An Innovative Formulation Tightening Approach for Job-Shop Scheduling

Bing Yan, *Member, IEEE*, Mikhail A. Bragin, *Member, IEEE*, 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 methods without much difficulty. The tightening process, however, is fundamentally challenging because of the existence of integer variables. In this paper, 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 linear problem after relaxing binary requirements. These vertices are then converted to tight 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 ILP 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 exit 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 much difficulty. In this paper, 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 ILP problems with similar characteristics and fundamentally changes the way how such problems are formulated and solved.**

***Index terms*–Manufacturing, job-shop scheduling, mixed-integer linear programming, formulation tightening**

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 jobs, 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 will be nonconvex 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 solution of the LP problem on the fly to get solutions to the MILP problem. If such solutions are obtained, the problem is directly solved. If not, the method replies 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

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).

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

This work is supported in part by the National Science Foundation (NSF) under the grant ECCS-1810108 and U.S. Department of Energy (DoE)'s Office of Energy Efficiency and Renewable Energy under the Advanced Manufacturing Office Award Number DE-EE0007613. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF or DoE.

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 for job-shop scheduling because of the existence of integer variables (e.g., beginning time) in addition to binary variables and of the interactions among multiple operations. In the literature, a few tightened single-part formulations were reported without a systematic approach. They were shown computationally efficient for the overall problems.

In this paper, the job-shop scheduling problem is first formulated in an integer programming form in Section III, and the objective is to minimize the total weighted tardiness and the total cycle time. Since tardiness is a nonlinear function of completion time, 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 since it is generalized from mixed-binary linear programming (MBLP) problems to mixed-integer linear programming 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 linear problem after relaxing binary requirements [4, 5]. These vertices are then converted to tight constraints. The number of resulting tight constraints, the number of variables involved, and constraint coefficients depend on part parameters. Since all parts must be processed within the scheduling horizon, the above also depends the length of the horizon. For general use purposes, these tight constraints are characterized by analyzing constraint structures and relationships 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, 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 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 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 MILP problems with unique relationships 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 Subsection A. Tightened constraints are reviewed in Subsection B.

A. Problem formulations and solution methodologies

With large numbers of decision variables and constraints in job-shop scheduling, developing efficient formulations is complex [7]. "Separable" and nonlinear formulations were established and efficiently exploited by decomposition and coordination-based Lagrangian relaxation methods in [8-13]. ILP models were also developed in [14-24]. Considering sequence-dependent setups, an ILP model was established in [14]. With additional variables, job successors and predecessors were modeled. In our previous work on high-volume and low-variety manufacturing [24], 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 linear programming 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 [14, 16-24], 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 [25]. For knapsack problems, tight formulations were obtained through the use of "structural" disjunctive cuts based on the problem structure [26].

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 [16]. 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 to 5 machines and 4 to 6 parts demonstrate 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 [14], and some of them are facet-defining cuts. Testing results based on randomly generated data for problems with 2 to 6 machines and 7 to 10 parts show that the computational time is much reduced with these tightened constraints. For both studies, testing results demonstrate computational efficiency of these tightened constraints.

In our previous work [6], a few processing time-related constraints were tightened for single parts based on integration of "constraint-and-vertex conversion" and "vertex projection" where non-integer values in vertices are rounded up or down to nearest feasible integers. For unit commitment problems in power systems, a systematic method was developed based on novel integration of "constraint-and-vertex conversion," "vertex elimination" and "parameterization" processes to tighten single-unit formulations in the data preprocessing stage for the first time [4, 5]. Results show that our formulation tightening is effective in terms of solution quality and computational efficiency.

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 Subsection A (part index i is omitted for brevity). Machine capacity constraints and the objective function are briefly described in Subsection 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. 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 indices k are considered as follows:

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

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.

Since the processing time p_{jm} is machine-type-dependent, a new set of binary decision variables is defined to identify the assignment of an 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.

a) 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.,

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

In the above, M_j denotes the set of machines 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.,

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

Logical Eq. (2) is linearized as follows:

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

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

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

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

b) Operation precedence constraints

It is assumed that the operation sequence of the part is fixed, and operation j+1 cannot start until j is finished, i.e.,

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

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

$$b_1 \ge a. \quad (7)$$

c) Machine type assignment constraints

Operation j can only be assigned to one machine type, i.e.,

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

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

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

d) Linearized tardiness

Tardiness T is formulated as follows,

$$\max(c_J - d, 0), \quad (10)$$

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

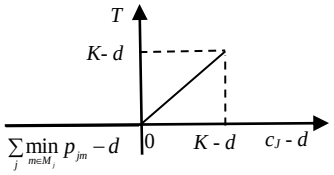

Figure 1. Tardiness function

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 [27]. Three continuous variables w^1, w^2, and w^3 ($0 \le w^1, w^2, w^3 \le 1$) are considered to represent 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,

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

$$T = o\omega^1 + 0\omega^2 + (K - d)\omega^3; \quad (12)$$

$$\alpha^l \ge \omega^l, 1 \le l \le 3; \quad (13)$$

$$\alpha^1 + \alpha^3 \le 1; \quad (14)$$

$$\sum_l \omega^l = 1; \quad (15)$$

$$\sum_l \alpha^l = 2. \quad (16)$$

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).

