Novel Formulation and Resolution of Job-Shop Scheduling Problems

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

***Abstract*–Job-shop scheduling is an important problem in planning and operation of manufacturing systems. For such difficult problems to be solved daily within short amounts of time, the only practical goal is to obtain near-optimal solutions with quantifiable quality fast. Recent developments of powerful Mixed-Integer Linear Programming (MILP) methods such as branch-and-cut provide an opportunity for a fresh perspective at new at effective MILP formulation and resolution of the problem. Moreover, formulation tightening is critically important since if constraints directly delineate the convex hull of an MILP problem, it can be solved by linear programming without combinatorial difficulties. To achieve the above goal, three major contributions of this paper are: 1) to efficiently formulate the problems in an MILP form; 2) to develop a novel systematic formulation tightening approach for the first time; and 3) to establish a decomposition and coordination method with exponential reduction of complexity and accelerated convergence to efficiently solve the problem. Testing results show that our formulation tightening is effective in terms of computational efficiency and solution quality. With decomposition, time-consuming branching is no longer needed when solving subproblems, and coordination is effective. For dynamic job-shop scheduling problems, schedule can be regenerated fast based on previous scheduling results. This work opens up new directions for more exploration to efficiently solve MILP problems.**

***Index terms*–Manufacturing, job-shop scheduling, mixed-integer linear programming, formulation tightening, branch-and-cut, surrogate absolute-value Lagrangian relaxation**

I. Introduction

In planning and operation of manufacturing systems, job-shop scheduling is an important yet a difficult problem.

In a job shop, each part has several operations to be processed, and each operation requires a fixed amount of time on one machine of a given set [1]. The problem is to find a schedule to process parts on available machines to minimize the required objective such as the total tardiness, while satisfying precedence and processing time constraints. For such difficult problems to be solved daily within short amounts of time, the only practical goal is to obtain near-optimal solutions with quantifiable quality fast. Recent development of powerful Mixed-Integer Linear Programming (MILP) methods such as branch-and-cut provide an opportunity for a fresh perspective at effective MILP formulation and resolution. Moreover, formulation tightening is critically important since if constraints directly delineate the convex hull of an MILP problem, it can be solved by Linear Programming (LP) without combinatorial difficulties.

As reviewed in Section II, metaheuristics have been frequently used to solve job-shop scheduling problems because of low computational requirements. However, if a solution is obtained, its quality cannot be quantified, it is thus very difficult to systemically improve the quality. To overcome complexity difficulties of standard job-shop scheduling formulations [2], formulations with "separable structures" were established to be effectively exploited by Lagrangian relaxation (LR) [3]. LR is a decomposition and coordination approach with major distinguishing features such as near-optimal solutions with quantifiable quality. However, traditional LR has slow convergence. Also with nonlinear formulations, the problem cannot be solved by MILP methods.

This paper is a pioneering effort to obtain near-optimal schedules with quantifiable quality for large-scale job-shops by three major contributions. The first one is to reformulate the problem in an MILP form with efficient linearization to make effective use of popular MILP methods in Section III. Complicated features such as batching and sequence-dependent setups are not considered. The second contribution is to develop a novel systematic approach to tighten MILP formulations for the first time based on a novel integration of "constraint-and-vertex conversion" and "vertex projection" processes in Section IV. Inspired by penalization principles of Augmented Lagrangian relaxation, the last contribution is to develop a decomposition and coordination method with exponential reduction of complexity and accelerated convergence in Section V. It is based on our recently developed Surrogate Absolute-Value Lagrangian relaxation (SAVLR). Although absolute-values are not differentiable, they have the advantage of being exactly linearizable through introduction of few extra variables and constraints.

The resulting method is implemented by using CPLEX, and two examples are presented in Section VI. The first is to demonstrate the effectiveness of formulation tightening. The second is to show computational efficiency and scalability of SAVLR combined with branch-and-cut.

II. Literature Review

Developing efficient formulation and resolution of job-shop scheduling is challenging because of its complex characteristics, large sizes of practical problems, and high combinatorial complexity. Existing problem formulations,

Manuscript received: February, 15, 2018; Revised May, 9, 2018; Accepted June, 7, 2018. This paper was recommended for publication by Editor Kevin Lynch upon evaluation of the Associate Editor and Reviewers' comments. This work is supported by the National Science Foundation under grant ECCS-1509666. 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 the NSF.

Bing Yan, Mikhail A. Bragin, and Peter B. Luh are with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269-4157, USA (bing.yan@uconn.edu, mikhail.bragin@uconn.edu, and Peter.Luh@uconn.edu).

Digital Object Identifier (DOI): see top of this page.

formulation tightening, and solution methodologies are reviewed in Subsections A, B and C, respectively.

