integer variables (e.g., beginning time b) are uniquely determined by binary variables (e.g., part status δ and machine-type assignment variable x) is innovatively established as to be proved in Section IV-B. Therefore, the MBLP principle of eliminating fractional vertices with respect to δ and x described above can be applied.

Since it is difficult to directly tighten the formulation with multiple operations, the idea is to first tighten the formulation of a single operation to explore relations among part status and beginning/completion time. The resulting processing time and beginning/completion time-related tightened constraints can be applied to every operation of parts with multiple operations. Then, the same method is used to tighten the formulation of two successive operations to explore their interactions. The resulting precedence-related tightened constraints can be used for every two consecutive operations of parts with multiple operations. The process can be repeated for the formulation with three and more operations.

1) One Operation: Given part parameters (due date d, processing time p, and arrival time a) and the length of the scheduling horizon (K) in numerical values, tightened constraints are established by an innovative and systematic method through four steps, as shown in Fig. 3.

Step 1 (Constraint-to-Vertex Conversion): After relaxing integrality requirements, the constraints are converted to the vertices of the integer-relaxed convex hull. The conversion is done by algebraic manipulation of part parameters and the scheduling horizon length with algorithms [29] well established in existing software Porta [30]. The constraints are input, and the output is vertices in numerical values.

Step 2 (Vertex Elimination): If all vertices obtained in Step 1 are integral, the formulation is tight. If not, fractional vertices are projected onto the original convex hull. For this particular problem, all integral vertices of the integer-relaxed convex hull are the same as the vertices of the original convex hull and vice versa, as will be proved in Section IV-B. Thus, vertex projection can be done by eliminating factional vertices.

Step 3 (Vertex-to-Constraint Conversion): In this step, the vertices obtained in Step 2 are converted back to tightened constraints by using Porta as a reverse process of that in Step 1. The resulting formulation with those constraints should be tight.

Step 4 (Parameterization): Constraints obtained above have coefficients in numerical values. To make them reusable for

```
c-b+1=3
δ₁+δ₂+δ₃+δ₄+δ₅+δ₆+δ₇+δ₈=3
c+8-8δ₁>=1
b-8+8δ₁<=1
c+8-8δ₂>=2
b-8+8δ₂<=2
c+8-8δ₃>=3
b-8+8δ₃<=3
c+8-8δ₄>=4
b-8+8δ₄<=4
c+8-8δ₅>=5
b-8+8δ₅<=5
c+8-8δ₆>=6
b-8+8δ₆<=6
c+8-8δ₇>=7
b-8+8δ₇<=7
c+8-8δ₈>=8
b-8+8δ₈<=8
```

Fig. 4. Original constraints of a single operation problem.

(1225)	1	3	1	1	0	7/8	1/8	0	0	0
(1226)	1	3	1	1	1/8	0	0	0	1/2	3/8
(1227)	1	3	1	1	1/8	7/8	0	0	0	0
(1228)	1	3	1	1	1/4	0	3/4	0	0	0
(1229)	1	3	1	1	3/8	0	0	5/8	0	0
(1230)	1	3	1	1	1/2	0	0	0	1/2	0
(1231)	1	3	1	1	5/8	0	0	0	0	3/8
(1232)	3	5	0	0	1	1	1	0	0	0
(1233)	2	4	0	1	1	1	0	0	0	0
(1234)	1	3	1	1	1	0	0	0	0	0

Fig. 5. Vertices of the LP problem.

(1)	1	3	1	1	1	0	0	0	0	0
(2)	2	4	0	1	1	1	0	0	0	0
(3)	3	5	0	0	1	1	1	0	0	0
(4)	4	6	0	0	0	1	1	1	0	0
(5)	5	7	0	0	0	0	1	1	1	0
(6)	6	8	0	0	0	0	0	1	1	1

Fig. 6. Integral vertices.

other parts, the idea is to convert numerical coefficients to part parameters (e.g., processing time) and the total number of time slots in the scheduling horizon. This parameterization is done by analyzing constraints and the relationship between numerical coefficients and part parameters and the scheduling horizon length. It is verified by checking the physical meanings of the resulting constraints with coefficients in part parameters and the scheduling horizon length under all possible combinations of binary variables. The resulting tightened constraints can be easily adjusted for problems with other data sets.

For a single part, the number of tightened constraints, the number of variables involved, and constraint coefficients depend on part parameters and the length of the scheduling horizon. For example, consider the first operation of a part with $p = 3$ and $K = 5$, and assume that this operation can only be processed on one machine type. Because of "nonpreemption," a contiguous time period with length of 3 is needed to process this operation. If the first time block is taken, then the contiguous time period cannot go beyond time block 3; otherwise, the process is disjunctive. Therefore, $\delta_1 + \delta_4 \leq 1$ and $\delta_2 + \delta_5 \leq 1$. Since the assumption is that the scheduling horizon is long enough so that the operation can be processed, $\delta_1 + \delta_4 = 1$ and $\delta_2 + \delta_5 = 1$. For the same part with $K = 6$, there is one more similar constraint. Note that after parameterization, the resulting tightened constraints can be used for individual operations of parts with multiple operations.

Numerical Example: To illustrate the above approach, a numerical example is presented. Consider the first operation of a prat with $p = 3$ and $K = 8$, and assume that this operation can only be processed on one machine type. Decision variables include part status δ_k, beginning time b, and completion time c. Constraints are processing time requirements (1) and (3)–(5). Without integrality requirements, the constraints to Porta are shown in Fig. 4.

By constraint-to-vertex conversion, 1234 vertices are obtained and the last ten are shown in Fig. 5. Six integral vertices remain after eliminating factional vertices, as shown in Fig. 6. By vertex-to-constraint conversion, tight constraints are generated by Porta in Fig. 7.

Equalities (2), (3), and (5) in Fig. 7 are converted to a set of processing time-related tightened constraints as follows:

$$\delta_1 + \delta_4 + \delta_7 = 1 \qquad (19a)$$
$$\delta_2 + \delta_5 + \delta_8 = 1 \qquad (19b)$$
$$\delta_3 + \delta_6 = 1. \qquad (19c)$$

Because of "nonpreemption," a contiguous time period with length of 3 is needed to process this operation. If the first time block is taken, then the contiguous time period cannot go beyond time block 3, and otherwise, the process is disjunctive. Therefore, $\delta_1 + \delta_4 + \delta_7 \leq 1$. Since it is assumed that the scheduling horizon is long enough so that the part can be processed, $\delta_1 + \delta_4 + \delta_7 = 1$ as shown in (19a). Similarly, one δ from time slots 2, 5, and 8 must be 1 as shown in (19b), and one δ from time slots 3 and 6 must be 1 as shown in (19c). Given (19) and binary requirements of δ, inequalities (11) and (12) in Fig. 7 are redundant. The processing time-related tightened constraint set has been reported in our previous work [6].

The above set of tightened constraints can be generalized for all operations with different processing time as follows:

$$\sum_{\tau=0}^{\tau=\lfloor K/p_{jm} \rfloor : p_{jm}\tau+k \leq K} \delta_{jm,k+p_{jm}\tau} = x_{jm}, \quad m \in M_j, \ k \in [1, p_{jm}]. \qquad (20)$$

2) Two Operations: Now, consider two operations. Given part parameters (due date d, processing time p_1 and p_2, and arrival time a) and the scheduling horizon length in numerical values, tightened constraints are established as follows.

