Ordinal-Optimization Concept Enabled Decomposition and Coordination of Mixed-Integer Linear Programming Problems

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

***Abstract*—Many important optimization problems, such as manufacturing scheduling and power system unit commitment, are formulated as Mixed-Integer Linear Programming (MILP) problems. Such problems are generally difficult to solve because of their combinatorial nature, and may subject to strict computation time limitations. Recently, our decomposition-and-coordination method "Surrogate Absolute Value Lagrangian Relaxation" (SAVLR) exploits the exponential reduction of complexity upon problem decomposition and effectively coordinates subproblem solutions. In the method, subproblems are generally solved by using Branch-and-Cut (B&C). When subproblems are complicated, however, the approach might not be able to generate high-quality solutions within time limitations. In this paper, motivated by the "Ordinal Optimization" concept, this difficulty is resolved through exploiting a specific property of SAVLR that subproblem solutions only need to be "good enough" to satisfy a convergence condition. Time consuming B&C is eliminated in many iterations through obtaining "good enough" subproblem solutions based on "crude models" (e.g., LP-relaxed problems) or from heuristics. Testing results on generalized assignment problems demonstrate that the approach obtains high-quality solutions in a computationally efficient manner and significantly outperforms other approaches. This approach also opens up a new way to solve practical MILP problems that are subject to strict computation time limitations.**

***Index Terms*—Planning, Scheduling and Coordination, Manufacturing, Surrogate absolute-value Lagrangian relaxation (SAVLR), Ordinal optimization (OO), Generalized assignment problems (GAPs), Mixed-integer linear programming (MILP)**

Manuscript received February 15, 2020; Revised May 16, 2020; Accepted June 9, 2020. This paper was recommended for publication by Editor Jingang Yi upon evaluation of the Associate Editor and Reviewers' comments. This work is supported by the National Key Research and Development Project of China, No. 2017YFC0704100.

A.-B. Liu is with the Center for Intelligent and Networked Systems (CFINS), Department of Automation, Tsinghua University, Beijing 100084, China (e-mail: liuab19@mails.tsinghua.edu.cn).

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

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

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

I. Introduction

Many optimization problems, such as manufacturing scheduling [1, 2], generalized assignment [3-5] and power system unit commitment [6-8], are formulated as Mixed-Integer Linear Programming (MILP) problems. The quality of their solutions significantly affects system performance. Moreover, many problems need to be solved within limited time (e.g., 10, or 20 min). Good MILP methods that can obtain near-optimal solutions within time limits are therefore very important. That, however, is generally difficult to achieve since problem complexity increases dramatically as the problem size increases because of the presence of discrete decision variables.

To efficiently obtain near-optimal solutions for practical MILP problems, our recently decomposition-and-coordination method Surrogate Absolute Value Lagrangian Relaxation (SAVLR) exploits the exponential reduction of complexity upon decomposition, and effectively coordinates subproblem solutions [9]. In the method, through relaxing system-wide coupling constraints, the original problem is decomposed into smaller subproblems, each with much reduced complexity. To coordinate subproblem solutions, multipliers are iteratively updated by using surrogate subgradient directions, which are obtained by solving one or a few subproblems subject to the algorithm converging "surrogate optimality condition." To exploit the linearity, Branch-and-Cut (B&C) is generally used to solve subproblems. When subproblems are complicated, however, the approach might not be able to generate high-quality solutions in time limits.

To overcome the difficulties mentioned above, a novel solution methodology based on SAVLR is developed in this paper, motivated by the Ordinal Optimization (OO) concepts [10-13]. This method exploits a specific property of SAVLR that subproblem solutions only need to be "good enough" to satisfy the surrogate optimality condition. Time consuming B&C can thus be eliminated in many iterations through obtaining "good enough" subproblem solutions based on "crude models" (e.g., LP-relaxed problems) or from heuristics. Computational efforts required can thus be drastically reduced. Nevertheless, since the convergence condition is satisfied at each iteration, subproblem solutions are still effectively coordinated, leading to high-quality overall solutions. The above is presented in Section III, after reviewing major MILP

methods and the Ordinal Optimization (OO) concepts in Section II.

To demonstrate the performance of our method, the standard generalized assignment problems and generalized assignment problems with machine availability and job release times are tested in Section IV. Results are also compared with those obtained by using SAVLR with B&C (and indirectly with those obtained by using B&C, standard Lagrangian Relaxation, and Surrogate Lagrangian Relaxation). Testing results demonstrate that "good enough" subproblem solutions are efficiently obtained in many iterations based on our crude model and time-consuming B&C is only used as needed. Moreover, our method obtains high-quality overall solutions in a computationally efficient manner and significantly outperforms other approaches.

A natural follow-up question to the above is whether B&C can be completely eliminated in solving subproblems. This is explored in Section V by examining the examples considered in Section IV. Testing results show that for the standard generalized assignment problems, the crude models (the LP-relaxed problems) can approximate the subproblems well, and overall solutions of reasonable quality are obtained within very short CPU times without using B&C to solve subproblems. This, however, is not the case for the generalized assignment problems with machine availability and job release time, because good enough subproblem solutions are difficult to obtain based on the LP-relaxed problems. Further efforts are needed to develop good crude models, and a promising direction for that is formulation tightening [1, 8, 14].

II. Literature Review

Branch-and-Cut is reviewed in Section II-A. Decomposition and coordination methods, such as Lagrangian Relaxation (LR) and Surrogate Absolute Value Lagrangian Relaxation (SAVLR), are reviewed in Section II-B. Ordinal Optimization (OO) is reviewed in Section II-C.

