An Effective Subgradient Method for Scheduling a Steelmaking-Continuous Casting Process

Kun Mao, Quan-Ke Pan, Tianyou Chai, *Fellow, IEEE*, and Peter B. Luh, *Fellow, IEEE*

***Abstract*—The steelmaking-continuous-casting (SCC) process, which includes steelmaking, refining and continuous casting, is one of the major bottlenecks of iron and steel production. Efficient and effective scheduling of this process is essential to improve the productivity and reduce the production costs of the entire production system. We present a time-index formulation for this scheduling problem and a Lagrangian relaxation (LR) approach based on the relaxation of the machine capacity constraints. The relaxed problem is solved using an efficient polynomial dynamic programming algorithm. The corresponding Lagrangian dual (LD) problem is solved using a deflected conditional subgradient level method. Unlike the conventional subgradient algorithms for the LD problem, our method guarantees convergence using the Brannlund's level control strategy to replace the strict convergence condition that the optimum of the dual problem is known *a priori.* Furthermore, our method enhances the efficiency by introducing a deflected conditional subgradient to weaken the zigzagging phenomena that slows the convergence of conventional subgradient algorithms. The computational results demonstrate that the approaches can quickly obtain high-quality solutions and are notably promising for the SCC scheduling.**

***Note to Practitioners*—Efficient and effective SCC schedule is vital for the manufacturing system of iron and steel production. Unfortunately, the scheduling is extremely difficult because of its combinatorial nature and practical complex constraints such as job grouping constraints, precedence constraints, different transport time, and setup times. To obtain high-quality solutions within an acceptable computational time, we can use a problem-oriented approach, which can be the LR. However, there are two deficiencies in this approach: its empirical termination criteria, such as maximal iteration number or running time, which make it difficult to find a golden rule for various problems, and the inefficiency, which is caused by the so-called zigzagging phenomena. To overcome these deficiencies, this paper develops an effective subgradient method for SCC scheduling based on the machine capacity relaxation. This method gives an objective termination criterion based on the convergence condition of the method, and improves the efficiency based on a new search direction or a new subgradient. Then, the work shows how this method can be applied to solve an SCC scheduling problem. The computational results confirm their effectiveness and efficiency. The approaches can also be applied to other similar production scheduling problems.**

***Index Terms*—Hybrid flowshop, Lagrangian relaxation, manufacturing system, scheduling, subgradient optimization.**

Manuscript received June 04, 2014; accepted June 19, 2014. Date of publication July 17, 2014; date of current version July 17, 2015. This paper was recommended for publication by Associate Editor C.-H. Chen and Editor L. Shi upon evaluation of the reviewers' comments. This work was supported in part by the National Science Foundation of China (61174187, 61333006, 61304107, 61104179 and 61104174), in part by the Program for New Century Excellent Talents in University (NCET-13-0106), in part by the Specialized Research Fund for the Doctoral Program of Higher Education (20130042110035), in part by the Science Foundation of Liaoning Province in China (2013020016), in part by the Basic Scientific Research Foundation of Northeast Universityunder Grant N110208001, in part by the Fundamental Research Funds for the Central Universities (N120708001 and N130508001), in part by the Starting Foundation of Northeast Universityunder Grant 29321006, in part by IAPI Fundamental Research Funds (2013ZCX02), in part by the Project 111 (B08015) and the State Key Laboratory of Synthetical Automation for Process Industries, and in part by the Postdoctoral Science Foundation of China (2014M552040). *(Corresponding author: Quan-Ke Pan.)*

K. Mao, Q.-K. Pan, and T. Chai are with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, 110819, China (e-mail: mao_kun@126.com; panquanke@mail.neu.edu.cn; tychai@mail.neu.edu.cn).