B. Machine capacity constraints and objective function

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

a) 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.,

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

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.

b) Objective function

The objective function is to minimize the weighted sum of total tardiness and total cycle time, i.e.,

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

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

The job-shop scheduling problem with Eqs. (1), (3)-(9), and (11-18) established above is an MILP problem. Most of the decision variables are binary (e.g., δ and x). There are also a few integer variables (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 [4-6], an innovative and systematic method is established to tighten the above single-part formulation in Subsection A. A numerical example is also presented to illustrate the tightening idea. Tightness is proved in Subsection B.

A. Formulation tightening

In our previous work on unit commitment in power systems, 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 $x_1 + x_2 \geqslant 0.5$. After relaxing integrality requirements, the vertices (blue dots in Fig. 2b) of the convex hull (blue lines in Fig. 2b) to the integer-relaxed problem are obtained. Then the vertices (red dots in Fig. 2a) of the original convex hull (red lines in Fig. 2a) can be obtained by simply eliminating the vertices with factional values (open blue dots in Fig. 2b) [4, 5]. These vertices are then converted to tight constraints for general use. The idea to tighten MBLP problems is the same.

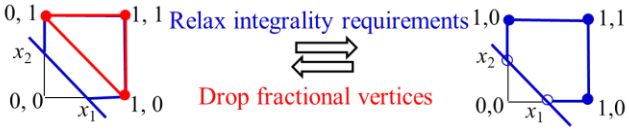

Figure 2(a). Convex hull of a BLP problem with binary variables x_1, x_2

Figure 2(b). Convex hull of its integer-relaxed problem

For the 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 is defined by blue lines in Fig. 2b.

Definition 2. For an integer-relaxed convex hull, 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 a "**fractional vertex.**" In terms of the simple example above, integral and fractional vertices are denoted by solid and open blue dots respectively in Fig. 2b.

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) are innovatively established as to be proved in Subsection 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.

a) 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 [28] well established in existing software Porta [29]. The constraints are input, and the outputs are vertices in numerical values.

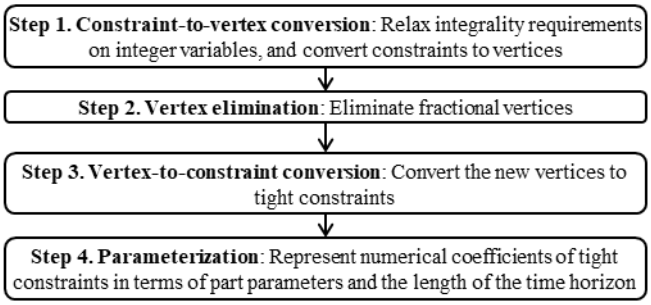

Figure 3. Flow chart of formulation tightening

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 Subsection 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 tight 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 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 relationships between numerical coefficients and part parameters and the scheduling horizon length. It is verified by checking 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 tight 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 this operation can only be processed on one machine type. Because of "non-preemption," 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 $\delta_1 + \delta_4 \leqslant 1$ and $\delta_2 + \delta_5 \leqslant 1$. 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 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 Eq. (1) and (3-5). Without integrality requirements, the constraints to Porta are shown in Fig.4.

c-b+1=3
$\delta_1+\delta_2+\delta_3+\delta_4+\delta_5+\delta_6+\delta_7+\delta_8=3$
c+8-8δ_1>=1
b-8+8δ_1<=1
c+8-8δ_2>=2
b-8+8δ_2<=2
c+8-8δ_3>=3
b-8+8δ_3<=3
c+8-8δ_4>=4
b-8+8δ_4<=4
c+8-8δ_5>=5
b-8+8δ_5<=5
c+8-8δ_6>=6
b-8+8δ_6<=6
c+8-8δ_7>=7
b-8+8δ_7<=7
c+8-8δ_8>=8
b-8+8δ_8<=8

Figure 4. Original constraints of a single operation problem

By constraint-to-vertex conversion, 1234 vertices are obtained and the last 10 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.

(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

Figure 5. Vertices of the linear programming 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

Figure 6. Integral vertices

(1) +c+2δ_1+2δ_2-7δ_3+δ_4+δ_5-8δ_6=0;
(2) -δ_1+δ_3-δ_4+δ_5-δ_7=0;
(3) -δ_2+δ_3-δ_5+δ_6-δ_8=0;
(4) -4b+3c-2δ_1-2δ_2-δ_3-δ_4-δ_5=0;
(5) +δ_3+δ_6=1;
(6) -δ_8<=0;
(7) -δ_7+δ_8<=0;
(8) -δ_6+δ_7<=0;
(9) -δ_5+δ_6-δ_8<=0;
(10) -δ_4+δ_5-δ_7+δ_8<=0;
(11) +δ_4+δ_7<=1;
(12) +δ_7<=1;