A. Problem formulation

Standard job-shop scheduling formulations are complex because of the large number of decision variables and constraints [2]. For these problems, formulations with "separable structures" were established and efficiently exploited by Lagrangian relaxation (LR) in [3]. However, the model of [3], as well as those of [4, 5], is nonlinear. Integer linear programming (ILP) models have also been considered [6, 7]. With sequence-dependent setups, ILP models have been developed by modeling an immediate job successor of a predecessor through the use of additional variables in [6]. In [8], an ILP scheduling formulation was developed for high-volume and low-variety manufacturing. However, with a large number of decision variables and constraints, those models cannot be effectively solved by MILP methods.

B. Formulation tightening

Formulation tightening is much overlooked but critically important since if constraints directly delineate the convex hull of an MILP problem, the problem can be solved by LP without combinatorial difficulties. However, the problem of obtaining the convex hull is fundamentally difficult and there are no clear ways to tighten formulations. In the literature, a few tightened constraints were presented without providing how they were obtained. In [6], a number of valid inequalities, including facet-defining cuts were obtained. In [7], a number of valid cuts were developed based on problem structures.

C. Solution methodologies

Metaheuristics such as Tabu search [9-11], simulated annealing [12], evolutionary algorithms [13, 14], and Particle swarm [15, 16] have been widely used because of their low computational requirements. A two-step tabu search algorithm was established in [11] to: (1) search for the best sequence of job operations and (2) find the best choice of machine alternatives. In [13], a multi-objective evolutionary algorithm-based proactive-reactive method was developed. Particle swarm optimization was distributed into a multi-agent system to decentralize decisions and to make sure such that each entity participates in the resolution of the whole problem in [15]. However, if a solution is obtained, its quality cannot be quantified, and it is difficult to systemically improve it.

Branch-and-cut (B&C) has also been used for job-shop scheduling problems with ILP models [6, 7]. In B&C, integrality requirements on integer variables are first relaxed, and the problem is solved by LP. If the values of all integer decision variables are integers, the solution is optimal to the original problem. If not, B&C attempts to obtain the convex hull by cutting off LP regions without cutting off feasible solutions by "valid inequalities" (or cuts). If the convex hull is obtained, the problem can be directly solved by LP without combinatorial difficulties. Otherwise, the method relies on time-consuming branching operations. Without exploiting "local" problem features, all constraints are treated as "global," affecting the entire solution process and leading to very slow convergence. In [6], the problem was solved by using CPLEX. However, because of the above difficulties, for a 10-part and 8-machine problem, a solution with a 26.7% Mixed-Integer Programming (MIP) gap was found after one hour. In [7], performance of the valid inequalities was investigated through testing using randomly-generated datasets in CPLEX. In [8], a two-phase approach was established in the framework of B&C.

By exploiting the beautiful property of exponential reduction of complexity upon decomposition, Lagrangian relaxation (LR) has also been widely used with major distinguishing features such as near-optimal solutions with quantifiable quality [3-5]. With separable structures, the problem was decomposed into part subproblems after relaxing machine capacity constraints in [3]. Subproblems were solved by dynamic programming, and solutions were coordinated by updating multipliers based on levels of constraint violations (subgradient directions). There are few other research papers on job-shop scheduling using LR [4, 5]. However, standard LR suffers from major difficulties, e.g., high computational effort, significant multiplier zigzagging, and the requirement of optimal dual value for convergence proof and practical implementations. As a result, the overall convergence may be very slow. These difficulties have been overcome by our recent surrogate Lagrangian relaxation (SLR), where the solution of one or few subproblems is sufficient to update multipliers [17]. Moreover, convergence has been proved without requiring the optimal dual value.

Inspired by fast convergence of Augmented Lagrangian relaxation, quadratic penalty terms with subsequent linear approximation was used to significantly improve the reduction of constraint violation [18]. Convergence was further improved by using "absolute-value" penalty functions [19]. Although not differentiable, such penalties have advantage of being exactly linearizable through the introduction of a few extra variables and constraints.

III. Problem Formulation

As reviewed in Section II, job-shop scheduling formulations with "separable structures" in [3] are nonlinear. To make effective use of powerful MILP solvers, an MILP formulation is established in this section.

A. Machine capacity constraints

Consider a job shop with M types of machines indexed by m. In the shop, I parts indexed by i need to be processed, and each part requires J_i operations indexed by j, where operation j of part i is denoted by (i, j). Assuming that the time horizon is long enough to process all parts required, the horizon is discretized into T time slots indexed by t. To capture whether an operation j for a part i is active or not at time t, a set of binary variables δ_{ijt} with three indices is introduced as follows:

$$\delta_{ijt} = \begin{cases} 1, & \text{if operation } j \text{ of part } i \text{ is active at time } t; \\ 0, & \text{otherwise.} \end{cases}$$

For each machine type m, the total number of active parts cannot exceed its capacity M_{ij} at any time slot, i.e.,

$$\sum_{\forall (i,j) \in O_m} \delta_{ijt} \le M_{ij}, \forall m, \forall t. \tag{1}$$