A. Branch-and-Cut (B&C)

Branch-and-Cut [15-17] is a method to solve MILP problems by exploiting linearity. The key idea is to find the convex hull of the problem. If the convex hull is found, then the optimal solution can be obtained at one of its vertices by solving a linear programming problem. To obtain the convex hull, the integrality requirement is first relaxed. The method then cuts off regions outside the convex hull by adding "valid cuts." The fundamental difficulty is that finding convex hulls itself is NP hard, and effective valid cuts may be difficult to obtain. If the convex hull cannot be effectively obtained, the method then relies on time-consuming Branch-and-Bound and heuristics.

B. Decomposition and coordination methods

Traditionally, Lagrangian Relaxation (LR) is a method for mixed-integer separable optimization problems by exploiting separability [6, 7, 18, 19]. In the method, a problem is decomposed into smaller subproblems by relaxing system-wide coupling constraints. The complexity of a subproblem is exponentially reduced as compared with that of the original problem, and subproblem solutions are coordinated by using multipliers which are updated based on subgradients. At convergence of multipliers, feasible solutions are obtained by using heuristics. Because subgradients are obtained after solving all subproblems, the computational requirements for one iteration are high and multipliers may suffer from severe zigzagging. Moreover, convergence requires the knowledge of the optimal dual value, which is generally not available. While adaptive estimates of the optimal dual value are used in practice, such adaptive adjustments are computationally expensive. As a result, the overall performance of the method is poor.

Surrogate Lagrangian Relaxation (SLR) overcomes all major difficulties of standard LR presented above [20]. Within the method, multipliers are updated by using surrogate subgradient directions, which can be obtained by solving the relaxed problem approximately (or solving one or multiple subproblems) subject to the algorithm converging surrogate optimality condition. The computational efforts required to update multipliers, therefore, are significantly reduced, and zigzagging behavior of multipliers is also alleviated. Moreover, convergence is proven without requiring the knowledge of the optimal dual value. The difficulty of SLR is that the levels of constraint violations may not be reduced fast enough, resulting in slow convergence.

To accelerate convergence, SAVLR is developed by penalizing constraint violations using "absolute-value" penalty terms, which are exactly linearized [9]. Moreover, a novel adjustment of penalty coefficients is developed. SAVLR is generally combined with B&C to exploit both separability and linearity for MILP problems.

C. Ordinal Optimization (OO)

Ordinal Optimization is an efficient approach to solve complicated simulation-based optimization problems [10-13]. The approach rests on two basic ideas. The first is that softening the goal makes hard problem easier. If the goal of obtaining an optimal solution is softened to obtaining a "good enough" solution (e.g., top 5%), the complexity of the problem will be significantly reduced [13]. Therefore, instead of finding the best solution for sure, the goal of OO is to find a good enough solution with a high probability. The second is that order is much easier to determine than value. Instead of using accurate performances which generally require long simulation times to obtain, OO uses noisy performances obtained from crude models or quick simulation runs to roughly evaluate the relative orders of solutions, and then select good enough solutions. Different from simulation-based optimization problems, the objective function of an MILP problem can be easily calculated, and "good enough" solutions (e.g., top 5%) might not be good enough in terms of performance. Therefore, OO generally has not been used to solve MILP problems.

III. Methodology

In subsection III-A, a generic MILP problem is formulated, and the key steps of Surrogate Absolute Value Lagrangian Relaxation (SAVLR) [9] plus Branch-and-Cut (B&C) is summarized. In subsection III-B, the method is conceptually improved by using a crude model to efficiently obtain good enough subproblem solutions based on the Ordinal Optimization concept [10-13] while satisfying the algorithm

convergence condition. In subsection III-C, implementation of the algorithm is presented.

A. Generic formulation of MILP problems and Surrogate Absolute-Value Lagrangian Relaxation plus Branch-and-Cut

As reviewed in the Introduction, many practical problems are formulated as MILP problems. These problems usually consist of individual subsystems coupled through system-wide constraints. The objective function is additive in terms of individual subsystems, and can be written in the following way:

$$\min_{x,y} \left\{ \left(d^x\right)^T x + \left(d^y\right)^T y \right\}, (x,y) \in \Omega \subset \mathrm{R}^{n_x} \times \mathrm{Z}^{n_y}, \quad (1)$$

where x is an $n^x \times 1$ real decision vector, y is an $n^y \times 1$ integer decision vector. Related cost vectors d^x and d^y have appropriate dimensions. Constraints of individual subsystems can be written as:

$$A_j^x x_j + A_j^y y_j \le b_j,\ j = 1,...,J, \quad (2)$$

where x_j and y_j are disjoint components of x and y and have dimensions $n_j^x \times 1$ and $n_j^y \times 1$, respectively, such that $\sum_j n_j^x = n^x$ and $\sum_j n_j^y = n^y$. Related matrices A_j^x, A_j^y and vector b_j have appropriate dimensions. Subsystems are coupled through system-wide constraints

$$A^{x,0} x + A^{y,0} y = b^0, \quad (3)$$

and related matrices $A^{x,0}$, $A^{y,0}$ and vector b^0 have appropriate dimensions.

Recently, SAVLR+B&C shows great abilities to solve problem formulated by (1)-(3) through decomposition and coordination. To decompose the problem, the system-wide coupling constraints (3) is relaxed resulting in the following "Relaxed Problem":

$$\min_{x,y} L_{c^k}\left(x, y, \lambda^k\right), \text{ s.t. } (2),\ (x,y) \in \Omega, \quad (4)$$

where

$$L_{c^k}\left(x, y, \lambda^k\right) = \left(d^x\right)^T x + \left(d^y\right)^T y + \left(\lambda^k\right)^T g(x,y) + c^k \left\| g(x,y) \right\|_1 \quad (5)$$

is the "absolute-value" Lagrangian function, in which

$$g(x,y) = A^{x,0} x + A^{y,0} y - b^0 \quad (6)$$

includes a vector of constraint violations and is penalized by using L^1-norms in (5). Subproblems are then formed based on (5) by optimizing decision variables associated with one (or multiple) subsystems while fixing variables associated with other subsystems at values obtained at previous iterations:

$$\min_{x_j, y_j} L_{c^k}\left(x_j, x_{-j}^{k-1}, y_j, y_{-j}^{k-1}, \lambda^k\right), \text{ s.t. } (2),\ (x_j, y_j) \in \Omega_j, \quad (7)$$

where x_{-j} and y_{-j} are components of x and y without x_j and y_j, respectively. Solving subproblems (7) are much easier than solving the original problem (1)-(3) because of their reduced dimensionalities. These subproblems are generally solved by using B&C to exploit linearity after penalty terms in (7) are linearized in a standard way.