For the first and second operations, they have their own constraints, such as processing time requirements. There is also an operation precedence constraint that couples the two operations together. Denote the operation-level constraints for the first and second operations as C_1 and C_2, respectively, and the coupling constraint as C_{1-2}. Apply the tightened constraints obtained by tightening the single operation formulation to C_1 and C_2, and obtain TC_1 and TC_2, respectively. With the constraint set $\{TC_1, TC_2, C_{1-2}\}$, tighten the two-operation formulation through the four steps presented above and obtain tightened constraints across two operations as TC_{1-2}.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

YAN *et al.*: INNOVATIVE FORMULATION TIGHTENING APPROACH FOR JOB-SHOP SCHEDULING

7

Fig. 7.   Tightened constraints.

Note that after parameterization, TC_{1-2} can be used for every two consecutive operations of parts with multiple operations.

Similar to the tightened constraints for every operation, the tightened constraints across two operations also depend on part parameters and the length of the scheduling horizon. For example, consider a part with $p_1 = 3$ and $p_2 = 1$, and $K = 5$. Because the part must be processed in the scheduling horizon, the latest completion time of operation one is 4 as operation two needs one time slot after it, and thus, $\delta_{1,5} = 0$. Similarly, the earliest beginning time of operation two is 4 as operation one needs three time slots before it, and thus, $\delta_{2,1} = \delta_{2,2} = \delta_{2,3} = 0$.

3) Multiple Operations: With tightened constraints for individual operations and every two consecutive operations, the tightening process is repeated for parts with more operations. Since the number of vertices increases exponentially in constraint-and-vertex conversion and so does the number of constraints, it is difficult to obtain a tight formulation. Our goal is thus to obtain "near-tight" formulations by partially tightening.

B. Tightness Proof

Tightness proof is established in Theorem 1.

Theorem 1: For the formulation of a single operation described by (1), (3)–(5), (8), (9), and (11)–(16), the integral vertices (Definition 2) of its integer-relaxed convex hull (Definition 1) $\mathrm{Conv}(P_{\mathrm{MILP-IR}})$ are the vertices of the convex hull $\mathrm{Conv}(P_{\mathrm{MILP}})$ of the original problem and vice versa.

Proof: The proof will be conducted in two steps. The major step is to show that the values of integer decision variables can be uniquely determined by the values of binary variables. The remaining step is to prove that integral vertices of the integer-relaxed convex hull $\mathrm{Conv}(P_{\mathrm{MILP-IR}})$ are the vertices of the original convex hull $\mathrm{Conv}(P_{\mathrm{MILP}})$ based on the theorems developed for MBLP problems in our previous work [5].

1) Step 1—Integer Variables Can Be Uniquely Determined by Binary Variables: An integral vertex of the integer-relaxed convex hull is feasible to the original MILP problem. Now, consider one operation. Since "nonpreemptive" processing time requirements modeled in (3)–(5) are satisfied, a contiguous time period of length $\sum_{m \in M} x_m p_m$ (operation index j is omitted for brevity, and it is assigned to one machine type in set M that can process this operation) should be assigned to process this operation. For any time k_0 such that $1 \le k_0 \le K$

$- \sum_{m \in M} x_m p_m + 1$, without loss of generality, it is assumed that this operation is processed during the time interval [k_0, $k_0 + \sum_{m \in M} x_m p_m - 1$]. Because of (5), $\sum_{m \in M} x_m p_m$ time slots will be occupied, and because of (3) and (4), these time slots will be contiguous. Since the operation is processed during time slot $k \in [k_0, k_0 + \sum_{m \in M} x_m p_m - 1]$, then $\sum_{m \in M} \delta_{mk}$ equals 1 during these time slots and 0 otherwise as required by the processing time requirements (5). Therefore, the terms with big number N in (3) and (4) disappear. By replacing c as $b + \sum_{m \in M} x_m p_m - 1$ in (3) and combining (3) and (4), the following inequality is obtained:

$$b \le k \le b + \sum_{m \in M} x_m p_m - 1. \qquad (21)$$

To analyze the relationship between b and k_0, (21) is evaluated at the two extreme points of k in [k_0, $k_0 + \sum_{m \in M} x_m p_m - 1$]. First, set k as k_0, and (21) becomes

$$b \le k_0 \le b + \sum_{m \in M} x_m p_m - 1. \qquad (22)$$

Then, set k as $k_0 + \sum_{m \in M} x_m p_m - 1$, and (21) becomes

$$b \le k_0 + \sum_{m \in M} x_m p_m - 1 \le b + \sum_{m \in M} x_m p_m - 1. \qquad (23)$$

Equation (23) can be rewritten as

$$b - \sum_{m \in M} x_m p_m + 1 \le k_0 \le b. \qquad (24)$$

Combining (22) and (24), obtain $k_0 \le b \le k_0$, which implies that $b = k_0$. Then, c can be described as $k_0 + \sum_{m \in M} x_m p_m - 1$. Therefore, the values of δ and x uniquely determine the values of b and c.

The above implies that when δ and x are binary and (3)–(5) are all satisfied, the values of integer decision variables b and c can be uniquely determined by the values of binary decision variables δ and x. Therefore, given a vertex of integer-relaxed convex hull $\mathrm{Conv}(P_{\mathrm{MILP-IR}})$, if all the binary variables have binary values, then the integer variables have integral values. Thus, the MILP problem under consideration can be treated as an MBLP problem in the tightening process.

2) Step 2—Tightness of MILP Problems: For an MBLP problem, it has been proved that the integral vertices of its integer-relaxed convex hull are all vertices of the original convex hull in our previous work [5] and vice versa. Since the MILP problem under consideration can be treated as an MBLP problem in the tightening process, integral vertices of its integer-relaxed convex hull $\mathrm{Conv}(P_{\mathrm{MILP-IR}})$ are the vertices of original convex hull $\mathrm{Conv}(P_{\mathrm{MILP}})$ and vice versa.

End: Based on Theorem 1, vertex projection can be simply done by eliminating factional vertices in Step 2 to tighten the single operation formulation. For parts with multiple operations, since the relations between b, δ, and x within individual operations still hold, the values of b and c can be uniquely determined by the values of δ and x. Thus, the formulation is still tight by applying the same idea as that for the single operation.

Generalization: Beyond MILP job-shop scheduling problems under consideration, this approach also applies to other

complex ILP and MILP problems with similar characteristics between integer and binary variables.

V. NUMERICAL RESULTS

The above tightening method is implemented by using Porta [30]. The job-shop scheduling problems are solved by using IBM ILOG CPLEX Optimization Studio V 12.8.0.0 [31] on a PC with 2.40-GHz Intel Xeon E-2286M CPU and 32-GB RAM. Three examples are presented. The first is to tighten the formulations of single parts to illustrate the idea and present the insights. Robustness of formulation tightening is shown in the second example. The last example is to demonstrate performance of tightened single-part formulations when solving the overall problems.