Figure 7. Tightened constraints

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

$$\delta_1 + \delta_4 + \delta_7 = 1, \quad (19a)$$

$$\delta_2 + \delta_5 + \delta_8 = 1, \quad (19b)$$

$$\delta_3 + \delta_6 = 1. \quad (19c)$$

Because of "non-preemption," 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 $\delta_1 + \delta_4 + \delta_7 \le 1$. Since it is assumed the scheduling horizon is long enough so that the operation can be processed, $\delta_1 + \delta_4 + \delta_7 = 1$ as shown in Eq. (19a). Similarly, one δ from time slots 2, 5 and 8 must be 1 as shown in Eq. (19b), and one δ from time slots 3 and 6 must be 1 as shown in Eq. (19c). Given Eq. (19) and binary requirements of δ, inequality (11) in Fig. 7 is redundant. This 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,

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

b) 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 $C_{1\text{-}2}$. 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 $\{TC_1, TC_2, C_{1\text{-}2}\}$, tighten the two-operation formulation through the four steps presented in the above subsubsection, and obtain tightened constraints across two operations as $TC_{1\text{-}2}$. Note that after parameterization, $TC_{1\text{-}2}$ 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 opearation 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, thus $\delta_{1,5} = 0$. Similarly, the earliest beginning time of operation two is 4 as operation one needs three time slots before it, thus $\delta_{2,1} = \delta_{2,2} = \delta_{2,3} = 0$.

c) 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 the following Theorem 1.

Theorem 1. For the formulation of a single operation described by Eqs. (1), (3-5), (8-9), and (11-16), the integral vertices (Definition 2) of its integer-relaxed convex hull (Definition 1) $Conv(P_{MILP\text{-}IR})$ are the vertices of the convex hull $Conv(P_{MILP})$ 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 $Conv(P_{MILP\text{-}IR})$ are the vertices of the original convex hull $Conv(P_{MILP})$ based on the theorems developed for MBLP problems in our previous work [5].

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. Since "non-preemptive" processing time requirements modeled in Eqs. (3-5) are satisfied, a contiguous time period of length $\sum_{m\in M} x_m p_m$ should be assigned to process the operation. For any time k_0 such that $1 \le k_0 \le T - \sum_{m\in M} x_m p_m + 1$, without loss of generality, it is assumed that this operation is processed during the time interval $[k_0, k_0 + \sum_{m\in M} x_m p_m - 1]$ (assigned to one machine type in set M that can process this operation). Because of Eq. (5), $\sum_{m\in M} x_m p_m$ time slots will be occupied, and because of Eqs. (3-4), 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 to 1 during these time slots, and 0 otherwise as required by the processing time requirements Eqs. (5). Therefore the terms with big number N in Eqs. (3-4) disappear. By replacing c by $b + \sum_{m\in M} x_m p_m - 1$ in Eq. (3) and combining Eqs. (3-4), the following inequality is obtained:

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

To analyze the relationship between b and k_0, Eq. (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 Eq. (21) becomes,

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

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

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

Eq. (23) can be rewritten as,

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

Combining Eqs. (22) and (24), obtain $k_0 \leqslant b \leqslant k_0$, which implies $b = k_0$. Then c can be described $k_0 + \sum_{m\in M} x_m p_m - 1$. 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 Eqs. (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 $Conv(P_{MILP\text{-}LR})$, 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 for the tightening process.

Step 2. Tightness of MILP problems

For an MBLP problem, it has been proved that, the integral vertices (Definition 2) 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 under consideration can be treated as MBLP in the tightening process, integral vertices of its integer-relaxed convex hull $Conv(P_{MILP\text{-}LPR})$ are the vertices original convex hull $Conv(P_{MILP})$, 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 δs. Thus the formulation is still tight by applying the same idea as that for the single operations.

Generalization. Beyond MILP job-shop scheduling problems under consideration, this approach also applies to other MILP problems with unique relationships between integer and binary variables.

V. Numerical Results

The above tightening method is implemented by using Porta [29]. The job-shop scheduling problems are solved by using IBM ILOG CPLEX Optimization Studio V 12.8.0.0 [30] on a PC with 2.40GHz Intel Xeon E-2286M CPU and 32G RAM. Three examples are presented. The first is to tighten 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.

***Example* 1**: Single part

a) One operation

Consider the scheduling problem with a single operation with $p = 3$ and $K = 8$ used in Subsection IV-A. Constraints (2), (3), (5) and (11) in Fig. 7 have been explored in Subsection IV-A, and inequality (12) is redundant given the binary requirements of δ. Therefore the focus is on exploring inequalities (6) to (10) and equalities (1) and (4) in Fig. 7.