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

B. Processing time requirements

Let b_{ij} and c_{ij} denote the beginning and completion time of operation (i, j). They are integer decision variables linked

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2850056, IEEE Robotics and Automation Letters

through processing time requirements. Part processing is assumed to be "non-preemptive" so that a contiguous time block of length p_{ij} is needed to process operation (i, j), i.e.,

$$c_{ij} = b_{ij} + p_{ij} - 1, \forall i, \forall j. \quad (2)$$

Within $[b_{ij}, c_{ij}]$, δ_{ijt} must be 1, and 0 otherwise, i.e.,

$$\delta_{ijt} = \begin{cases} 1, & \text{if } b_{ij} \le t \le c_{ij}; \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

The above logical constraint can be linearized by the standard big-M method [20]. Since the pure big-M method may not be efficient, our idea is to linearize (3) as follows:

$$t \le c_{ij} + T\left(1 - \delta_{ijt}\right), t \ge b_{ij} - T\left(1 - \delta_{ijt}\right), \forall i, \forall j, \forall t; \quad (4)$$

$$\sum_t \delta_{ijt} = p_{ij}, \forall i, \forall j. \quad (5)$$

The above constraints guarantee that $\delta_{ijt} = 1$ iff $b_{ij} \leqslant t \leqslant c_{ij}$; and $\delta_{ijt} = 0$ when $t < b_{ij}$ or $t > c_{ij}$. Therefore linear constraints (4) and (5) are equal to the logical constraint (3).

C. Operation precedence constraints

It is assumed there is a fixed sequence of operations for a particular part. Operation precedence constraints require that operation $(i, j+1)$ cannot start before (i, j) is completed, i.e.,

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

Also, operation j cannot start before part i is arrived, i.e.,

$$b_{ij} \ge a_{ij}, \forall i, \forall j. \quad (7)$$

In the above, a_{ij} is the arrival time of operation (i, j).

D. Objective function

The objective function is to minimize the total weighted tardiness as described below:

$$\sum_i \left(\omega_i \cdot \max(c_{iJ_i} - d_i, 0)\right). \quad (8)$$

Here, ω_i is a weight for part i, and d_i denotes its due date. The tardiness function is modeled by a piecewise-linear function and linearized by special ordered set techniques [21]. The upper and lower bounds for c_{iJi} - d_i are p_{ij} - d_i and T - d_i, with corresponding tardiness of 0 and T - d_i. For this function, the three break points are p_{ij} - d_i, 0 and T - d_i (if p_{ij} - $d_i < 0 < T$ - d_i). Based on special ordered set techniques, three continuous variables w_1, w_2, and w_3 are used for each i to denote weights of these three points and three binary variables Y_1, Y_2 and Y_3 to restrict the upper bound of these weights. After the conversion, the above formulation is purely linear. Here and later in the paper, the *max* function is kept for compactness of notation.

IV. Formulation Tightening

In this section, a systematic formulation tightening approach is developed through a novel integration of "constraint-and-vertex conversion" and "vertex projection" processes based on our previous work on unit commitment problems in power systems [22]. The linearity of the objective function is important, but irrelevant for formulation tightening. The goal is to tighten a single part formulation as system-wide machine capacity constraints will be relaxed in Section V.

Given part parameters (processing time p and arrival time a) in numerical values, the idea is to relax integrality requirements on discrete decision variables, and generate vertices from constraints of the resulting LP-relaxed problem in numerical values by using linear algebra with algorithms well established and software available [23]. If all integer decision variables are integers at all vertices, then the formulation is tight. If not, those non-integer values are rounded up or down to nearest feasible integers - essentially projecting vertices onto the original convex hull. These projected vertices are converted back to constraints, again by using software, and the resulting formulation should be tight.

If there are too many non-integer vertices, the process can be carried out in an iterative manner, with a few vertices projected onto the original convex hull at each iteration. The process terminates when all vertices are feasible, i.e., integer decision variables are integers for all vertices. If the process is stopped before termination, the formulation is not tight, but should be tighter as compared to the original one.

For illustration purposes, consider a problem with one part, one operation and a scheduling horizon of 7 ($T = 7$), subject to processing time requirements (2), (4) and (5) only. Decision variables are processing status of part δ_t (binary), beginning time b (integer), and completion time c (integer). For this problem with $p = 3$ and $a = 1$, after relaxing integrality requirements, the constraints are shown in Fig. 1 (a) (x_1: b; x_2: c; x_3 - x_9: δ_1 - δ_7). By constraint-to-vertex conversion, 529 vertices are obtained with some of them shown in Fig. 1 (b), and there are only 5 integer vertices out of 529.

```
DIM = 9

LOWER_BOUNDS
1 1 0 0 0 0 0 0 0
UPPER_BOUNDS
7 7 1 1 1 1 1 1 1
VALID
1 3 1 1 1 0 0 0 0

INEQUALITIES_SECTION
x2-x1+1=3
x3+x4+x5+x6+x7+x8+x9=3


x2+7-7x3>=1
x1-7+7x3<=1
x2+7-7x4>=2
x1-7+7x4<=2
x2+7-7x5>=3
x1-7+7x5<=3
x2+7-7x6>=4
x1-7+7x6<=4
x2+7-7x7>=5
x1-7+7x7<=5
x2+7-7x8>=6
x1-7+7x8<=6
x2+7-7x9>=7
x1-7+7x9<=7
```