A. Example 1: Single Part

1) One Operation: Consider the scheduling problem for the first operation of a prat with $p = 3$ and $K = 8$ used in Section IV-A. Constraints (2), (3), and (5) in Fig. 7 have been explored in Section IV-A, and inequalities (11) and (12) are redundant given the binary requirements of δ. Therefore, the focus is on exploring inequalities (6)–(10) and equalities (1) and (4) in Fig. 7.

a) Inequality constraints: Inequality (9) in Fig. 7 is converted to a set of processing time-related tightened constraints as follows:

$$\delta_2 + \delta_3 \leq 2(\delta_1 + \delta_4) \tag{25a}$$
$$\delta_3 + \delta_4 \leq 2(\delta_2 + \delta_5) \tag{25b}$$
$$\delta_4 + \delta_5 \leq 2(\delta_3 + \delta_6) \tag{25c}$$
$$\delta_5 + \delta_6 \leq 2(\delta_4 + \delta_7) \tag{25d}$$
$$\delta_6 + \delta_7 \leq 2(\delta_5 + \delta_8). \tag{25e}$$

Equation (25a) implies that if δ_2 and δ_3 are both 1, either δ_1 or δ_4 must be 1 because of "nonpreemptive" processing time requirements, similar for (25b)–(25e). The above set of processing time-related tightened constraints has been reported in our previous work [6].

The above set of tightened constraints can be generalized for all operations with different processing times as follows:

$$\sum_{\tau=k+1}^{k+p_{jm}-1} \delta_{jm\tau} \leq (p_{jm} - 1)(\delta_{jmk} + \delta_{jm,k+p_{jm}}),$$
$$m \in M_j, \ k \in [1, K - p_{jm}]. \tag{26}$$

Equation (26) implies that if $\delta_{jm\tau}$ are all 1, either δ_{jmk} or $\delta_{jm,k+p_{jm}}$ must be 1.

When examining inequalities (6)–(10) in Fig. 7 as a group, they can be put together in the matrix form as follows:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & -1 & 1 & 0 \\ 0 & -1 & 1 & 0 & -1 \\ -1 & 1 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \delta_4 \\ \delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \end{pmatrix} \leq 0. \tag{27}$$

Note that δ_1–δ_3 do not show up in (27). The reason is that if δ_4–δ_8 are properly regulated to satisfy the "nonpreemptive"

TABLE I
CONSTRAINT ANALYSIS FOR (27-2)

Case	δ_7	δ_8	Eq. (25-2)	Satisfy or not
1	0	0	$-0 + 0 \leq 0$	Yes
2	**0**	**1**	**$-0 + 1 \leq 0$**	**No**
3	1	0	$-1 + 0 \leq 0$	Yes
4	1	1	$-1 + 1 \leq 0$	Yes

TABLE II
CONSTRAINT ANALYSIS FOR (27-4)

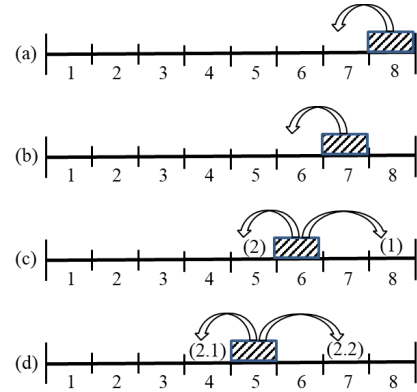| Case	δ_5	δ_6	δ_7	δ_8	Eq. (24-4)	Satisfy or not
1	0	0	-	0	$-0 + 0 -0 \leq 0$	Yes
2	0	0	-	1	$-0 + 0 -1 \leq 0$	Yes
3	**0**	**1**	**-**	**0**	**$-0 + 1 -0 \leq 0$**	**No**
4	0	1	-	1	$-0 + 1 -1 \leq 0$	Yes
5	1	0	-	0	$-1 + 0 -0 \leq 0$	Yes
6	1	0	-	1	$-1 + 0 -1 \leq 0$	Yes
7	1	1	-	0	$-1 + 1 -0 \leq 0$	Yes
8	**1**	**1**	**-**	**1**	**$-1 + 1 -1 \leq 0$**	**Yes (infeasible)**

Fig. 8. Intuitive description of (27). (a) If $\delta_8 = 1$ then $\delta_7 = 1$. (b) If $\delta_7 = 1$ then $\delta_6 = 1$. (c) If $\delta_6 = 1$ then $\delta_5 = 1$ or $\delta_8 = 1$. (d) If $\delta_5 = 1$ then $\delta_4 = 1$ or $\delta_7 = 1$.

processing time requirements by (27), and then, δ_1–δ_3 are expected to satisfy the requirements because of the processing time-related tightened constraint (19). In addition, although the first row of (27), denoted as (27-1), is redundant given the binary requirements of δ, it helps put the constraints together in a square matrix form. Physical meanings of (27-2)–(27-5) are analyzed one by one in the following.

Physical meanings of (27-2) under all combinations of binary variables involved are shown in Table I.

It can be seen that (27-2) guarantees that if δ_8 is 1, δ_7 must be 1 as implied by the second row of Table I. This is reasonable because the last possible three time slots to process the operation are 6–8, given that the processing time is 3. If the eighth time slot is taken, then the seventh must be taken too, and otherwise, the operation cannot be completed within the scheduling horizon. The physical meaning of (27-3) is similar, and if δ_7 is 1, δ_6 must be 1, given that the processing time is 3.

Physical meanings of (27-4) under all combinations of binary variables involved are shown in Table II.

It can be seen that (27-4) guarantees that δ_6 cannot be 1 when δ_5 and δ_8 are both 0 as implied by the third row of Table II. In other words, if δ_6 is 1, one of δ_5 and δ_8 has to be 1. This is reasonable because if neither the fifth time slot

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

YAN *et al.*: INNOVATIVE FORMULATION TIGHTENING APPROACH FOR JOB-SHOP SCHEDULING

9

nor the eighth is taken, then the sixth cannot be taken based on the processing time requirement. Note that Case 8 is not feasible because δ_5 and δ_8 cannot be 1 at the same time as guaranteed by (19b).

Physical meanings of (27-5) under all combinations of binary variables involved are shown in Table VI in the Appendix, as well as the analysis. The physical meanings of (27) can be intuitively shown in Fig. 8.

In Fig. 8(a), if δ_8 is 1, then δ_7 must be 1. In Fig. 8(b), if δ_7 is 1, then δ_6 must be 1. In Fig. 8(c), if δ_6 is 1, then one of δ_5 and δ_8 has to be 1. In Fig. 8(d), if δ_5 is 1, then one of δ_4 and δ_7 has to be 1. Equation (27) with the processing time-related tightened constraint (19) guarantees a contiguous time period with a length of 3 to process the operation. For example, if δ_6 is 1, there are two possibilities: 1) δ_8 is 1 or 2) δ_5 is 1. The first situation is simple since δ_7 will be 1 when δ_8 is 1, and thus, time slots 6–8 are used to process the operation. For the second situation, δ_5 is 1, and there are two possibilities again: 1) δ_4 is 1 or 2) δ_7 is 1. For the first situation, time slots 4–6 are used to process the operation. For the second situation, time slots 5–7 are used. Among all situations described above, a contiguous time period with length of 3 is guaranteed to process the operation. A similar analysis can be performed for other δ.

With further analysis on the meanings of (27), it can be extended to a set of tightened constraints (28) for problems with longer time periods as follows:

$$
\begin{pmatrix}
\dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
\dots & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
\dots & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 \\
\dots & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 1 \\
\dots & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 \\
\dots & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\
\dots & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 & 1 \\
\dots & -1 & 1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 \\
\dots & 1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots
\end{pmatrix}
\begin{pmatrix}
\delta_4 \\ \delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \\ \delta_9 \\ \delta_{10} \\ \delta_{11} \\ \dots \\ \delta_K
\end{pmatrix} \le 0.
$$

(28)

The entries y_{nh} of the above matrix are presented as follows.