1) *Inequality constraints*

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

$$\delta_2+\delta_3 \le 2(\delta_1+\delta_4) \tag{25a}$$

$$\delta_3+\delta_4 \le 2(\delta_2+\delta_5), \tag{25b}$$

$$\delta_4+\delta_5 \le 2(\delta_3+\delta_6), \tag{25c}$$

$$\delta_5+\delta_6 \le 2(\delta_4+\delta_7), \tag{25d}$$

$$\delta_6+\delta_7 \le 2(\delta_5+\delta_8). \tag{25e}$$

Eq. (25a) implies that if δ_2 and δ_3 are both 1, either δ_1 or δ_4 must be 1 because of "non-preemptive" processing time requirements, similar for Eqs. (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 time as follows,

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

Eq. (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) to (10) in Fig. 7 as a group, they can be put together in the matrix form in Eq. (27) below,

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

It is noted that δ_1 to δ_3 do not show up in the above equation. The reason is that if δ_4 to δ_8 are properly regulated to satisfy the "non-preemptive" processing time requirements by Eq. (27), then δ_1 to δ_3 are expected to satisfy the requirements too because of the processing time related tightened constraint Eq. (19). In addition, although the first row of Eq. (27), denoted as Eq. (27-1), is redundant given the binary requirements of δ, it helps put the constraints together in a square matrix form. Physical meanings of Eqs. (27-2)-(27-5) are analyzed one by one below.

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

TABLE I. Constraint Analysis for Eq.(27-2)

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

It can be seen Eq. (27-2) guarantees that if δ_8 is 1, δ_7 must be 1 as implied by the 2nd row of Table I. This is reasonable because the last possible three time slots to process the operation is 6, 7, and 8 given the processing time is 3. If the 8th time slot is taken, then the 7th must be taken too, otherwise the operation cannot be completed within the scheduling horizon. The physical meaning of Eq. (27-3) is similar, if δ_7 is 1, δ_6 must be 1.

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

TABLE II. Constraint Analysis for Eq.(27-4)

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

It can be seen Eq. (27-4) guarantees that δ_6 cannot be 1 when δ_5 and δ_8 are both 0 as implied by the 3rd 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 of the 5th time slot or the 8th is taken, then the 6th 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 Eq. (19b).

Physical meanings of Eq. (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 Eq. (27) can be intuitively shown in Fig. 8 below.

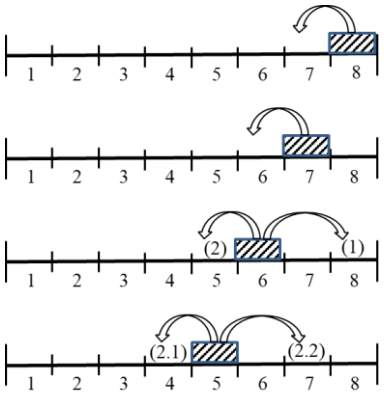

Figure 8. Intuitive description of Eq. (27)

As shown in the figure: (a) if δ_8 is 1, then δ_7 must be 1; (b) if δ_7 is 1, then δ_6 must be 1; (c) if δ_6 is 1, then one of δ_5 and δ_8 has to be 1; and (d) if δ_5 is 1, then one of δ_4 and δ_7 has to be 1. Eq. (27) with the processing time related tightened constraint Eq. (19) guarantee a contiguous time period with 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, thus time slots 6, 7 and 8 are used to process the operation. For the second situation, δ_5 is 1, and there are two possibilities again: (2.1) δ_4 is 1; or (2.2) δ_7 is 1. For the first situation, time slots 4, 5 and 6 are used to process the operation. For the second situation, time slots 5, 6 and 7 are used. Among all situations described above, a contiguous time period with length of 3 is guaranteed to process the operation. Similar analysis can be performed for other δ.

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

$$\begin{pmatrix} \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ \dots & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 \\ \dots & 0 & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 1 \\ \dots & 0 & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 \\ \dots & 0 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\ \dots & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 & 1 \\ \dots & -1 & 1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 \\ \dots & 1 & 0 & -1 & 1 & 0 & -1 & 1 & 0 & -1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} \delta_4 \\ \delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \\ \delta_9 \\ \delta_{10} \\ \delta_{11} \\ \dots \\ \delta_K \end{pmatrix} \le 0. \quad (28)$$

The entries x_{nh} of the above matrix are presented below.

Let $n + h = K - 4 + y, y \in Z$

$$x_{nh} = \begin{cases} 0, & y \le 0 \\ -1, & y > 0 \text{ and } y \bmod 3 = 1 \\ 1, & y > 0 \text{ and } y \bmod 3 = 2 \\ 0, & y > 0 \text{ and } y \bmod 3 = 0 \end{cases} \quad (29)$$

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

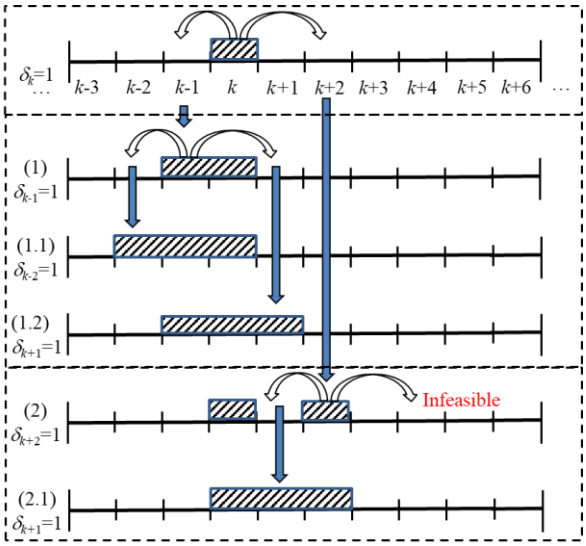

Figure 9. Intuitive description of Eq. (28)

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: (1.1) δ_{k-2} is 1; or (1.2) 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: (2.1) δ_{k+1} is 1; or (2.2) 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 Eq. (28), one of δ_{k+3} and δ_{k+6} has to be 1. This is contradictory with Eq. (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. Similar analysis can be performed for other δ.