After solving a subproblem at iteration k, if the solution (x_j^k, y_j^k) satisfies the surrogate optimality condition

$$L_{c^k}\left(x_j^k, x_{-j}^{k-1}, y_j^k, y_{-j}^{k-1}, \lambda^k\right) < L_{c^k}\left(x_j^{k-1}, x_{-j}^{k-1}, y_j^{k-1}, y_{-j}^{k-1}, \lambda^k\right), \quad (8)$$

then multipliers are updated as

$$\lambda^{k+1} = \lambda^k + s^k g(x^k, y^k), \quad (9)$$

where

$$g\left(x^k, y^k\right) = A^{x,0} x^k + A^{y,0} y^k - b^0 \quad (10)$$

is a surrogate subgradient, with

$$x^k = (x_1^{k-1}, ..., x_j^k, ... x_J^{k-1}),\ y^k = (y_1^{k-1}, ..., y_j^k, ... y_J^{k-1}). \quad (11)$$

If the subproblem solution obtained does not satisfy (8), the method moves to solve the next subproblem. Since subproblem do not need to be fully solved, B&C is stopped when a subproblem solution that satisfies (8) is found to save CPU time. To guarantee convergence without the knowledge of the optimal dual value, the stepsize in (9) is set as

$$s^k = \alpha_k \frac{s^{k-1} \left\| g(x^{k-1}, y^{k-1}) \right\|_2}{\left\| g(x^k, y^k) \right\|_2},\ 0 < \alpha_k < 1, \quad (12)$$

with

$$\alpha_k = 1 - \frac{1}{Mk^\rho},\ \rho = 1 - \frac{1}{k^r},\ M \ge 1,\ 0 < r < 1. \quad (13)$$

After updating multipliers, the penalty coefficient is increased as:

$$c^{k+1} = \beta \cdot c^k,\ \beta > 1. \quad (14)$$

When the penalty coefficient becomes too large, feasibility is overemphasized and surrogate optimality condition (8) might not be satisfied even after solving all subproblems. In this case, the penalty coefficient is decreased as:

$$c^{k+1} = c^k / \beta,\ \beta > 1. \quad (15)$$

B. A crude method for good enough subproblem solutions

At each iteration of SAVLR, it is not necessary to fully solve a subproblem, and a "good enough" solution subject to the surrogate optimality condition (8) is sufficient. As reviewed in subsection 2.3, finding good enough solutions is much easier than finding the optimal one. Accurate optimization methods, e.g., B&C, therefore, might not be needed, and computationally efficient methods can be developed to generate good enough subproblem solutions.

Inspired by the ordinal comparison concepts, an efficient method to obtain good enough subproblem solutions is developed below based on the idea of "approximate solutions." LP-relaxed problems, which are formed by relaxing integer requirements in subproblems (7), are used as "crude models" to approximate subproblems, and the "approximate solution" (x', y') is the solution obtained by using a linear programming method. If all components of y' are integers, then (x', y') is the optimal solution to the subproblem. If not, the non-integer components are modified by rounding up or down to generate a set of solutions feasible to (7), and the set is denoted as S. To illustrate this step, suppose that all n_y components of y' are non-integers:

$$y' = (y_1, y_2, ..., y_{n_y}). \quad (16)$$

Through rounding up or down, multiple integer variables are obtained as:

$$\begin{aligned}
&\tilde{y}_0 = (\lfloor y_1 \rfloor, \lfloor y_2 \rfloor, ..., \lfloor y_{n_y} \rfloor),\\
&\tilde{y}_1 = (\lceil y_1 \rceil, \lfloor y_2 \rfloor, ..., \lfloor y_{n_y} \rfloor),\\
&\tilde{y}_2 = (\lfloor y_1 \rfloor, \lceil y_2 \rceil, ..., \lfloor y_{n_y} \rfloor),\\
&\vdots\\
&\tilde{y}_{2^n} = (\lceil y_1 \rceil, \lceil y_2 \rceil, ..., \lceil y_{n_y} \rceil).
\end{aligned} \tag{17}$$

The corresponding continuous variables are modified to satisfy (2). Set S is then formed by selecting from the above solutions feasible to subproblem (7). Since subproblem solutions in S are close to the approximate solution, they are expected to have high performances and contain good enough subproblem solutions with a high probability. A subproblem solution with the minimal cost can be obtained from S through calculating subproblem costs in (7). If the solution satisfies the surrogate optimality condition (8), it is accepted as a good enough solution. If not, B&C is used to solve the subproblems.

Because the crude models (LP-relaxed problems) are generally easy to solve, the computational efforts required are much less than those required by B&C. If such solutions are obtained in a significant number of iterations, then the speed-up can be drastic. Moreover, since the convergence condition is satisfied at each iteration, subproblem solutions are still effectively coordinated, leading to high-quality overall solutions.

C. Implementation of SAVLR+OO

In this subsection, algorithm initialization, searching for solutions feasible to the original problem (1)-(3), and stopping criteria are discussed. The steps of the algorithm are then presented.