Let $n + h = K - 3 + z, z \in Z$

$$
y_{nh} = \begin{cases}
0, & z \le 0 \\
-1, & z > 0 \text{ and } z \bmod 3 = 1 \\
1, & z > 0 \text{ and } z \bmod 3 = 2 \\
0, & z > 0 \text{ and } z \bmod 3 = 0.
\end{cases}
$$

(29)

The meaning of (28) is similar to (27). For example, when δ_k is 1, the meaning is intuitively shown in Fig. 9.

For example, if δ_k is 1, there are two possibilities: 1) δ_{k-1} is 1 or 2) δ_{k+2} is 1. For Case 1), there are two possibilities again: 1) δ_{k-2} is 1 or 2) δ_{k+1} is 1. For each of them, a contiguous time period with length of 3 is guaranteed to process the operation. For Case 2), δ_{k+2} is 1, and there are two possibilities again: 1) δ_{k+1} is 1 or 2) δ_{k+4} is 1. For Case (2.1), a contiguous time period with length of 3 is guaranteed to process the operation. For Case (2.2), according to (28), one of δ_{k+3} and δ_{k+6} has

to be 1. This is contradictory with (20), where δ_{k+3} and δ_{k+6} have to be zero as δ_k is 1. Among all the situations described above, a contiguous time period with length of 3 is guaranteed. A similar analysis can be performed for other δ.

When the processing time is 2 and 4, the corresponding sets of tightened constraints for problems with longer time periods are shown in (30) and (32), respectively. Their entries are presented in (31) and (33)

$$
\begin{pmatrix}
\dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
\dots & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\
\dots & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 \\
\dots & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 1 & -1 \\
\dots & 0 & 0 & 0 & 1 & -1 & -1 & 1 & -1 & 1 \\
\dots & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 1 & -1 \\
\dots & 0 & 0 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
\dots & 0 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\
\dots & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots
\end{pmatrix}
\begin{pmatrix}
\delta_3 \\ \delta_4 \\ \delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \\ \delta_9 \\ \delta_{10} \\ \dots \\ \delta_K
\end{pmatrix} \le 0
$$

(30)

$P = 2, \quad n + h = K - 2 + z, \ z \in Z$

$$
y_{nh} = \begin{cases}
0, & z \le 0 \\
-1, & z > 0 \text{ and } y \bmod 2 = 1 \\
1, & z > 0 \text{ and } y \bmod 2 = 0
\end{cases}
$$

(31)

$$
\begin{pmatrix}
\dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
\dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\
\dots & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\
\dots & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\
\dots & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 \\
\dots & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 \\
\dots & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 \\
\dots & & -1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\
\dots & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 \\
\dots & 1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots
\end{pmatrix}
\begin{pmatrix}
\delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \\ \delta_9 \\ \delta_{10} \\ \delta_{11} \\ \delta_{12} \\ \dots \\ \delta_K
\end{pmatrix} \le 0.
$$

(32)

$P = 4, \quad n + h = K - 4 + z, \ z \in Z$

$$
y_{nh} = \begin{cases}
0, & z \le 0 \\
-1, & z > 0 \text{ and } y \bmod 4 = 1 \\
1, & z > 0 \text{ and } y \bmod 4 = 2 \\
0, & z > 0 \text{ and } y \bmod 4 = 3 \\
0, & z > 0 \text{ and } y \bmod 4 = 0.
\end{cases}
$$

(33)

b) Equality Constraints: With further analysis on the meanings of equalities (1) and (4) in Fig. 7 under all possible part statuses, i.e., active or not at each time slot, two new sets of beginning/completion time-related tightened constraints are obtained as follows:

$$
b = K - p + 1 - 2(\delta_1 + \delta_2) - (\delta_3 + \delta_4 + \delta_5) - 0(\delta_6 + \delta_7 + \delta_8)
$$

(34)

$$
c = -0(\delta_8 + \delta_7) + K\delta_6 - (\delta_5 + \delta_4) + (K - 1)\delta_3 - 2(\delta_2 + \delta_1).
$$

(35)

Since the processing time is p and the operation must be completed within the scheduling horizon, the largest beginning

time is $K - p + 1$ with δ_6–δ_8 as 1 as implied in (34). When the starting of nonzero δ moves earlier, b gets smaller. The earlier δ, the larger the negative impacts on b. The meaning of (35) is similar. The largest completion time is K with δ_6–δ_8 as 1. When the starting of nonzero δ moves earlier, c gets smaller. It can be verified that these constraints are meaningful under all possible part statuses of δ_1–δ_8.

The above two tightened constraints can be generalized for all operations with different processing times as follows:

$$
b_j = K - \sum_{m \in M_j} x_{jm} p_{jm} + 1
$$
$$
- \sum_{m \in M_j} \sum_{n=0}^{n=\lfloor K/p_{jm} \rfloor} n \left(\sum_{\tau=0}^{\tau=p_{jm}-1:np_{jm}+\tau<K} \delta_{jm,K-np_{jm}-\tau} \right)
\tag{36}
$$

$$
c_j = \sum_{m \in M_j} \sum_{n=0}^{n=\lfloor K/p_{jm} \rfloor}
$$
$$
\times \left(\begin{array}{l} -n \left(\sum_{\tau=0}^{\tau=p_{jm}-2:np_{jm}+\tau<K} \delta_{jm,K-np_{jm}-\tau} \right) \\ +(K-n) \sum_{\tau=1}^{\tau=1:np_{jm}+p_{jm}-\tau<K} \delta_{jm,K-np_{jm}-p_{jm}+\tau} \end{array} \right).
\tag{37}
$$

Equations (20), (26), (36), and (37) directly constrain δ_1–δ_8, b, and c within one operation. For the single-operation part problem with $p = 3$ and $K = 8$ under consideration, with (19) and (25), the total number of vertices decreases from 1234 to 250 in a major way. With (34) and (35), it is further reduced to 42. After replacing (25) by (27), the total number of vertices is 6, and all of the vertices are integral vertices, implying that the formulation is tight. For the single operation scheduling problem with the processing time as 1–4, tight formulations are obtained.

2) Two Operations: Now, add another operation with a processing time of 3 to the problem in Section V-A1, with additional operation precedence constraint (6). For each operation, decision variables include b, c, and one set of δ_k. With the standard formulation established in Section III, after relaxing integrality requirements, 63 872 vertices are obtained by constraint-to-vertex conversion. After applying (20) and (26) obtained in the one-operation example to both operations, a total number of 23 206 vertices remains. With (36) and (37) applied to both operations, 333 vertices remain. After eliminating factional vertices, there are six integral vertices. After vertex-to-constraint conversion, the resulting tight constraints are obtained, as shown in Fig. 10.

After analyzing the meanings of equalities (2)–(7) in Fig. 10 under possible part statuses, two sets of operation precedence-related tightened constraints are obtained as follows:

$$
\delta_{1,k} = 0, \quad k \in [K - p_2 + 1, K]
\tag{38}
$$
$$
\delta_{2,k} = 0, \quad k \in [1, p_1].
\tag{39}
$$

Since the two operations need to be completed in the scheduling horizon, the largest completion time for operation two is K, with the beginning time of $K - p_2 + 1$. Therefore, operation one must be completed by that time, and $\delta_{1,k}$ must

be 0 for period of $k \in [K - p_2 + 1, K]$ as implied in (38). The meaning of (39) is similar. The smallest beginning time of operation one is 1, with completion time of p_1. Therefore, operation two cannot start before p_1, and $\delta_{2,k}$ must be 0 for period of $k \in [1, p_1]$.