When the process time is 2 and 4, the corresponding sets of tightened constraints for problems with longer time periods are shown in Eqs. (30) and (32), respectively. The entries of these two matrices are presented in Eqs. (31) and (33), respectively.

$$\begin{pmatrix} \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 \\ \dots & 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 1 \\ \dots & 0 & 0 & 0 & 0 & 1 & 1 & -1 & 1 & -1 \\ \dots & 0 & 0 & 0 & 1 & -1 & -1 & 1 & -1 & 1 \\ \dots & 0 & 0 & 0 & -1 & 1 & 1 & -1 & 1 & -1 \\ \dots & 0 & 0 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ \dots & 0 & -1 & 1 & -1 & 1 & 1 & -1 & 1 & -1 \\ \dots & -1 & 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} \delta_3 \\ \delta_4 \\ \delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \\ \delta_9 \\ \delta_{10} \\ \dots \\ \delta_K \end{pmatrix} \le 0, \quad (30)$$

$$P = 2, n + h = K - 3 + y, y \in Z$$

$$x_{nh} = \begin{cases} 0, & y \le 0 \\ -1, & y > 0 \text{ and } y \bmod 3 = 1 \\ 1, & y > 0 \text{ and } y \bmod 3 = 0 \end{cases} \quad (31)$$

$$\begin{pmatrix} \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ \dots & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 \\ \dots & 0 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ \dots & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 \\ \dots & 0 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 \\ \dots & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 & 0 \\ \dots & & -1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ \dots & -1 & 1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 \\ \dots & 1 & 0 & 0 & -1 & 1 & 0 & 0 & -1 & 1 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \end{pmatrix} \begin{pmatrix} \delta_5 \\ \delta_6 \\ \delta_7 \\ \delta_8 \\ \delta_9 \\ \delta_{10} \\ \delta_{11} \\ \delta_{12} \\ \dots \\ \delta_K \end{pmatrix} \le 0. \quad (32)$$

$$P = 4, n + h = K - 5 + y, y \in Z$$

$$x_{nh} = \begin{cases} 0, & y \le 0 \\ -1, & y > 0 \text{ and } y \bmod 4 = 1 \\ 1, & y > 0 \text{ and } y \bmod 4 = 2 \\ 0, & y > 0 \text{ and } y \bmod 4 = 3 \\ 0, & y > 0 \text{ and } y \bmod 4 = 0 \end{cases} \quad (33)$$

2) 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,

$$b = K - p + 1 - 2(\delta_1 + \delta_2) - (\delta_3 + \delta_4 + \delta_5) - 0(\delta_6 + \delta_7 + \delta_8), \quad (34)$$

$$c = -0(\delta_8 + \delta_7) + K\delta_6 - (\delta_5 + \delta_4) + (K-1)\delta_3 - 2(\delta_2 + \delta_1). \quad (35)$$

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_7 and δ_8 as 1 as implied in Eq. (34). When the starting of nonzero δ moves earlier, b gets smaller. The earlier the δ, the larger the negative impacts on b. The meaning of Eq. (35) is similar. The largest completion time is K with δ_6, δ_7 and 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 time as follows,

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

$$c = \sum_{m \in M_j} \sum_{n=0}^{n=\lfloor K/p_{jm} \rfloor} \begin{pmatrix} -n \left(\sum_{\tau=0}^{\tau = p_{jm} - 2 + np_{jm} + \tau < K} \delta_{jm, K - np_{jm} - \tau} \right) \\ +(K-n) \sum_{\tau=1}^{\tau = 1 + np_{jm} + p_{jm} - \tau < K} \delta_{jm, K - np_{jm} - p_{jm} + \tau} \end{pmatrix}. \quad (37)$$

Eqs. (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 Eqs. (19) and (25), the total number of vertices decreases from 1234 to 250 in a major way. With Eqs. (34-35), it is further reduced to 42. After replacing Eq. (25) by Eq. (27), the total number of vertices is 6, and all of the vertices are integral vertices, implying the formulation is tight. For the single operation scheduling problem with the processing time as 1, 2, 3, and 4, tight formulations are obtained.

b) Two operations

Now add another operation with processing time of 3 to the problem in *a*), with additional operation precedence constraint Eq. (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 Eqs. (20) and (26) obtained in the one-operation example to both operations, a total number of 23,206 vertices remain. With Eqs. (36-37) applied to both operations, 333 vertices remain. After eliminating factional vertices, there are 6 integral vertices. After vertex-to-constraint conversion, the resulting tight constraints are obtained as shown in Fig. 10.