Figure 1 (a): Original constraints

```
(500)   1   3   1   0   1   0  5/ 7   0  2/ 7
(501)   1   3   1   0   1   0  5/ 7  2/ 7   0
(502)   1   3   1   0   1  2/ 7  5/ 7   0   0
(503)   1   3   1   0   1  3/ 7   0  4/ 7   0
(504)   1   3   1   0   1  4/ 7   0   0  3/ 7
(505)   1   3   1   0   1  6/ 7   0   0  1/ 7
(506)   1   3   1   0   1  6/ 7   0  1/ 7   0
(507)   1   3   1   0   1  6/ 7  1/ 7   0   0
(508)   1   3   1  1/ 7  1  6/ 7   0   0   0
(509)   1   3   1  2/ 7  1   0  5/ 7   0   0
(510)   1   3   1  3/ 7  1   0   0  4/ 7   0
(511)   1   3   1  4/ 7  1   0   0   0  3/ 7
(512)   1   3   1   1   1   0   0  4/ 7  3/ 7
(513)   1   3   1   1   1   0   0  3/ 7  4/ 7   0
(514)   1   3   1   1   1   0   0  4/ 7   0  3/ 7
(515)   1   3   1   1   1   0   0  5/ 7   0  2/ 7
(516)   1   3   1   1   1   0   0  5/ 7  2/ 7   0
(517)   1   3   1   1   0  2/ 7  5/ 7   0   0
(518)   1   3   1   1   0  3/ 7   0  4/ 7   0
(519)   1   3   1   1   0  4/ 7   0   0  3/ 7
(520)   1   3   1   1   0  6/ 7   0   0  1/ 7
(521)   1   3   1   1   0  6/ 7   0  1/ 7   0
(522)   1   3   1   1   0  6/ 7  1/ 7   0   0
(523)   1   3   1   1  1/ 7  6/ 7   0   0   0
(524)   1   3   1   1  2/ 7   0  5/ 7   0   0
(525)   1   3   1   1  3/ 7   0   0  4/ 7   0
(526)   1   3   1   1  4/ 7   0   0   0  3/ 7
(527)   3   5   0   0   1   1   1   0   0
(528)   2   4   0   1   1   1   0   0   0
(529)   1   3   1   1   1   0   0   0   0

END
```

Figure 1 (b): Vertices

New vertices are obtained by projection as shown in Fig. 2 (a), and constraints are generated through "vertex-to-constraint conversion," as shown in Fig. 2 (b) below.

```
DIM = 9

CONV_SECTION
(1) 1 3   1   1   1   0   0 0 0
(2) 2 4   0   1   1   1   0 0 0
(3) 3 5   0   0   1   1   1 0 0
(4) 4 6   0   0   0   1   1 1 0
(5) 5 7   0   0   0   0   1 1 1
END
```

Figure 2 (a): Projected vertices

```
DIM = 9

VALID
5 7 0 0 0 0 1 1 1

INEQUALITIES_SECTION
(  1)     + x2+2x3-6x4+ x5+ x6-7x7       == 0
(  2)             + x4- x5    + x7-x8     == 0
(  3)         - x3    + x5- x6    +x8-x9  == 0
(  4) -7x1+5x2-4x3-2x4-2x5-2x6            == 0
(  5)                 + x5        +x8     == 1

(  1)          -x9 <= 0
(  2)    -x7+x8    <= 0
(  3)       -x8+x9 <= 0
(  4) -x6+x7   -x9 <= 0
(  5) +x6      +x9 <= 1

END
```

Figure 2 (b): Tightened constraints

Equality constraints (2), (3) and (5) in Fig. 2(b) can be converted to a set of tightened constraints as follows,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2850056, IEEE Robotics and Automation Letters

$$\delta_1+\delta_4+\delta_7=\delta_2+\delta_5=\delta_3+\delta_6=1. \quad (9)$$

Since the processing time is 3, three consecutive δ must be 1. Therefore, one of δ from time slots 1, 4, and 7 has to be 1 as implied in Eq. (9), same for time slots 2 and 5, and 3 and 6.