Equations (38) and (39) directly constrain $\delta_{1,k}$ and $\delta_{2,k}$ across operations. With them, the total number of vertices decreases to 14 from 333 in a major way.

The above two tightened constraints can be generalized for all operations with different processing times as follows:

$$
\delta_{jmk} = 0, \quad j \in [1, J-1], \ m \in M_j,
$$
$$
k \in \left[K - \sum_{g=j+1}^{J} \min_{m \in M_g} p_{gm} + 1, K \right]
\tag{40}
$$

$$
\delta_{jmk} = 0, \quad j \in [2, J], \ m \in M_j, \ k \in \left[1, \sum_{g=1}^{j-1} \min_{m \in M_g} p_{gm} \right].
\tag{41}
$$

3) One Operation With Linearized Tardiness: Now, add the constraints associated with tardiness to the problem in Section V-A1, assuming due date d is 2. Constraints under consideration are processing time requirements (1) and (3)–(5) and tardiness constraints (11)–(16). Decision variables include b, c, a set of δ_k, a set of continuous variables w, a set of binary variables α, and tardiness T. With the standard formulation established in Section III, after relaxing integrality requirements, 6170 vertices are obtained by constraint-to-vertex conversion. With (19), (25), (34), and (35) obtained in the one-operation example, a total number of 210 vertices remain. Eliminate factional vertices, and obtain eight integral vertices. After vertex-to-constraint conversion, the resulting tight constraints are shown in Fig. 11.

By combining equalities (2), (5), and (10) in Fig. 11, the following constraint is obtained:

$$
T = \delta_3 + \delta_4 + \delta_5 + 2\delta_6 + 2\delta_7 + 2\delta_8.
\tag{42}
$$

Since the processing time is 3 and the due date is 2, the smallest tardiness is 1 with δ_1–δ_3 as 1, and the largest tardiness is 6 with δ_6–δ_8 as 1 as implied in (42). When the starting of nonzero δ moves earlier, T gets smaller. The earlier δ, the smaller the impacts on T. Equation (42) directly constrains δ_1–δ_8 and tardiness T. With it, the total number of vertices decreases to 84 from 210.

After analyzing the physical meanings, (42) is converted to the following constraint in a generic form (43), as shown at the bottom of the next page:

The tightened constraints obtained in Sections V-A1–V-A3 tighten the formulation. However, they can hardly be obtained manually without going through the above tightening process. The formulation with them is much tighter (not tight yet) than the original one. Those tightened constraints can be extended to other parts with more operations and processing time other than 3, whose due date is positive [only for (43)].

B. Example 2: Medium-Sized Problems

This medium-sized example is to demonstrate the effectiveness and robustness of formulation tightening. The instance is

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

YAN *et al.*: INNOVATIVE FORMULATION TIGHTENING APPROACH FOR JOB-SHOP SCHEDULING

11

TABLE III
COMPARISON OF FORMULATIONS: MEDIUM-SIZED

Formulation	Total tardiness	Total cycle time	MIP gap (%)	CPU (s)	Solve (s)	Cut (s)	Branch (s)
(a): Original	35,089	477	0.01	932.3	930.9	20.1	866.3
(b): (a) + (20), (26)	35,089	484	0.01	144.3	142.6	10.7	94.8
(c): (b) + (36)-(37)	35,089	487	0.01	4.8	2.9	1.0	0.0
(d): (c) + (40)-(41)	35,089	487	0.01	4.7	2.8	1.0	0.0
(e): (d) + (43)	35,089	482	0.01	4.7	3.1	1.1	0.0
(f): replace (26) in (e) by (28), (30), (32)	35,089	492	0.01	6.0	3.8	0.9	0.0

created based on the first 89 parts and all machines in [9]. The largest number of operations of parts is 6. According to which parts/operations that machines can process, machines are categorized into 17 types, and each type has 1–6 machines with the same function. Assume that each operation of each pat can only be processed on one machine type, and machines are always available for simplicity. The number of time slots under consideration is 220 so that all the parts can be processed. There are three values for tardiness weights, 1, 10, and 100, and they are randomly assigned to parts with the percentage of 50%, 40%, and 10%, respectively. The weight for the total tardiness is 0.9.

Before and after adding tightened constraints, the overall job-shop scheduling problems are solved by using branch-and-cut implemented in CPLEX. In CPLEX, optimization stops when computational time reaches the preset stop time or the relative Mixed-Integer Programming (MIP) gap (relative difference between the objectives of the optimal relaxed solution and current integer solution) falls below the preset gap. Here, the stop MIP gap is set as 0.01% and there is no time limit.

With different formulations, the results are presented in Table III: (a) the original formulation; (b) adding (20) and (26); (c) adding (36) and (37); (d) adding (40) and (41); (e) adding (43) to operations with processing time of 1–3 time slots; and (f) replacing (26) in (e) by (28), (30), and (32) for operations with a processing time of 2–4 time slots. From 1) to 5), it is accumulative. CPU time consists of three parts: data and model loading, solving, and solution outputting.

According to Table III, the solutions with the same quality are obtained under different formulations. The CPU, solving, cutting, and branching time are much reduced by adding new tightened constraints (20), (26), (36), (37), (40), (41), and (43). With the standard formulation, a feasible solution with the total weighted tardiness of 35 089 and total cycle time of 477 is obtained in 932 s, while the time on cutting and branching is 20 and 866 s, respectively. By adding new
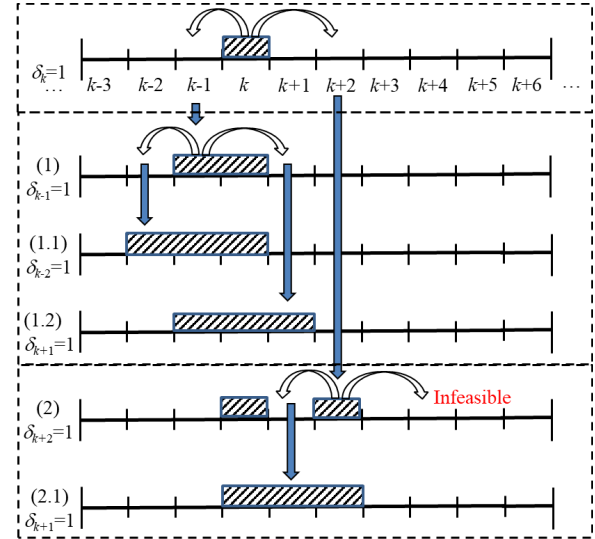

Fig. 9.   Intuitive description of (28).

tightened constraints (20), (26), (36), (37), (40), (41), and (43), a similar feasible solution is obtained in 4.7 s, while the cutting time is 1.1 s and there are no branching operations. With tightened constraints, less time is required to obtain the same MIP gap of 0.01% compared with the standard formulation.

When replacing (26) by (28), (30), and (32) (the formulation becomes tighter), a similar solution is obtained in 6 s, with cutting time of 0.9 s. The reason is that when solving the problem by using branch-and-cut, cuts are performed around the optimal solution to the integer-relaxed problem [31], not on the entire feasible region as mentioned in Section I. Therefore, more tightened constraints may not guarantee better computational efficiency. There is a tradeoff between tightness and computational efficiency.

The problem is also solved with randomly assigned part tardiness weights considering formulations (a), (e), and (f) presented above. Cutting and branching time for problems with different sets of weights are compared in Fig. 12. Then, for each part, a random variable following $U(-5, 5)$ is generated and added to the part due date, while tardiness weights are the same as the original problem. The results are shown in Fig. 13.