(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.

Figure 10. Ex.1-*b)* Tightened constraints

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

$$\delta_{1,k} = 0, k \in [K - p_2 + 1, K], \quad (38)$$

$$\delta_{2,k} = 0, k \in [1, p_1]. \quad (39)$$

Since the two operations need to be completed in the scheduling horizon, the largest completion time for operation 2 is K, with beginning time of $K - p_2 + 1$. Therefore operation 1 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 Eq. (38). The meaning of Eq. (39) is similar. The smallest beginning time of operation 1 is 1, with completion time of p_1. Therefore operation 2 cannot start before p_1, and $\delta_{2,k}$ must be 0 for period of $k \in [1, p_{1m}]$.

Eqs. (38-39) 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 time as follows,

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

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

c) One operation with linearized tardiness

Now add the constraints associated with tardiness to the problem in *a*), assuming due date *d* is 2. Constraints under consideration are processing time requirements Eq. (1) and Eqs. (3-5), and tardiness constraints Eqs. (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, 6,170 vertices are obtained by constraint-to-vertex conversion. With Eqs. (19), (25), (34-35) obtained in the one-operation example, a total number of 210 vertices remain. After eliminating factional vertices, there are 8 integral vertices. After vertex-to-constraint conversion, the resulting tight constraints are shown in Fig. 11.

(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) $-\delta_5-\delta_6-\delta_7-2\delta_8+T<=2$.

Figure 11. Ex.1-*c)* Tightened constraints

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

$$T = \delta_3 + \delta_4 + \delta_5 + 2\delta_6 + 2\delta_7 + 2\delta_8. \quad (42)$$

Since the processing time is 3 and the due date is 2, the smallest tardiness is 1 with δ_1, δ_2 and δ_3 as 1, and the largest tardiness is 6 with δ_6, δ_7 and δ_8 as 1 as implied in Eq. (42). When the starting of nonzero δ moves earlier, *T* gets smaller. The earlier the δ, the smaller the impacts on *T*. Eq. (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, Eq. (42) is converted to the following constraint in a generic form,

$$T = \sum_{m \in M_j} \begin{cases} \sum_{n=1}^{\lfloor (K-d+1)/p_{Jm} \rfloor + 1} \left(\begin{array}{l} (p_{Jm} - d + n - 1) \sum_{\tau=0}^{p_{Jm}:p_{Jm}n+\tau \le K} \delta_{Jm, pn+\tau} \\ + n \sum_{\tau=1}^{p_{Jm}-1:p_{Jm}n+\tau \le K} \delta_{Jm, p_{Jm}n+\tau} \end{array} \right), & p_{Jm} \ge d \\ \sum_{n=1}^{\lfloor (K-d+1)/p_{Jm} \rfloor + 1} \left(n \sum_{\tau=1}^{p_{Jm}:d+p_{Jm}(n-1)+\tau \le K} \delta_{Jm, d+p_{Jm}(n-1)+\tau} \right), & p_{Jm} < d. \end{cases} \quad (43)$$

The tightened constraints obtained in *a)*, *b)*, and *c)* 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, and whose due date is positive.

Example 2: Medium-sized problems

This medium-sized example is to demonstrate effectiveness and robustness of formulation tightening. The instance is created based on the first 89 parts and all machines in [8]. 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 to 6 machines with the same function. Assume 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 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. In CPLEX, optimization stops when computational time reaches the pre-set stop time or the relative mixed-integer programming gap (relative difference between the objectives of the optimal relaxed solution and current integer solution) falls below the pre-set gap. Here the stop MIP gap is set as 0.01% and there is no time limit.

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

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

According to Table III, the solutions with the same quality are obtained under different formulations. The CPU, solving, cutting and branching time is much reduced by adding new tightened constraints Eqs. (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 seconds, while the time on cutting and branching is 20s and 866s, respectively. By adding new tightened constraints Eqs. (20), (26), (36), (37), (40), (41) and (43), a similar feasible solution is obtained in 4.7s, while the cutting time is 1.1s and there are no branching operations. With tightened constraints less time is required to obtain the same MIP gap of 0.01% as compared with the standard formulation.

When replacing Eq. (26) by Eqs. (28), (30) and (32) (the formulation becomes tighter), a similar solution is obtained in 6s, with cutting time as 0.9s. 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 [30], not on the entire feasible region as mentioned in Section I. Therefore more tightened constraints may not guarantee better computational efficiency. There is a trade-off 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 due date, while tardiness weights are the same as the original problem. The results are shown in Fig. 13.