P. B. Luh, is with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT 06269-2157 USA (e-mail: peter.luh@uconn.edu).

Digital Object Identifier 10.1109/TASE.2014.2332511

I. Introduction

THE iron and steel industry plays an important role in modern global economy by providing raw materials for many other industries such as machinery production, automobiles, aircrafts, housing and food services, etc. The manufacturing process of the iron and steel production is a high-temperature and high-weight material flow with complicated technological process, including iron-making, steelmaking-continuous casting (SCC), and steel rolling. To head towards continuous, fast and automated process along a large infrastructure to attain different types of high-quality and low-cost products, modern iron and steel companies use the computer integrated manufacturing system (CIMS) to improve the productivity, reduce the production costs, and enable efficient material and energy utilization [1]. To satisfy the fluctuating demands for different types of products, various rolling mills in the steel-rolling phase have been designed with sufficient production capacity. Because SCC is a complicated technological multistage process, which requires expensive and energy-intensive equipments that continuously run, its capacity is always below the actual capacity of the steel-rolling phase. Thus, the SCC process is a bottleneck in the iron and steel production. The effective and efficient scheduling of the SCC process is a vital component in improving the productivity of the entire production system. An efficient and effective scheduling algorithm is also important for the CIMS system.

Optimal scheduling of the SCC process can produce obvious benefits such as profit growth, production cost saving, material and energy consumption reduction, and customer satisfaction improvement. However, the SCC scheduling is extremely complicated because of its combinatorial nature, complex practical constraints, strict requirements on material continuity and flow time, and technological requirements to ensure the practical feasibility of the resulting scheduling. It is difficult to obtain high-quality solutions within acceptable computational times. Although the experience-based manual scheduling system and informal coordination perform well in some situations, they are

restricted to simple manufacturing processes and small-scale problems.

Indeed, the SCC scheduling problem is composed of some nice mathematical structures that are coupled with certain complicated constraints that make any solution procedure difficult. The LR approach can decouple these complicated constraints such as machine capacity constraints, and exploit the special structures to solve the problem. Specifically, the LR approach relaxes the coupling constraints of the scheduling problem with Lagrangian multiplies. Then, the relaxed problem can be decomposed into tractable subproblems. The corresponding dual problem, which is called Lagrangian dual (LD) problem, can be solved using subgradient optimization methods, where a subgradient vector is obtained by optimizing the relaxed primal problem, and the multipliers are updated iteratively along the direction of this subgradient vector. At the end of each multipliers updating iteration, a heuristic is applied to obtain a feasible one from the relaxed problem solutions. At the termination of the subgradient optimization methods, the best feasible solution is chosen as the solution to the primal problem. The dual theory guarantees that the solution of the LD problem can provide the lower bound of the primal problem, whereas the best feasible solution offers an upper bound of the primal problem [2]. The LR approach has emerged as a practical approach for various scheduling problems such as single machine scheduling problem [3], parallel machine scheduling [4], flowshop scheduling [5], jobshop scheduling [6], [7], hybrid flowshop [8], [9], etc. Some researchers such as [10] and [11] applied the LR approach to the SCC scheduling problem under different production environments based on the machine capacity relaxation.

The successful application of the LR approach for many combinatorial optimization problems including scheduling problems, can be traced back to the pioneering works of Held and Karp on the traveling salesman problem [12]. However, a major challenge in the LR approach is to effectively and efficiently optimize the LD problem, which is nondifferentiable and piecewise affine [2]. The subgradient optimization method is a commonly used method for the LD problems because of its simplicity and low computational complexity. However, there are two major drawbacks to the pure subgradient method: the strict convergent condition that the optimum of the LD problem must be known in advance [2], and the slow convergence, which is mainly caused by the so-called zigzagging phenomena [13].