For every instance, solutions with the same quality are obtained with and without tightened constraints. By adding tightened constraints, the total cutting and branching time is significantly reduced, and the reduction is mainly from the reduction of branching time. Results demonstrate the effectiveness and robustness of our formulation tightening.

$$T = \sum_{m \in M_J} \begin{cases} \sum_{n=1}^{\lfloor (K-d+1)/p_{Jm} \rfloor +1} \left((p_{Jm} - d + n - 1) \sum_{\tau=0}^{0:p_{Jm}n+\tau \leq K} \delta_{Jm,p_{Jm}n+\tau} \\ + n \sum_{\tau=1}^{p_{Jm}-1:p_{Jm}n+\tau \leq K} \delta_{Jm,p_{Jm}n+\tau} \right), & p_{Jm} \geq d \\ \sum_{n=1}^{\lfloor (K-d+1)/p_{Jm} \rfloor +1} \left(n \sum_{\tau=1}^{p_{Jm}:d+p_{Jm}(n-1)+\tau \leq K} \delta_{Jm,d+p_{Jm}(n-1)+\tau} \right), & p_{Jm} < d. \end{cases} \tag{43}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

12

IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING

$$
\begin{aligned}
&(\;\; 1) + c_2 + \delta_{2,4} + \delta_{2,5} - 8\delta_{2,6} = 0;\\
&(\;\; 2) -\delta_{1,6} = 0;\\
&(\;\; 3) -\delta_{1,7} = 0;\\
&(\;\; 4) -\delta_{1,8} = 0;\\
&(\;\; 5) -\delta_{2,1} = 0;\\
&(\;\; 6) -\delta_{2,2} = 0;\\
&(\;\; 7) -\delta_{2,3} = 0;\\
&(\;\; 8) +2\delta_{1,3} + b_2 - c_2 = 0;\\
&(\;\; 9) -\delta_{1,1} + \delta_{1,3} - \delta_{1,4} = 0;\\
&(10) -\delta_{1,2} + \delta_{1,3} - \delta_{1,5} = 0;\\
&(11) -\delta_{2,4} + \delta_{2,6} - \delta_{2,7} = 0;\\
&(12) -\delta_{2,5} + \delta_{2,6} - \delta_{2,8} = 0;\\
&(13) -4b_2 + 3c_2 - \delta_{2,4} - \delta_{2,5} = 0;\\
&(14) + c_1 + \delta_{1,1} + \delta_{1,2} - 5\delta_{1,3} = 0;\\
&(15) -5b_1 + 3c_2 - 2\delta_{1,1} - 2\delta_{1,2} = 0;\\
&(16) + \delta_{2,6} = 1;\\
&(17) -\delta_{1,5} <= 0;\\
&(18) -\delta_{1,4} + \delta_{1,5} <= 0;\\
&(19) -\delta_{2,7} + \delta_{2,8} <= 0;\\
&(20) +\delta_{1,5} - \delta_{2,8} <= 0;\\
&(21) +\delta_{1,4} - \delta_{2,7} <= 0;\\
&(22) +\delta_{2,7} <= 1.
\end{aligned}
$$

Fig. 10. Ex.1-b) tightened constraints.

$$
\begin{aligned}
&(\;\; 1) +c + 2\delta_1 + 2\delta_2 - 7\delta_3 + \delta_4 + \delta_5 - 8\delta_6 = 0;\\
&(\;\; 2) +\delta_4 + \delta_5 + \delta_6 + 2\delta_7 + 2\delta_8 + w_2 - 15w_3 = 0;\\
&(\;\; 3) -w_1 = 0;\\
&(\;\; 4) -\alpha_1 = 0;\\
&(\;\; 5) +16w_3 - T = 0;\\
&(\;\; 6) +\alpha_2 - \alpha_3 = 0;\\
&(\;\; 7) +w_2 + w_3 - \alpha_2 = 0;\\
&(\;\; 8) -\delta_1 + \delta_3 - \delta_4 + \delta_6 - \delta_7 = 0;\\
&(\;\; 9) -\delta_2 + \delta_3 - \delta_5 + \delta_6 - \delta_8 = 0;\\
&(10) +15\delta_3 - \delta_4 - \delta_5 + 14\delta_6 - 2\delta_7 - 2\delta_8 - 16w_2 = 0;\\
&(11) -4b + 3c - 2\delta_1 - 2\delta_2 - \delta_3 - \delta_4 - \delta_5 = 0;\\
&(12) +\alpha_3 = 1;\\
&(13) +2\delta_5 + \delta_6 + \delta_7 + 3\delta_8 - T <= -1;\\
&(14) -\delta_8 <= 0;\\
&(15) -\delta_6 + \delta_7 <= 0;\\
&(16) -\delta_7 + \delta_8 <= 0;\\
&(17) -\delta_5 + \delta_6 - \delta_8 <= 0;\\
&(18) -\delta_5 - \delta_6 - \delta_7 - 2\delta_8 + T <= 2.
\end{aligned}
$$

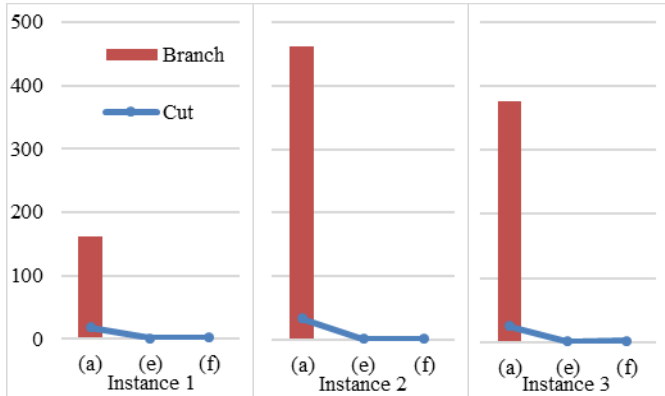Fig. 11. Ex.1-c) tightened constraints.

Fig. 12. Cutting, branching, and other times under different weights.

C. Example 3: Large-Sized Problems

This large-sized example is to demonstrate the performance of tightened single-part formulations. The instance is created based on all 127 parts and all machines in [9]. The number of time slots under consideration is 300 so that all the parts can be processed. The other problem setup is the same as that in
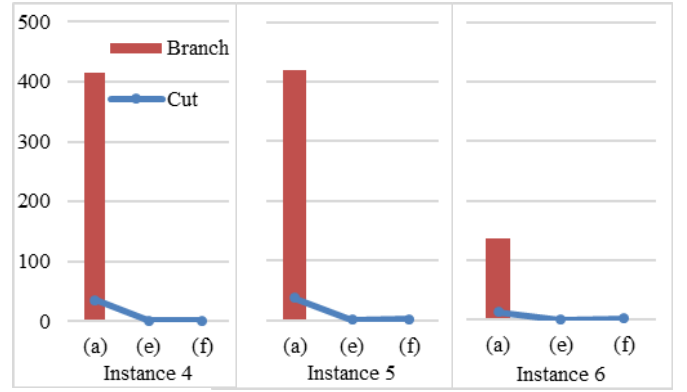

Fig. 13. Cutting, branching, and other times under different due dates.