Inequality constraint (4) in Fig. 2(b) can be converted to another set of tightened constraints is obtained as follows,

$$\delta_2+\delta_3\le 2(\delta_1+\delta_4),\delta_3+\delta_4\le 2(\delta_2+\delta_5),$$
$$\delta_4+\delta_5\le 2(\delta_3+\delta_6),\delta_5+\delta_6\le 2(\delta_4+\delta_7). \quad (10)$$

The first inequality implies that if δ_2 and δ_3 are both 1, either δ_1 or δ_4 must be 1, similar for the other three constraints.

The above tightened constraints directly constrain variables δ_1 - δ_7 and tighten the formulation, but can hardly be obtained manually without going through this tightening process. With those constraints, the total number of vertices is dramatically reduced from 529 to 60, and the resulting formulation is much tighter than the original one (not tight yet). The above tightened constraints can be extended to other parts/operations whose processing time is not 3.

Coefficients of these tightened constraints, however, are in numerical values, not generic in terms of part parameters (i.e., p and a). To overcome this issue, the idea is to analyze physical meanings of constraints under possible part statuses (i.e., active or not), and convert constraints back to generic forms while remaining meaningful under all possible part statuses.

V. Solution Methodology

This section is on the solution methodology based on our recent Surrogate Absolute-Value Lagrangian Relaxation (SAVLR) with accelerated convergence [19].

A. Standard Lagrangian Relaxation

In standard Lagrangian Relaxation, after relaxing system-wide machine capacity coupling constraints (1) by using Lagrangian multipliers λ, the relaxed problem becomes:

$$\min_{c,\delta}\left\{\sum_i\left(\omega_i\cdot\max\left(c_{iJ_i}-d_i,0\right)\right)+\sum_{t,m}\lambda_{tm}^k\left(\sum_{\forall(i,j)\in O_m}\delta_{ijt}-M_{ij}\right)\right\}. \quad (11)$$

With system-wide constraints relaxed, the relaxed problem can be decomposed into individual part subproblems:

$$\min_{c,\delta}\left\{\omega_i\cdot\max\left(c_{iJ_i}-d_i,0\right)+\sum_{t,m}\lambda_{tm}^k\left(\sum_{(i,j)\in O_m}\delta_{ijt}\right)\right\}. \quad (12)$$

Complexity of each subproblem (12) is drastically reduced as compared to that of the original problem. Multipliers are updated based on appropriately chosen stepsizes and constraint violations (subgradient directions) as:

$$\lambda_{tm}^{k+1}=\lambda_{tm}^k+s^k\left(\sum_{(i,j)\in O_m}\delta_{ijt}^k-M_{ij}\right), \quad (13)$$

where ${\delta_{ijt}}^k$ is latest available value of decision variable δ_{ijt}. However, in order to obtain subgradient directions, all subproblems need to be solved to optimality. Because of this, as explained in Section II, standard LR suffers from high computational effort, and multipliers may suffer from zigzagging, resulting in slow convergence.

B. Surrogate Lagrangian Relaxation (SLR) [17]

In the method, computational effort is reduced by solving one or few subproblems at a time before updating multipliers. Within the multiplier-updating formula (13), instead of subgradient directions, *surrogate subgradient directions* are used and defined as:

$$\tilde{g}(\delta^k)=\sum_{\forall(i,j)\in O_m}\delta_{ijt}^k-M_{ij}. \quad (14)$$

Since not all subproblems are solved at a time, surrogate directions do not change drastically from one iteration to the next and zigzagging difficulties are thus alleviated.

Within SLR, convergence is guaranteed without using the optimal dual value by using the following stepsizes:

$$s^k=\alpha_k\frac{s^{k-1}\left\|\tilde{g}\left(\delta^{k-1}\right)\right\|_2}{\left\|\tilde{g}\left(\delta^k\right)\right\|_2},\ 0<\alpha_k<1, \quad (15)$$

where δ represents all the decision variables δ_{ijt}, and

$$\alpha_k=1-\frac{1}{Mk^\rho},\ \rho=1-\frac{1}{k^r},\ M\ge 1,\ 0<r<1. \quad (16)$$

C. Surrogate Absolute-Value Lagrangian Relaxation (SAVLR) [19]

To accelerate reduction of constraint violations and improve convergence of SLR, violations of relaxed machine capacity constraints (1) are penalized by using "absolute-value" penalty terms with positive penalty coefficients v^k. The "absolute-value" relaxed problem is formulated:

$$\min_{c,\delta,z}\left\{\begin{array}{l}\sum_i\left(\omega_i\cdot\max\left(c_{iJ_i}-d_i,0\right)\right)+\lambda_{tm}^k\left(\sum_{\forall(i,j)\in O_m}\delta_{ijt}+z_{ij}-M_{ij}\right)+\\ \frac{v^k}{2}\left(\left|\sum_{\forall(i,j)\in O_m}\delta_{ijt}+z_{ij}-M_{ij}\right|\right)\end{array}\right\}. \quad (17)$$

where z_{ij} are real-valued non-negative slack variables. Subproblems can be formed based on (17) by selecting variables associated with one part i as decision variables and fixing decision variables associated with other subproblems at previously obtained values as:

$$\min_{c,\delta,z}\left\{\begin{array}{l}\omega_i\cdot\max\left(c_{iJ_i}-d_i,0\right)+\sum_{t,m}\lambda_{tm}^k\left(\sum_{(i,j)\in O_m}\delta_{ijt}+z_{ij}\right)+\\ \frac{v^k}{2}\sum_{t,m}\left(\left|\sum_{(i'\ne i,j)\in O_m}\delta_{i'jt}^k+\sum_{(i,j)\in O_m}\delta_{ijt}+z_{ij}-M_{ij}\right|\right)\end{array}\right\}. \quad (18)$$