1) Initialization, Finding Feasible Solutions and Stopping Criteria

Good multiplier initialization can reduce the number of iterations needed. The multipliers are generally initialized by using heuristics [18]. For daily operation optimization problems which need to be solved multiple times a day, multipliers can also be initialized with the values obtained from the previous optimization run. In this paper, for simplicity, multipliers are initialized at zero. For a given set of initial multipliers, subproblem solutions are initialized by solving the relaxed problem (4), and the stepsize is then initialized as ([20, equation (76)]:

$$s^0 = \frac{q - L_{c^0}\left(x^0, y^0, \lambda^0\right)}{\left\| g(x^0, y^0) \right\|_2^2}, \tag{18}$$

where q is the best estimation of cost for problem (1)-(3).

Since system-wide constraints (3) have been relaxed, solutions obtained from subproblems, when put together, generally do not form a solution feasible to the original problem (1)-(3). When the norm of constraint violations (6) reduce below a threshold, i.e.,

$$\left\| g(x^k, y^k) \right\|^2 \le \delta, \tag{19}$$

heuristics are used to find an overall feasible solution. This step is generally problem dependent. In this paper, feasible solutions are obtained through fixing variables of most subsystems at their latest available values, and optimizing remaining variables to satisfy the system-wide constraints (3) by using B&C.

When the norm of constraint violations (6) or the stepsize reduces to a small value, i.e.,

$$\left\| g(x^k, y^k) \right\|^2 \le \varepsilon \text{ or } s^k \le \gamma, \tag{20}$$

multipliers change little. At this time, the algorithm stops.

2) Steps of the Algorithm

The steps are as follows:

Step 0: Initialize λ^0, x^0 and s^0;

Step 1: For given λ^k and c^k, solve the LP-relaxed problem corresponding to subproblem (7) to obtain the approximate solution (x', y');

Step 2: Modify (x', y') through rounding non-integer components of y' to obtain multiple feasible solutions, denoted as set S;

Step 3: Compute the subproblem costs in (7) for solutions in S, and select the best one. If the surrogate optimality condition (8) is satisfied, go to Step 6;

Step 4: Solve the subproblem by using B&C. If the surrogate optimality condition (8) is satisfied, go to Step 6;

Step 5: If the surrogate optimality condition (8) is not satisfied after solving all the subproblems, reduce the penalty coefficient c^k per (15) and go to Step 1. If not, move to the next subproblem and go to Step 1;

Step 6: Update λ^k, s^k and c^k as (9)-(14);

Step 7: Check the criteria (19). If satisfied, search for solutions feasible to the original problem. Otherwise, go to step 1;

Step 8: Check the stopping criteria (20). If satisfied, stop. Otherwise go to step 1.

IV. Numerical Testing

The method was implemented by using MATLAB R2018a and CPLEX 12.8.0.0, and is tested on a computer with the Intel I5-8250U processor with four cores at 3.3-GHz, 8GB of RAM, and Windows 10. The dual simplex method of the CPLEX linear programming solver is used to solve the crude models (LP-relaxed problems) and the B&C solver is used to solve MILP subproblems. During testing, parallel computing of the CPLEX solver is on. To demonstrate the efficiency of SAVLR+OO, two examples are presented. In Example 1, the standard generalized assignment problems [3-5] are considered. In Example 2, job release times and machine availability are included following [9]. The codes implementing SAVLR+OO (as well as SAVLR+B&C) for both examples, and the corresponding data sets are available in the supplementary material of this paper.

Example 1. The Standard Generalized Assignment Problems

A standard generalized assignment problem is to assign a set of jobs to a set of machines. To capture assignments of jobs to machines, a set of binary variables is introduced:

$$\begin{aligned}
&y_{i,j} = \begin{cases} 1, & \text{if job } i \text{ is assigned to machine } j,\\ 0, & \text{otherwise,} \end{cases}\\
&i = 1, ..., I,\ j = 1, ..., J,
\end{aligned} \tag{21}$$

where I is the total number of jobs, and J is the total number of machines. One job can only be assigned to one machine, i.e.,

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.2020.3005125, IEEE Robotics and Automation Letters

$$\sum_{j=1}^{J} y_{i,j} = 1,\ i = 1,\ ...,\ I. \tag{22}$$

For machine j, its capacity T_j cannot be exceeded, i.e.,

$$\sum_{i=1}^{I} t_{i,j} y_{i,j} \le T_j,\ j = 1,\ ...,\ J, \tag{23}$$

where $t_{i,j}$ is the time required to process job i on machine j. The objective function is to minimize the total assignment cost, i.e.,

$$\min_{y_{i,j}} \sum_{j=1}^{J} \sum_{i=1}^{I} g_{i,j} y_{i,j}, \tag{24}$$

where $g_{i,j}$ is the cost for assigning job i to machine j.

After relaxing assignment constraints (22) and adding absolute-value penalty terms, a subproblem associated with machine j can be formulated as ([9, equation (25)]:

$$\begin{aligned} &\min_{y_{i,j},q_i} \sum_{i=1}^{I} g_{i,j} y_{i,j} + \lambda_i^k y_{i,j} + \frac{c^k}{2} q_i, \\ &s.t.\ (23),\ -q_i \le y_{i,j} + \sum_{s=1 \text{ to } J:\, s \ne j} y_{i,s}^{k-1} - 1 \le q_i\ \ i = 1,...,I. \end{aligned} \tag{25}$$

In the above, absolute-value penalty terms are linearized by introducing extra variables q_i. In actual implementation of this example, four machines are grouped to form one subproblem, and solving one subproblem is counted as one iteration. The multipliers are updated as in (9), where the stepsizes are set as in (12) and (13), and a surrogate subgradient is given by:

$$\tilde{g}\left(y^k\right) = \sum_{j=1}^{J} y_{i,j}^k - 1. \tag{26}$$

The penalty coefficient is updated as in (14) and (15). Three problem instances with 20, 40 and 80 machines, each with 2400 jobs, are considered. The iterative process stops when the norm of constraint violations reduces below 3 or stepsize $s^k < 0.01$.