TABLE IV

COMPARISON OF FORMULATIONS: LARGE-SIZED

Formulation	Total tardiness	Total cycle time	MIP gap (%)	CPU time (s)	Solving time (s)
(a): Original	N/A	N/A	N/A	1206.7	1203.8
(b): (a) + (20), (26)	14,831	780	0.93	1204.1	1201.0
(c): (b) + (36)-(37)	14,825	749	0.01	13.8	10.9
(d): (c) + (40)-(41)	14,825	749	0.01	13.3	10.6
(e): (d) + (43)	14,825	749	0.01	11.6	8.8
(f): replace (26) in (e) by (28), (30), (32)	14,825	752	0.01	43.0	38.8

TABLE V

COMPARISON OF FORMULATIONS: LARGE-SIZED
(MULTIPLE MACHINE TYPES)

Formulation	Total tardiness	Total cycle time	MIP gap (%)	CPU time (s)	Solving time (s)
(a): Original	N/A	N/A	N/A	1231.6	1224.0
(b): (a) + (20), (26)	14,923	864	1.78	1236.1	1201.3
(c): (b) + (36)-(37)	14,827	796	0.01	52.16	14.8
(d): (c) + (40)-(41)	14,827	794	0.01	50.0	14.7
(e): (d) + (43)	14,827	793	0.01	50.0	14.7

Example 2. Before and after adding tightened constraints, the job-shop scheduling problems are solved by using branch-and-cut, and the results are shown in Table IV. Since the problem is more complicated than Example 2, two stopping criteria are considered: 1200-s CPU time or 0.01% MIP gap.

According to Table IV, both the solution quality and computational efficiency are significantly improved by adding tightened constraints. With the standard formulation, no feasible solution can be found in 20 min. By adding new tightened constraints (20), (26), (36), (37), (40), (41), and (43), a feasible solution with an MIP gap of 0.01% is obtained in 12 s. The results show that tightening single parts also improves the solution quality and computational efficiency when solving the overall problems. Similar to Example 2, when replacing (26) by (28), (30), and (32) (the formulation becomes tighter), both the CPU and solving time increases.

Now, consider that some of the 127 parts can be processed on multiple machine types. The other problem setup is the same as that in the above. Before and after adding tightened constraints, the job-shop scheduling problems are solved by using branch-and-cut, and the results are shown in Table V.

According to Table V, both the solution quality and computational efficiency are significantly improved by adding

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

YAN *et al.*: INNOVATIVE FORMULATION TIGHTENING APPROACH FOR JOB-SHOP SCHEDULING

13

TABLE VI
CONSTRAINT ANALYSIS FOR (27-5)

Case	δ_4	δ_5	δ_6	δ_7	δ_8	Eq. (23-5)	Satisfy or not
1	0	0	-	0	0	-0 + 0 -0 +0 ⩽ 0	Yes
2	0	0	-	0	1	-0 + 0 -0 +1 ⩽ 0	No
3	0	0	-	1	0	-0 + 0 -1 +0 ⩽ 0	Yes (infeasible)
4	0	0	-	1	1	-0 + 0 -1 +1 ⩽ 0	Yes
5	0	1	-	0	0	-0 + 1 -0 +0 ⩽ 0	No
6	0	1	-	0	1	-0 +1 - 0 +1 ⩽ 0	No
7	0	1	-	1	0	-0 +1 - 1 -0 ⩽ 0	Yes
8	0	1	-	1	1	-0 + 1 -1 +1 ⩽ 0	No
9	1	0	-	0	0	-1 + 0 -0 +0 ⩽ 0	Yes
10	1	0	-	0	1	-1 + 0 -0 +1 ⩽ 0	Yes (infeasible)
11	1	0	-	1	0	-1 + 0 -1 +0 ⩽ 0	Yes (infeasible)
12	1	0	-	1	1	-1 + 0 -1 +1 ⩽ 0	Yes (infeasible)
13	1	1	-	0	0	-1 + 1 -0 +0 ⩽ 0	Yes
14	1	1	-	0	1	-1 +1 - 0 +1 ⩽ 0	No
15	1	1	-	1	0	-1 +1 - 1 +0 ⩽ 0	Yes (infeasible)
16	1	1	-	1	1	-1 + 1 -1 +1 ⩽ 0	Yes (infeasible)

tightened constraints. With the standard formulation, no feasible solution can be found in 20 min. By adding new tightened constraints, a feasible solution with an MIP gap of 0.01% is obtained in 50 s.

The above results demonstrate the great potential of our formulation tightening method for complex ILP and MILP problems where the values of integer variables are uniquely determined by the values of binary variables.

VI. CONCLUSION

In this article, an innovative and systematic method is established for the first time to tighten the formulations of individual parts with multiple operations in the data preprocessing stage. It is a major advancement of our previous work on problems with binary and continuous variables to integer variables. The idea is to first link integer variables to binary variables by innovatively combining constraints so that the integer variables are uniquely determined by the binary variables. With binary variables only, the vertices of the convex hull can be obtained based on the vertices of the linear problem after relaxing binary requirements with proved tightness. These vertices are then converted back to tightened constraints with coefficients characterized by part parameters and the length of the scheduling horizon. The tightening process only needs to perform once, and the resulting tightened constraints can be easily adjusted for other data sets after parameterization and can be directly applied in the data preprocessing stage, tremendously reducing online computational requirements. This method significantly improves our previous results on tightening individual operations. Numerical results demonstrate significant benefits on solution quality and computational efficiency.

Beyond MILP job-shop scheduling problems under consideration, this approach also applies to other complex ILP and MILP problems with similar characteristics between integer and binary variables, such as job-shop scheduling problems with other features like sequence-dependent setups. For practical applications, the idea is to obtain "near-tight" formulations by partially tightening. The approach fundamentally

changes the way how such problems are formulated and solved. In addition, this method goes naturally with part-based decomposition and coordination approaches, a subject worthy of further exploration.

APPENDIX

A. Physical Meanings of Tight Constraints

Physical meanings of (27-5) under all combinations of binary variables involved are shown in Table VI.

It can be seen that (27-5) guarantees that δ_5 cannot be 1 when δ_4 and δ_7 are both 0 as implied by the fifth row of Table VI. In other words, if δ_5 is 1, one of δ_4 and δ_7 has to be 1. This is reasonable because if neither of the fourth time slot or the seventh is taken, the fifth cannot be taken based on the processing time requirement. Note that Case 10 in Table VI is not feasible because if δ_8 is 1, δ_7 must be 1 as guaranteed by (27-2). Cases 11, 12, 15, and 16 are not feasible as guaranteed by (19). For Case 3, since δ_6 cannot be 1 when δ_5 and δ_8 are both 0 as guaranteed by (27-4), δ_6 has to be 0. However, if δ_7 is 1, δ_6 must be 1 as guaranteed by (27-3). Therefore, Case 3 is not infeasible either.