Following standard practice[1], subproblems are linearized exactly and subproblem i can be written in MILP form after introducing continuous decision variables q_{tm} as:

$$\min_{c_i,\delta_i,z}\left\{\begin{array}{l}\omega_i\cdot\max\left(c_{iJ_i}-d_i,0\right)+\\ \sum_{t,m}\lambda_{tm}^k\left(\sum_{(i,j)\in O_m}\delta_{ijt}+z_{ij}\right)+\frac{v^k}{2}\sum_{t,m}q_{tm}\end{array}\right\}, \quad (19)$$

$$s.t.(2),\ (4)-(7),-q_{tm}\le\sum_{(i'\ne i,j)\in O_m}\delta_{i'jt}^k+\sum_{(i,j)\in O_m}\delta_{ijt}+z_{ij}-M_{ij}\le q_{tm}. \quad (20)$$

Subproblems (19)-(20) are linear and combinatorial, and are solved by using branch-and-cut. Since complexity of subproblems is significantly reduced upon decomposition, obtaining subproblem solutions is much easier as compared to

[1] The linearization of absolute-value functions is performed in a standard way. Consider a simple problem:

$$\min_{x,y}\left\{x+\left|y-a^y\right|\right\}.$$

This problem is linearized by introducing a continuous decision variable q^y, and two constraints. The linearized problem can be equivalently written as:

$$\min_{x,y,q^y}\left\{x+q^y\right\}, s.t.-q^y\le y-a^y\le q^y.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/LRA.2018.2850056, IEEE Robotics and Automation Letters

that of the original problem (1), (2), (4)-(8). After one or several subproblems are solved, multipliers are updated as in (13), where stepsizes are defined in (15) and (16) and surrogate subgradient directions are:

$$\tilde{g}(\delta^k) = \sum_{\forall (i,j) \in O_m} \delta_{ijt}^k + z_{ij}^k - M_{ij}. \quad (21)$$

Job-shop scheduling are dynamic and schedules have to be executed in a stochastic environment. With our decomposition and coordination approach, the schedule can be regenerated fast with multipliers from the previous scheduling results.

VI. Numerical Results

The above tightening approach is implemented by using software Porta [23], and the decomposition and coordination method is implemented by using CPLEX 12.7.1.0 [24]. Two examples are tested on a laptop with the processor Intel® Xeon® CPU E3-1535M v6 @ 3.1-GHz and 32.00 GB of RAM. The first example demonstrates the effectiveness of formulation tightening. The second shows the computational efficiency and scalability of SAVLR with branch-and-cut.

Example 1: Medium-size problems

This example is to demonstrate the effectiveness of the formulation tightening. Two medium-size problem instances are considered. For the first one, data is taken from Pratt & Whitney's Development Operation shop [3] and the first 89 parts are considered. There are 19 machine types characterized by parts/operations they can process, and each type consists of 1 to 6 machines. For simplicity, it is assumed that all machines are available during all 200 time slots and the tardiness weights are 1 for all parts. With and without tightened constraints (9) and (10), the problem is solved by using branch-and-cut (B&C) with stopping criteria as 120 seconds (s) and 1% MIP gap. Results are shown in Table I. CPU time includes data and model loading, solving and solution outputting time.

TABLE I Comparison of Different Formulations: 19 Machine Types and 89 Parts

Formulation	Total weighted tardiness	MIP gap (%)	CPU time (s)	Solving time (s)	Branching time (s)
(1): Original	1793	1.03	80	72	58
(2): (1) + (9)	1793	1.06	67	60	34
(3): (2) + (10)	1792	1.01	40	34	0.5

Results show that CPU time is reduced by tightening and the solution quality is still high. In addition, branching time is dramatically reduced by adding both tightened constraints.

The second instance with 20 machines and 100 parts is taken from the standard OR-library [25], and the total number of time slots is 300. With different linearization methods and tightening constraints, the problem is solved by using B&C. The stopping criteria are 1200 s and 1% MIP gap. Testing results are shown in Table II.