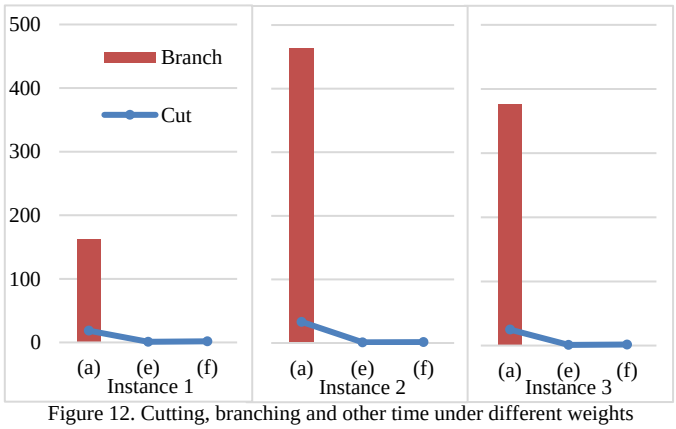

Figure 12. Cutting, branching and other time under different weights

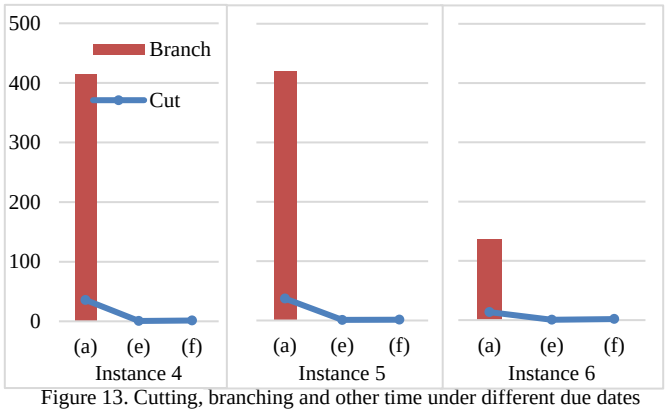

Figure 13. Cutting, branching and other time under different due dates

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 demonstrates effectiveness and robustness of our formulation tightening.

Example 3: Large-sized problems

This large-sized example is to demonstrate performance of tightened single-part formulations. The instance is created based on all 127 parts and all machines in [8]. 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 in Example 2. Before and after adding tightened constraints, the job-shop scheduling problems are solved by using branch-and-cut, and results are shown in Table IV. Since the problem is more complicated than Example 2, two stopping criteria are considered: 1200 second (s) CPU time or 0.01% MIP gap.

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

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

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

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

According to Table V, both the solution quality and computational efficiency is significantly improved by adding new tightened constraints. With the standard formulation, no feasible solution can be found in 20 minutes. By adding new tightened constraints, a feasible solution with a MIP gap of 0.01% is obtained in 50s.

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

VI. CONCLUSION

In this paper, 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 tight constraints with coefficients characterized by part parameters and the length of

the scheduling horizon. The tightening process only needs to performed once, 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 MILP problems with unique relationships 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.

VII. Appendix

A. Physical meanings of tight constraints

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

TABLE VI. Constraint Analysis for Eq. (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
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	Yes
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
11	1	0	-	1	0	-1 + 0 -1 +0 ⩽ 0	Yes
12	1	0	-	1	1	-1 + 0 -1 +1 ⩽ 0	Yes
13	1	1	-	0	0	-1 + 1 -0 +0 ⩽ 0	Yes
14	1	1	-	0	1	-1 -1 + 0 -1 ⩽ 0	Yes
15	1	1	-	1	0	-1 -1 + 1 -0 ⩽ 0	Yes
16	1	1	-	1	1	-1 + 1 -1 +1 ⩽ 0	Yes

It can be seen that Eq. (27-5) guarantees that δ_5 cannot be 1 when δ_4 when δ_7 are both 0 as implied by the 5th 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 4th time slot or the 7th is taken, the 5th cannot be taken based on the processing time requirement. Note Cases 2 and 10 in Table VI are not feasible because if δ_8 is 1, δ_7 must be 1 as guaranteed by Eq. (27-2). Cases 6, 8, 11, 12, 14, and 15 are not feasible as guaranteed by Eq. (19). For Case 3, since δ_6 cannot be 1 when δ_5 and δ_8 are both 0 as guaranteed by Eq. (27-4), δ_6 has to be 0. However, if δ_7 is 1, δ_6 must be 1 as guaranteed by Eq. (27-3). Therefore Case 3 is infeasible too.

References

[1] P. Brucker, Scheduling Algorithms, 5th ed, Springer-Verlag, Berlin, 2006.

[2] P. Brucker, "The job-shop problem: Old and new challenges." in *Proceedings of the 3rd Multidisciplinary International Conference on Scheduling: Theory and Applications*, pp. 15-22, 2007.

[3] D. P. Bertsekas, Nonlinear programming, 3rd ed, Athena scientific, 2016.

[4] B. Yan, P. B. Luh, E. Litvinov, T. Zheng, D. Schiro, M. A. Bragin, F. Zhao, J. Zhao, and I. Lelic "A Systematical Approach to Tighten Unit Commitment Formulations," in *Proceeding of 2018 IEEE Power and Energy Society General Meeting*.

[5] B. Yan, P. B. Luh, T. Zheng, D. Schiro, M. A. Bragin, F. Zhao, J. Zhao, and I. Lelic "A Systematic Formulation Tightening Approach for Unit Commitment Problems," *IEEE Transactions on Power Systems*, Vol. 35, Issue 1, pp. 782 - 794, 2019.

[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, Issue 4, pp. 3387 - 3393, 2018.