For the problem instance with 20 machines and 2400 jobs, feasible costs and lower bounds against CPU times are plotted in Fig. 1. In the figure, CPU times include subproblem solving time, multiplier updating time and feasible solution searching time. The lower bound is obtained at convergence by evaluating the absolute-value Lagrangian function (5). The results show that near-optimal solutions are effectively obtained. A cost of 142,945 is obtained after 318s, and 142,894 after 539s. The gaps are 0.064% and 0.007%, respectively, showing that the feasible solutions obtained are good. There are three reasons for such efficiency. First, the complexity of the original problem is significantly reduced through decomposition. Second, subproblem solutions are effectively coordinated by using SAVLR. Third, good enough subproblem solutions are efficiently obtained based on the crude model. To demonstrate the last aspect, computational efforts required and the quality of subproblem solutions obtained by using B&C and the crude method per iteration are compared in Table I. It can be seen that CPU time consumed by solving LP-relaxed problems and modifying the approximate solutions is only 0.0442s on average, as compared to that of 0.2652s by using B&C. The "average performance difference" in the table is the average percentage difference in subproblem cost vs. the optimal subproblem cost, i.e.,

$$D^k = \frac{p^k - p^{k,*}}{p^{k,*}} \times 100\%,\ k = 1, 2, ...$$

with $p^{k,*}$ the optimal subproblem cost at iteration k and p^k the subproblem cost obtained by using either the crude model or by using B&C. It can be seen that the average performance difference of the solutions obtained based on crude model is 0.7139%. As counted, these solutions satisfy the surrogate optimality condition in 1,455 out of a total 3,033 iterations, leading to significant speed-up.

To better demonstrate the usage of B&C in our method, the number of iterations in which B&C is used is depicted in Fig. 2. In the figure, the horizontal axis is the number of iterations, and the vertical axis is the number of B&C used thus far. As can be seen, at the early stage of optimization, good enough subproblem solutions are obtained based on the crude model at almost all iterations, and B&C is rarely called. This, however, is not the case at the later stage of optimization. This is because the surrogate optimality condition is more difficult to be satisfied as the multipliers converge. Nevertheless, as shown in Fig. 1, even the first solution obtained is with a very small duality gap.

For comparison purpose, SAVLR+B&C is also tested. As shown in Fig. 1, near-optimal solutions are obtained slower than those obtained by using SAVLR+OO. A cost of 142,976 is

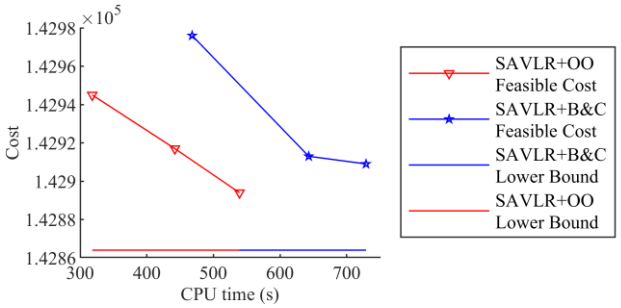

Fig. 1. Comparison of SAVLR+OO versus SAVLR+B&C for the standard GAPs with 20 machines 2400 jobs.

TABLE I
COMPARISON OF CRUDE METHOD AND BRANCH-AND-CUT IN EXAMPLE 1

Method	Instance	CPU time consumed per iteration (s)	Average performance difference
Branch-and-Cut	202400	0.2652	0.1100%
	402400	0.2589	0.3686%
	802400	0.2614	0.8676%
Crude method[1]	202400	0.0392	0.7139%
	402400	0.0473	1.4625%
	802400	0.0491	3.7655%

[1]: Solving LP-relaxed problem plus modifying the approximate solution

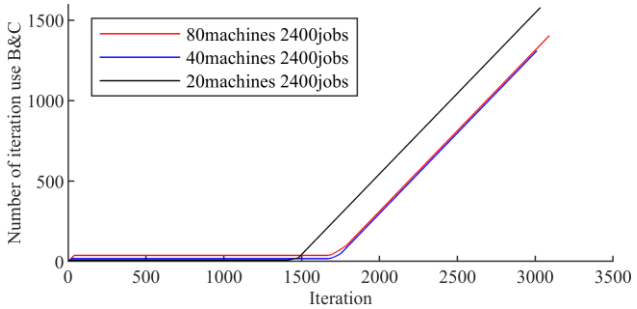

Fig. 2. The number of using B&C during the optimization process in Example 1.

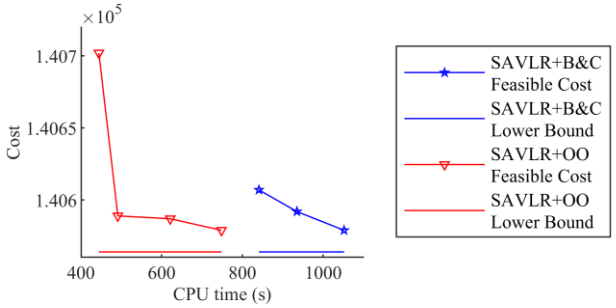

Fig. 3. Comparison of SAVLR+OO versus SAVLR+B&C for the standard GAPs with 40 machines 2400 jobs.