TABLE II Comparison of Different Formulations: 20 Machines and 100 Parts

Formulation	Total weighted tardiness	MIP gap (%)	CPU time (s)	Solving time (s)	Branching time (s)
(1) If-then in CPLEX	/	/	1446	1200	/
(2) Standard Big-M	1504	2.86	1213	1201	1019
(3): Our original	1471	0.85	609	602	469
(4): (3) + (9)	1466	0.51	162	153	63
(5): (4) + (10)	1463	0.31	108	99	28

Results show that the CPU and branching time is much reduced by our linearization and tightening. The results are also compared with other recent results in the literature. In [26], it takes roughly 10,000 s to solve the 20-mahcine and 100-job problem on a workstation equipped with an Intel Core i5-4570 CPU @3.2GHz and 16 GB RAM in. Our results are obtained on a workstation equipped an Intel Xeon CPU E3-1535M v6 @ 3.1-GHz and 32 GB RAM. According to [27], the multi-core integer speed of our CPU is 40% faster than the one in [26], therefore, it would take roughly 7143 s to solve the problem by using the approach in [26] with our CPU. Although the testing data may not be exactly the same, it can still demonstrate computational efficiency of our approach. The above results on two instances demonstrate the great potential of formulation tightening for complicated MILP problems.

Example 2: Large-size problems

This example is to show computational efficiency and scalability of SAVLR +with B&C. The first problem instance is taken from [3] with 127 parts, and the total number of time slots is 300. Other settings are the same as in Example 1. With and without tightening, the problem is solved by B&C with stopping criteria as3600 s and 1 % MIP gap. With the tightened formulation, the problem is also solved by SAVLR with B&C, and the stopping criterion is 2 for the norm of constraint violations. The CPU time includes data and model loading, subproblem solving, surrogate subgradient and multiplier updating, feasible solution searching and solution outputting time, and solving time excludes loading and outputting time. Results are shown in Table III and Fig. 3 below.

TABLE III Comparison of Different Formulations and Method: 19 Machine Types and 127 Parts

Formulation	Original	Tightened	
Approach	B&C	B&C	SAVLR + B&C
Total weighted tardiness	2059	1962	1961
Lower bound	1855.4	1843.2	1958.2
Gap (%)	9.89 (MIP)	6.05 (MIP)	0.14 (Duality)
CPU time (s)	3699.1	3690.4	1260.4
Solving time (s)	3612.2	3600.2	636.9
Cutting time (s)	52.4	88.9	/
Branching time(s)	3545.8	3522.7	160.8

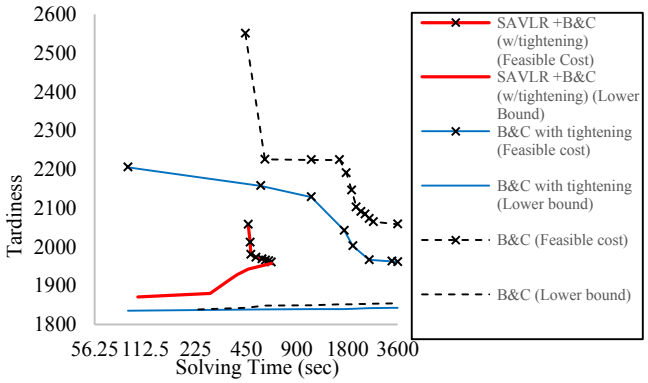

Figure. 3. Comparison of different formulations and methods: 19 machine types and 127 parts

As seen from the results, with our formulation tightening, B&C obtains a feasible solution with a gap of 6% in 3699 s, with 3546 s on branching. Within the same CPU time, the gap is 9% without formulation tightening. SAVLR with B&C

obtains a feasible solution with a gap of 0.14% in 1260 s, with 161 s on branching when finding feasible solutions.

To test scalability, a problem instance with 20 machines and 200 parts is taken from the OR-library [25], with 400 time slots. Results by SAVLR with B&C are shown in Table IV below. With formulation tightening, a feasible solution with a gap of 10.79% is obtained after 1500 s with solving time of 600 s. With systematic quality improvement, the method obtains another solution with a gap of 2.81% in 3000 s with solving time of 1200 s. Comparison with pure B&C is not included since it cannot obtain any feasible solution in 2 hours.

TABLE IV PERFORMANCE OF SAVLR FOR THE PROBLEM WITH 20 MACHINES AND 200 PARTS

CPU time (s)	1500	3000
Solving Time (s)	600	1200
Feasible cost	3642	3343
Duality gap (%)	10.79	2.81

All the results are obtained with initial multipliers as 0, and re-optimization with latest multipliers should be much faster. They demonstrate great potential of our formulation tightening and decomposition and coordination approach for complicated MILP problems.