REFERENCES

[1] P. Brucker, *Scheduling Algorithms*, 5th ed. Berlin, Germany: Springer, 2006.
[2] P. Brucker, "The job-shop problem: Old and new challenges," in *Proc. 3rd Multidisciplinary Int. Conf. Scheduling: Theory Appl.*, 2007, pp. 15–22.
[3] D. P. Bertsekas, *Nonlinear Programming*, 3rd ed. Nashua, NH, USA: Athena Scientific, 2016.
[4] B. Yan *et al.*, "A systematical approach to tighten unit commitment formulations," in *Proc. IEEE Power Energy Soc. Gen. Meeting (PESGM)*, Aug. 2018, pp. 1–5.
[5] B. Yan *et al.*, "A systematic formulation tightening approach for unit commitment problems," *IEEE Trans. Power Syst.*, vol. 35, no. 1, pp. 782–794, Jan. 2020.
[6] B. Yan, M. A. Bragin, and P. B. Luh, "Novel formulation and resolution of job-shop scheduling problems," *IEEE Robot. Autom. Lett.*, vol. 3, no. 4, pp. 3387–3393, Oct. 2018.
[7] B. Yan, M. A. Bragin, and P. B. Luh, "Tightened formulation and resolution of energy-efficient job-shop scheduling," in *Proc. IEEE 16th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2020, pp. 741–746.
[8] T. Yamada and N. R. Nakano, *Job Shop Scheduling* (IEE Control Engineering Series), 1997.
[9] D. J. Hoitomt, P. B. Luh, and K. R. Pattipati, "A practical approach to job-shop scheduling problems," *IEEE Trans. Robot. Autom.*, vol. 9, no. 1, pp. 1–13, Feb. 1993.
[10] T. Sun, P. B. Luh, and M. Liu, "Lagrangian relaxation for complex job shop scheduling," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2006, pp. 1432–1437.
[11] T. Nishi, Y. Hiranaka, and M. Inuiguchi, "Lagrangian relaxation with cut generation for hybrid flowshop scheduling problems to minimize the total weighted tardiness," *Comput. Oper. Res.*, vol. 37, no. 1, pp. 189–198, Jan. 2010.
[12] K. Mao, Q.-K. Pan, X. Pang, and T. Chai, "A novel Lagrangian relaxation approach for a hybrid flowshop scheduling problem in the steelmaking-continuous casting process," *Eur. J. Oper. Res.*, vol. 236, no. 1, pp. 51–60, Jul. 2014.
[13] E. Asadi-Gangraj, "Lagrangian relaxation approach to minimize makespan for hybrid flow shop scheduling problem with unrelated parallel machines," *Scientia Iranica*, vol. 25, no. 6, pp. 3765–3775, 2018.

[14] B.-H. Zhou, L.-M. Hu, and Z.-Y. Zhong, "A hybrid differential evolution algorithm with estimation of distribution algorithm for reentrant hybrid flow shop scheduling problem," *Neural Comput. Appl.*, vol. 30, no. 1, pp. 193–209, Jul. 2018.

[15] R. Z. Ríos-Mercado and J. F. Bard, "Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups," *Comput. Oper. Res.*, vol. 25, no. 5, pp. 351–366, May 1998.

[16] J. C.-H. Pan and J.-S. Chen, "Mixed binary integer programming formulations for the reentrant job shop scheduling problem," *Comput. Oper. Res.*, vol. 32, no. 5, pp. 1197–1212, May 2005.

[17] M. Karimi-Nasab and M. Modarres, "Lot sizing and job shop scheduling with compressible process times: A cut and branch approach," *Comput. Ind. Eng.*, vol. 85, pp. 196–205, Jul. 2015.

[18] C. Özgüven, Y. Yavuz, and L. Özbakır, "Mixed integer goal programming models for the flexible job-shop scheduling problems with separable and non-separable sequence dependent setup times," *Appl. Math. Model.*, vol. 36, no. 2, pp. 846–858, Feb. 2012.

[19] M. Karimi-Nasab and S. M. Seyedhoseini, "Multi-level lot sizing and job shop scheduling with compressible process times: A cutting plane approach," *Eur. J. Oper. Res.*, vol. 231, no. 3, pp. 598–616, Dec. 2013.

[20] J. Cheng, F. Chu, and M. Zhou, "An improved model for parallel machine scheduling under time-of-use electricity price," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 896–899, Apr. 2018.

[21] S. Zhang and S. Wang, "Flexible assembly job-shop scheduling with sequence-dependent setup times and part sharing in a dynamic environment: Constraint programming model, mixed-integer programming model, and dispatching rules," *IEEE Trans. Eng. Manag.*, vol. 65, no. 3, pp. 487–504, Aug. 2018.

[22] L. Meng, C. Zhang, B. Zhang, and Y. Ren, "Mathematical modeling and optimization of energy-conscious flexible job shop scheduling problem with worker flexibility," *IEEE Access*, vol. 7, pp. 68043–68059, 2019.

[23] 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," *Int. J. Prod. Res.*, vol. 57, no. 1, pp. 61–82, Jan. 2019.

[24] 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," *Comput. Ind. Eng.*, vol. 142, Apr. 2020, Art. no. 106347.

[25] B. Yan, H. Y. Chen, P. B. Luh, S. Wang, and J. Chang, "Litho machine scheduling with convex hull analyses," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 4, pp. 928–937, Oct. 2013.

[26] H. D. Sherali and P. J. Driscoll, "On tightening the relaxations of Miller-Tucker-Zemlin formulations for asymmetric traveling salesman problems," *Oper. Res.*, vol. 50, no. 4, pp. 656–669, Aug. 2002.

[27] D. Bienstock and B. McClosky, "Tightening simple mixed-integer sets with guaranteed bounds," *Math. Program.*, vol. 133, nos. 1–2, pp. 337–363, Jun. 2012.

[28] E. M. L. Beale and J. J. H. Forrest, "Global optimization using special ordered sets," *Math. Program.*, vol. 10, no. 1, pp. 52–69, Dec. 1976.

[29] G. B. Dantzig and B. C. Eaves, "Fourier-Motzkin elimination and its dual," *J. Combinat. Theory A*, vol. 14, no. 3, pp. 288–297, May 1973.

[30] *Heidelberg University*. Accessed: Oct. 1, 2019. [Online]. Available: http://www.iwr.uni-heidelberg.de/groups/comopt/software/PORTA/

[31] *IBM, ILOG CPLEX V 12.1 User's Manual*, 2018.

Bing Yan (Member, IEEE) received the B.S. degree from the Renmin University of China, Beijing, China, in 2010, and the M.S. and Ph.D. degrees from the University of Connecticut, Storrs, CT, USA, in 2012 and 2016, respectively.

She is currently an Assistant Professor with the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology, Rochester, NY, USA. Before joining the Rochester Institute of Technology, she was an Assistant Research Professor with the Department of Electrical and Computer Engineering, University of Connecticut. Her research interests include manufacturing system scheduling, power system optimization, mathematical optimization, formulation tightening, and operation optimization of microgrids and distributed energy systems.

Mikhail A. Bragin (Member, IEEE) received the B.S. and M.S. degrees in mathematics from Voronezh State University, Voronezh, Russia, in 2004, the M.S. degree in physics and astronomy from the University of Nebraska–Lincoln, Lincoln, NE, USA, in 2006, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Connecticut, Storrs, CT, USA, in 2014 and 2016, respectively.

He is currently an Assistant Research Professor in electrical and computer engineering at the University of Connecticut. His research interests include operations research, mathematical optimization, including power system optimization, grid integration of renewables (wind and solar), energy-based operation optimization of distributed energy systems, scheduling of manufacturing systems, and machine learning through deep neural networks.

Peter B. Luh (Life Fellow, IEEE) received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1973, the M.S. degree from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1977, and the Ph.D. degree from Harvard University, Cambridge, in 1980.

He has been with the University of Connecticut since 1980, where he is currently a Board of Trustees Distinguished Professor and the SNET Professor of communications and information technologies. His research interests include intelligent manufacturing, energy-smart buildings, and smart grid.

Dr. Luh was the Chair of the IEEE TAB Periodicals Committee from 2018 to 2019 and the Founding Editor-in-Chief of IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. He is also the Chair of the IEEE TAB Periodicals Review and Advisory Committee for the period 2020–2021.