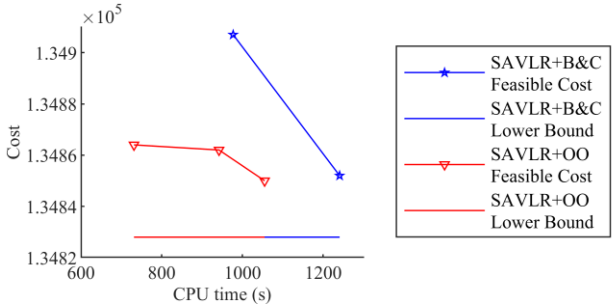

Fig. 4. Comparison of SAVLR+OO versus SAVLR+B&C for the standard GAPs with 80 machines 2400 jobs.

obtained after 468s, and 142,909 after 729s. Although B&C is stopped when a subproblem solution satisfying the surrogate optimality condition is found, as shown in Table I, it consumes 0.2652s on average while using the crude model only consumes 0.0392s on average. Moreover, SAVLR+B&C requires 2,964 iterations, and SAVLR+OO requires 3,033 iterations, showing that eliminating B&C does not significantly increase the number of iterations needed.

For the problem instance with 40 machines and 2400 jobs, SAVLR+OO obtains near-optimal solutions fast and significantly outperforms SAVLR+B&C. As shown in Fig. 3, a cost of 140,702 is obtained after 444s, and 140,579 after 748s.

For the problem instance with 80 machines and 2400 jobs, near-optimal overall solutions are also obtained in a computationally efficient manner. A cost of 134,864 with gap 0.0267% is obtained after 732s and good enough subproblem solutions are efficiently obtained base on the crude model in 1,687 out of a total of 3,091 iterations. This instance demonstrates the scalability of our method.

As reported in [9], SAVLR+B&C significantly outperforms other methods including B&C, Lagrangian Relaxation (LR), Surrogate Lagrangian Relaxation (SLR) and Alternate Direction Method of Multipliers (ADMM) because of decomposition and effective coordination. Since SAVLR+OO outperforms SAVLR+B&C, it also significantly outperforms B&C, LR, SLR, and ADMM.

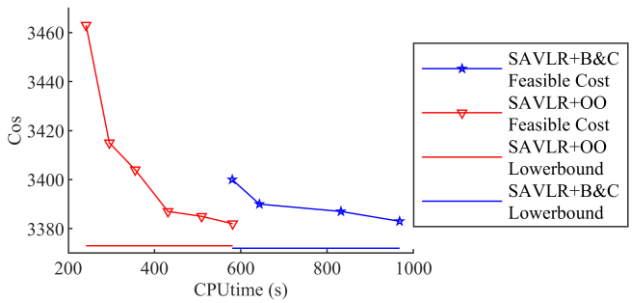

Fig. 5. Comparison of SAVLR+B&C and SAVLR+OO with 60 machines and 200 jobs and with a_j^{machine} and a_i^{job} generated using the uniform distribution U[0, 200].

Example 2. Generalized Assignment Problems with Machine Availability and Job Release Times. In this example, jobs release time and machine availability time are considered as in [9, p. 538]. Job i is not released until a certain time a_i^{job}, and machine j is only available after a certain time a_j^{machine} and until certain time T_j. To account for job release and machine availability, integer decision variables $b_{i,j}$ for all i and j are introduced representing the beginning time of job i on machine j subject to the following constraints:

$$b_{i,j} \ge a_i^{\text{job}},\ b_{i,j} \ge a_j^{\text{machine}},\ i=1,...,I,\ j=1,...,J. \tag{27}$$

Also, integer decision variables $c_{i,j}$ for all i and j are introduced representing the completion time of job i on machine j subject to the following constraints:

$$c_{i,j} \le T_j,\ i=1,...,I,\ j=1,...,J. \tag{28}$$

As presented in [9, equations (29) and (31)], if job i is assigned to machine j, i.e., $y_{i,j}$=1, its completion time should equal its beginning time plus processing time on machine j, i.e.,

$$y_{i,j}=1 \Rightarrow c_{i,j}-b_{i,j}=t_{i,j},\ i=1,...,I,\ j=1,...,J, \tag{29}$$

and time slots $[b_{i,j}, c_{i,j}]$ during which jobs are processed on machine j should not intersect for all j, i.e.,

$$c_{i,j} \le b_{i',j} \text{ OR } c_{i',j} \le b_{i,j},\ i \ne i',\ i',i=1,...,I,\ j=1,...,J. \tag{30}$$

The assignment constraints and the objective function are the same as in (22) and (24), respectively. After relaxing assignment constraints (22), a subproblem associated with machine j can be formulated as in [9]:

$$\begin{aligned}
&\min_{y_{i,j},c_{i,j},b_{i,j},q_i,z^1_{i,i',j},z^2_{i,i',j}} \sum_{i=1}^{I} g_{i,j}y_{i,j} + \lambda_i^k y_{i,j} + \frac{c^k}{2}q_i, \\
&\text{s.t. } (27),(28),\ -q_i \le y_{i,j} + \sum_{s=1 \text{ to } J:\, s\ne j} y_{i,s}^{k-1} - 1 \le q_i, \\
&\quad t_{i,j} - M\left(1-y_{i,j}\right) \le c_{i,j} - b_{i,j} \le t_{i,j} + M\left(1-y_{i,j}\right), \\
&\quad c_{i,j} \le b_{i',j} + Mz^1_{i,i',j}, c_{i',j} \le b_{i,j} + Mz^2_{i,i',j}, z^1_{i,i',j} + z^2_{i,i',j} = 1, \\
&\quad i,i' = 1,...,I, i \ne i'.
\end{aligned} \tag{31}$$

In the above, the logic constraints (29) and (30) are linearized as in [9, equations (30) and (32)]. Because the problem is complicated, one subproblem in this example only contains one machine. The updating of multipliers and the adjustments of the penalty coefficient are the same as in Example 1. The iterative process stops when the norm of constraint violations reduces below 3 or stepsize $s^k < 0.005$.

Three problem instances with 60 machines and 200 jobs are considered, and these instances are larger than those with 40 machines and 100 jobs in [9]. Data a_j^{machine} and a_i^{job} in three instances are individually generated from uniform distributions U[0, 200], U[0, 400], and U[0, 600], which have wider ranges as compared to U[0, 100] and U[0, 300] in [9].