VII. CONCLUSION

This paper is a pioneering effort toward obtaining near-optimal solutions with quantifiable quality fast for job-shop scheduling by: (1) reformulating the problems in an MILP form to make effective use of popular MILP methods such as branch-and-cut; (2) establishing a decomposition and coordination framework based on the problem reformulated in (1) with exponential reduction of complexity and accelerated convergence; (3) and developing a novel systematic approach to tighten subproblem MILP formulations in (2) for the first time. Testing results demonstrate that formulation tightening leads to significant computational improvement, and decomposition and coordination is efficient. For dynamic job-shop scheduling, the schedule can be regenerated fast based on the previous scheduling results. In the future work, more features such as energy efficiency will be considered with subsequent tightening. Moreover, motivated by Industry 4.0 and smart manufacturing, a distributed and asynchronous implementation of the approach will be investigated. We believe that this paper opens up new directions for more exploration to efficiently solve MILP problems.

REFERENCES

[1] I. G. Drobouchevitch, and V. Strusevich, "Heuristics for the two-stage job shop scheduling problem with a bottleneck machine," *European Journal of Operational Research*, Vol. 123, No. 2, pp. 229-240, 2000.

[2] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, Vol. 1, No. 2, pp. 117-129, 1976.

[3] D. J. Hoitomt, P. B. Luh, 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.

[4] C. A. Kaskavelis, and M. C. Caramanis, "Efficient Lagrangian relaxation algorithms for industry size job-shop scheduling problems," *IIE transactions*, Vol. 30, No. 11, pp. 1085-1097, 1998.

[5] H. Chen, C. Chu, J. M. and Proth, "An improvement of the Lagrangean relaxation approach for job shop scheduling: a dynamic programming method," *IEEE Transactions on Robotics and Automation*, Vol. 14, No. 5, pp. 786-795, 1998.

[6] 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.

[7] 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.

[8] 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.

[9] J. Hurink, B. Jurisch, and M. Thole, "Tabu search for the job-shop scheduling problem with multi-purpose machines," *Operations-Research-Spektrum*, Vol. 15, No. 4, pp.205-215, 1994.

[10] S. Dauzère-Pérès, and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," *Annals of Operations Research*, vol. 70, No.0, pp.281-306, 1997.

[11] M. Saidi-Mehrabad and P. Fattahi, "Flexible job shop scheduling with tabu search algorithms," *International Journal of Advanced Manufacturing Technology*, Vol. 32, No. 5-6, pp. 563-570, 2007.

[12] J. Kuhpfahl, and C. and Bierwirth, "A study on local search neighborhoods for the job shop scheduling problem with total weighted tardiness objective," *Computers & Operations Research*, Vol. 66, pp.44-57, 2016.

[13] X. N. Shen, and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Information Sciences*, Vol. 298, pp.198-224, 2015.

[14] S. Nguyen, M. Zhang, M. Johnston, and K. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Transactions on Evolutionary Computation*, Vol. 18, No.2, pp. 193-208, 2013.

[15] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing*, Vol. 29, No.3, pp. 603-615, 2018.

[16] G. Zhang, X. Shao, P. Li, and L. Gao, "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem," *Computers & Industrial Engineering*, Vol. 56, No.4, pp.1309-1318, 2009.

[17] M. A. Bragin, P. B. Luh, J. H. Yan, N. Yu, and G. A. Stern, "Convergence of the surrogate Lagrangian relaxation method," *Journal of Optimization Theory and Applications*, Vol. 164, No. 1, pp. 173-201, 2015.

[18] X. Sun, P. B. Luh, M. A. Bragin, Y. Chen, J. Wan, and F. Wang, "A decomposition and coordination approach for large-scale security constrained unit commitment problems with combined cycle units," *IEEE Transactions on Power Systems*, published online March 2018, DOI 10.1109/TPWRS.2018.2808272.

[19] M. A. Bragin, P. B. Luh, B. Yan, and X. Sun, "A Scalable Solution Methodology for Mixed-Integer Linear Programming Problems Arising in Automation," *IEEE Transactions on Automation Science and Engineering*, published online June 2018, DOI: 10.1109/TASE.2018.2835298.

[20] G. Belov, P. J. Stuckey, G. Tack, and M. Wallace, M., 2016, "Improved linearization of constraint programming models," in *Proceeding of* International Conference on Principles and Practice of Constraint Programming, pp. 49-65, Springer, Cham, 2016.

[21] 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.

[22] 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*.

[23] Heidelberg University, http://www.iwr.uni-heidelberg.de/groups/comopt/software/PORTA/

[24] IBM ILOG CPLEX V 12.1 User's Manual.

[25] Beasley'sOR-Library, http://mistic.heig-vd.ch/taillard/problemes.dir/problemes.html

[26] R. Braune, and G. Zäpfel, "Shifting bottleneck scheduling for total weighted tardiness minimization-A computational evaluation of subproblem and re-optimization heuristics," *Computers & Operations Research*, Vol. 66, pp. 130-140, 2016.

[27] http://cpu.userbenchmark.com/Compare/Intel-Core-i5-4570-vs-Intel-Xeon-E3-1505M-v6/2770vsm233404