[7] T. Yamada and N. Ryohei Nakano, "Job shop scheduling," *IEE control Engineering series 55*, pp. 134-134, 1997.

[8] 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.

[9] T. Sun, P. B. Luh, and L. Min, "Lagrangian relaxation for complex job shop scheduling," in *Proceedings of 2006 IEEE International Conference on Robotics and Automation*, pp. 1432 - 1437, 2006.

[10] T. Nishi, Y. Hiranaka, and M. Inuiguchi, "Lagrangian relaxation with cut generation for hybrid flowshop scheduling problems to minimize the total weighted tardiness," *Computers & Operations Research*, Vol. 37, Issue 1, pp. 189-198, 2010.

[11] 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," *European Journal of Operational Research*, Vol. 236, Issue 1, pp. 51-60, 2014.

[12] E. Asadi-Gangraj, "Lagrangian relaxation approach to minimize makespan for hybrid flow shop scheduling problem with unrelated parallel machines," *Scientia Iranica*, Vol. 25, Issue 6, pp. 3765-3775, 2018.

[13] 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 Computing and Applications*, Vol. 30, Issue 1, pp.193-209, 2018.

[14] R. Z. Ríos-Mercado and J. F. Bard, "Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups," *Computers & Operations Research*, Vol. 25, No. 5, pp. 351-366, 1998.

[15] J. C. H. Pan and J. S. Chen, "Mixed binary integer programming formulations for the reentrant job shop scheduling problem," *Computers & Operations Research*, Vol. 32, No. 5, pp.1197-1212, 2005.

[16] 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.

[17] C. Özgüven, Z. 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," *Applied Mathematical Modelling*, Vol. 36, Issue 2, pp.846-858, 2012.

[18] M. Karimi-Nasab and S. M. Seyedhoseini, "Multi-Level Lot Sizing and Job Shop Scheduling with Compressible Process Times: A Cutting Plane Approach," *European Journal of Operational Research*, Vol. 231, pp. 598-616, 2013.

[19] J. Cheng, F. Chu, and M. Zhou, "An improved model for parallel machine scheduling under time-of-use electricity price," *IEEE Transactions on Automation Science and Engineering*, Vol. 15, No. 2, pp. 896-899, 2017.

[20] 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 Transactions on Engineering Management*, Vol. 65, No. 3, pp. 487-504, 2018.

[21] 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.

[22] 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.

[23] 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, pp.106347, 2020.

[24] B. Yan, H. Y. Chen, P. B. Luh, S. Wang, and J. Chang, "Litho machine scheduling with convex hull analyses," *IEEE Transactions on Automation Science and Engineering*, Vol. 10, No. 4, pp. 928-937, 2013.
[25] H. D. Sherali and P. J. Driscoll, "On tightening the relaxations of Miller–Tucker–Zemlin formulations for asymmetric traveling salesman problems," *Operations Research*, Vol. 50, pp. 656-669, 2002.
[26] D. Bienstock and B. McClosky, "Tightening simple mixed-integer sets with guaranteed bounds," *Mathematical programming*, Vol. 133, pp. 337-363, 2012.
[27] E. M. L. Beale and J. J. H. Forrest, "Global optimization using special ordered sets," *Mathematical Programming*, Vol. 10, No. 1, pp. 52-69, 1976.
[28] G. B. Dantzig and B. Curtis Eaves, "Fourier-Motzkin elimination and its dual," *Journal of Combinatorial Theory, Series A*, Vol. 14, No. 3, pp. 288-297, 1973.
[29] Heidelberg University, http://www.iwr.uni-heidelberg.de/groups/comopt/software/PORTA/
[30] IBM ILG CPLEX V 12.1 User's Manual.

Bing Yan (S'11-M'17) received her B.S. degree from Renmin University of China in 2010, M.S. and Ph.D. degrees from University of Connecticut in 2012 and 2016, respectively. She is currently an Assistant Professor in the Department of Electrical and Microelectronic Engineering, Rochester Institute of Technology. Before joining Rochester Institute of Technology, she was an Assistant Research Professor in 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 (S'11-M'17) received his B.S. and M.S. degrees in Mathematics from the Voronezh State University, Russia, in 2004, the M.S. degree in Physics and Astronomy from the University of Nebraska-Lincoln, USA, in 2006, and the M.S. and Ph.D. degree in Electrical and Computer Engineering from the University of Connecticut, USA, in 2014 and 2016, respectively. He is 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 (S'77–M'80–SM'91–F'95-LF'16) received his B.S. degree from National Taiwan University, M.S. degree from M.I.T., and Ph.D. degree from Harvard University. He has been with the University of Connecticut since 1980, and is a Board of Trustees Distinguished Professor and the SNET Professor of communications & information technologies. His interests include intelligent manufacturing, energy smart buildings, and smart grid. He is a life fellow of IEEE, the Chair of IEEE TAB Periodicals Review and Advisory Committee 2020-21, the Chair of IEEE TAB Periodicals Committee 2018-19, and the Founding Editor-in-Chief of IEEE Transactions on Automation Science and Engineering.