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.2020.3005125, IEEE Robotics and Automation Letters

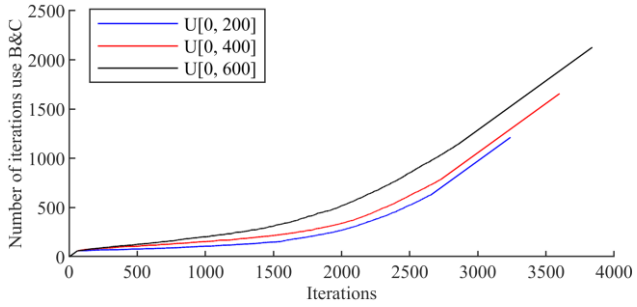

Fig. 6. The number of using B&C during the optimization process in Example 2.

TABLE II
COMPARISON OF CRUDE METHOD AND BRANCH-AND-CUT IN EXAMPLE 2

Method	Instance	CPU time consumed per iteration (s)	Average performance difference
Branch-and-Cut	U[0, 200]	0.3602	2.28%[1]
	U[0, 400]	0.4131	3.46%[1]
	U[0, 600]	0.4225	4.11%[1]
Crude method[3]	U[0, 200]	0.0060	24.66%[2]
	U[0, 400]	0.0068	26.37%[2]
	U[0, 600]	0.0063	27.62%[2]

[1]: Average MIP gap reported by CPLEX solver [2]: Average difference between costs obtained by the crude method and optimal costs [3]: Solving LP-relaxed problem plus modifications

Results with a_j^{machine} and a_i^{job} generated by using the uniform distribution U[0, 200] are presented first. As shown in Fig. 5, near-optimal overall solutions are efficiently obtained. A cost of 3,382 is obtained after 581s, and the lower bound obtained at convergence is 3,373, with a gap of 0.27%, showing that the solution obtained is good. As shown in Table II, CPU time required by crude model is only 0.006s, and the average performance difference is 24.66%, which is significantly larger than that of B&C. Nevertheless, as counted, the crude model is good enough in 2,027 out of a total of 3,240 iterations, and therefore the speed-up is drastic.

For comparison purpose, SAVLR+B&C is also tested. As shown in Table II, using B&C to solve a subproblem takes 0.3602s on average, which is much longer than using the crude model of 0.006s. As shown in Fig. 5, a cost of 3,400 is obtained after 580s and a cost of 3,383 is obtained after 968s. Therefore, near-optimal solutions are obtained much slower using SAVLR+R&C than those obtained by using SAVLR+OO. Moreover, SAVLR+B&C requires 3,060 iterations, and SAVLR+OO requires 3,240 iterations, showing that eliminating B&C does not significantly increase the number of iterations needed.

Results with a_j^{machine} and a_i^{job} generated by using the uniform distribution U[0, 400] are shown in Fig. 6. With SAVLR+OO, good enough subproblem solutions are efficiently obtained based on the crude model in 1,657 iterations, and a cost of 3,484 with gap 0.52% is efficiently obtained after 738s.

Lastly, results with a_j^{machine} and a_i^{job} generated by using the uniform distribution U[0, 600] are presented. Although a_j^{machine} and a_i^{job} have wide ranges, good enough subproblem solutions are still efficiently obtained based on the crude model and near-optimal solutions are obtained fast. As shown in Fig. 8, a

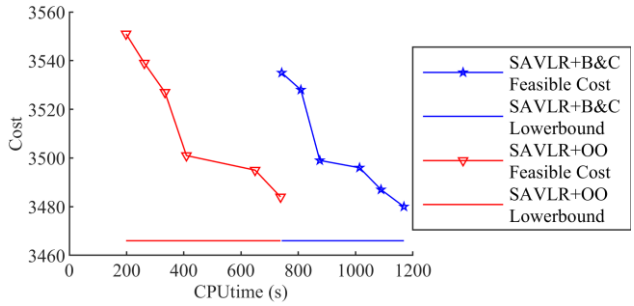

Fig. 7. Comparison of SAVLR+B&C and SAVLR+OO with 60 machines and 200 jobs and with a_j^{machine} and a_i^{job} generated using the uniform distribution U[0, 400].

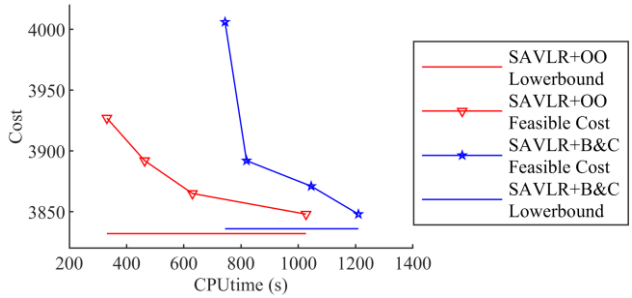

Fig. 8. Comparison of SAVLR+B&C and SAVLR+OO with 60 machines and 200 jobs and with a_j^{machine} and a_i^{job} generated using the uniform distribution U[0, 600].

cost of 3,927 with gap 2.42% is obtained after 331s, and a cost of 3,848 with gap 0.42% is obtained after 1,027s.

V. THE METHOD WITHOUT BRANCH-AND-CUT TO SOLVE SUBPROBLEMS

As mentioned in Section III, B&C is used to solve subproblems when the crude method cannot find a good enough solution. A natural follow-up question is whether B&C can be completely eliminated in solving subproblems. This possibility is explored by examining both Examples 1 and 2 with test settings the same as in Section IV. The results are shown in Table III.

For Example 1, SAVLR+OO without B&C to solve subproblems can obtain solutions with less than 5% gap within about 200s. This is because for the standard generalized assignment problems, the LP-relaxed problems approximate the subproblems well, and therefore the crude model can generate good enough subproblem solutions within almost all first 1500 iterations. This can also be seen from Fig. 2. Therefore, multipliers can approach the optimal multipliers,

TABLE III
RESULT OF SOLVING STANDARD GAPS BY USING SAVLR+OO WITHOUT B&C TO SOLVE SUBPROBLEMS

Instance	Feasible Cost	Lower Bound[1]	Gap	CPU time (s)
20M2400J	143,971	142,864	0.77%	74.5
40M2400J	142,747	140,564	1.55%	109
80M2400J	137,662	134,828	2.10%	201

[1]: Since the multipliers cannot fully converge in this section, the lower bound may not be obtained by calculating (5). The lower bound shown here is the value obtained from Section IV.

leading to a reasonable overall solution. For Example 2, however, the method cannot obtain an overall feasible solution without B&C. This is because good enough subproblem solutions are difficult to obtain based on LP-relaxed problems only. Therefore, multipliers cannot approach the optimal multipliers, and the norms of constraint violations cannot be reduced to a small value. As a result, an overall feasible solution cannot be obtained.

In practice, many formulations are themselves crude. Therefore, the gap requirement might not be strict, and 5% or 10% might still be acceptable especially under strict time limitations. Moreover, many practical problems may need to be solved with limited computing resources (in terms of both hardware and software) such that B&C may be difficult or too expensive to implement, e.g., problems to be solved by embedded processors. For such problems, SAVLR+OO without B&C to solve subproblems and using heuristics to obtain overall feasible solutions might be an efficient and viable approach. Further efforts are needed to develop good crude models, and a promising direction for that is formulation tightening [1, 8, 14].

VI. Conclusion

In this paper, a novel decomposition and coordination methodology for MILP problems is developed. Through obtaining good enough subproblem solutions by using a crude method, computational requirements of SAVLR+B&C is drastically reduced. The possibility of totally eliminating B&C in solving subproblems also opens up a way for fast resolution of MILP problems with strict time limitations.

Acknowledgment

The authors would like to thank Professor Yu-Chi Ho of Harvard University for his insightful and inspirational suggestion of combining Surrogate Absolute Value Lagrangian Relaxation with the Ordinal Optimization concept.

References

[1] B. Yan, M. A. Bragin, and P. B. Luh, "Novel Formulation and Resolution of Job-Shop Scheduling Problems," *IEEE Robotics and Automation Letters,* vol. 3, no. 4, pp. 3387-3393, 2018.

[2] A. Che, W. Lei, J. Feng, and C. Chu, "An improved mixed integer programming approach for multi-hoist cyclic scheduling problem," *IEEE Transactions on Automation Science and Engineering,* vol. 11, no. 1, pp. 302-309, 2013.

[3] G. T. Ross and R. M. Soland, "A branch and bound algorithm for the generalized assignment problem," *Mathematical programming,* vol. 8, no. 1, pp. 91-103, 1975.

[4] D. G. Cattrysse and L. N. Van Wassenhove, "A survey of algorithms for the generalized assignment problem," *European journal of operational research,* vol. 60, no. 3, pp. 260-272, 1992.

[5] M. L. Fisher, R. Jaikumar, and L. N. Van Wassenhove, "A multiplier adjustment method for the generalized assignment problem," *Management science,* vol. 32, no. 9, pp. 1095-1103, 1986.

[6] W. Ongsakul and N. Petcharaks, "Unit commitment by enhanced adaptive Lagrangian relaxation," *IEEE Transactions on Power Systems,* vol. 19, no. 1, pp. 620-628, 2004.

[7] Q. Zhai, X. Guan, and J. Cui, "Unit commitment with identical units successive subproblem solving method based on Lagrangian relaxation," *IEEE Transactions on Power Systems,* vol. 17, no. 4, pp. 1250-1257, 2002.

[8] B. Yan, P. Luh, T. Zheng, D. Schiro, M. Bragin, F. Zhao, J. Zhao, and I. Lelic, "A Systematic Formulation Tightening Approach for Unit Commitment Problems," *IEEE Transactions on Power Systems,* 2019.

[9] M. A. Bragin, P. B. Luh, B. Yan, and X. Sun, "A Scalable Solution Methodology for Mixed-Integer Linear Programming Problems Arising in Automation," *IEEE Transactions on Automation Science and Engineering,* vol. 16, no. 2, pp. 531-541, 2018.

[10] Y.-C. Ho, R. S. Sreenivas, and P. Vakili, "Ordinal optimization of DEDS," *Discrete event dynamic systems,* vol. 2, no. 1, pp. 61-88, 1992.

[11] Y.-C. Ho, Q.-C. Zhao, and Q.-S. Jia, *Ordinal optimization: Soft optimization for hard problems*. Springer Science & Business Media, 2008.

[12] T. E. Lau and Y.-C. Ho, "Universal alignment probabilities and subset selection for ordinal optimization," *Journal of Optimization Theory and Applications,* vol. 93, no. 3, pp. 455-489, 1997.

[13] L. H. Lee, T. W. E. Lau, and Y. C. Ho, "Explanation of goal softening in ordinal optimization," *IEEE Transactions on Automatic Control,* vol. 44, no. 1, pp. 94-99, 1999.

[14] 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 *2018 IEEE Power & Energy Society General Meeting (PESGM)*, 2018: IEEE, pp. 1-5.

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

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

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

[18] X. Guan, P. B. Luh, H. Yan, and J. Amalfi, "An optimization-based method for unit commitment," *International Journal of Electrical Power & Energy Systems,* vol. 14, no. 1, pp. 9-17, 1992.

[19] F. Zhuang and F. D. Galiana, "Towards a more rigorous and practical unit commitment by Lagrangian relaxation," *IEEE Transactions on Power Systems,* vol. 3, no. 2, pp. 763-773, 1988.

[20] 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.