For the first drawback, most LR approaches for scheduling problems use an empirical termination criterion such as maximum iteration number or maximum running time. It is difficult to find a golden rule to determine a suitable termination criterion for various scale problems. Some researchers proposed alternative convergence conditions based on various selections of step-length methods [14]–[17]. However, these methods often come at the cost of sophistication, which hinders potential applications. For the second drawback, various search-direction methods have been developed to improve the convergence by weakening the zigzagging phenomena [13], [18]–[20]. However, these methods cannot fully handle the zigzagging problems. Unlike the above methods, [21] developed a surrogate gradient method where a subgradient direction can be obtained without optimizing all subproblems. Later, [22] extended this method to a surrogate subgradient framework and showed that it has similar properties to those of the subgradient algorithm. Therefore, like the subgradient method, the surrogate subgradient also faces the first drawback. Recently, [23] combined the constraint programming and extended subgradient information to improve the LR convergence for production scheduling.

Motivated by the above work, we develop an efficient subgradient method for the SCC scheduling problems based on the LR approach. To replace the strict convergent condition, the method dynamically adjusts the overestimation of the optimum of dual problem, which is indeed the Brannlund's level control strategy [15], [16]. Furthermore, it introduces a deflected conditional subgradient to improve the convergence. The details of the method and the proof are provided in Section IV and the Appendix, respectively.

The remainder of this paper is organized as follows. Section II describes the SCC process and briefly reviews the SCC scheduling methods. The relaxation method is described in Section III. Computational experiments and comparisons on various scale problems are provided in Section V.

II. Formulation of the SCC Scheduling Problem

A. The Production Process of Steelmaking-Continuous Casting (SCC)

SCC processes the hot metal to steel with a well-defined chemical composition and solidify the steel to slabs. This process consists of three stages: steelmaking, refining and continuous casting. Each stage has multiple parallel identical machines. We call the produced molten iron in the same furnace a charge or job. All jobs follow the same production flow from steelmaking to continuous casting. At each stage, a job cannot be simultaneously processed by more than one machine, and a machine can process at most one job at a time. Job processing also cannot be interrupted. The transportation times between any two machines may be different because of the different distances. A set of jobs must be contiguously processed on the same caster at the last stage because of technological constraints. A set of jobs is called a cast or batch in the scheduling system. There is setup time between two adjacent batches on the same caster. The operation sequence of the jobs in a batch is predefined, and the machines for each batch are known.

The SCC process can be viewed as a complex hybrid flow shop (HFS) with the following features: job grouping and precedence constraints, different transportation times and setup time constraints on the machines of the last stage. This paper aims to minimize the total weighted completion time and the total weighted job-waiting time. The minimization of the total weighted completion time helps reducing the production cost, the work-in-process inventory level, and the delivery tardiness of final products, whereas the minimization of the total weighted job-waiting time reduces the energy consumption and provides the continuity of the production process.

For the complexity, [24] showed that the problem $F2||\sum c_j$ (two-machine flow shop problem to minimize the sum of completion times of all jobs) is strongly NP-hard. Our scheduling problem is much more complex; we can conclude that the considered problem is also strongly NP-hard. As a complex

case of HFS, the SCC scheduling problems have been studied using various techniques considering different real-world production environments. Several recent comprehensive reviews on production planning and scheduling techniques for steel production can be found in [25]–[27]. The methods for SCC scheduling problems can be broadly divided into three categories: mathematical programming, heuristics, and artificial intelligence. For the mathematical programming methods, [1] combined linear programming and heuristic to address a realistic case of SCC scheduling. For the heuristics, [28] used a combinatorial auction-based approach to a three-stage HFS of SCC. Reference [29] developed a unit-specific event-based continuous-time mixed-integer linear optimization model for the SCC scheduling problem. For the artificial intelligence methods, [30] combined ant colony optimization and nonlinear optimization methods for a three-stage SCC process. Recently, [27] proposed an effective artificial bee colony algorithm for a real-world HFS problem in steelmaking process.

It is worth noting that the above literature assumed the transportation times among the machines are the same. However, in practice, the distances among the machines are different because of the shop layout, which always leads to different transportation times among the machines. The transportation times may affect the starting times of the downstream operations, particularly the continuous-casting operation in the last stage, and the entire schedule. Therefore, we consider different transportation times among the machines in this paper.

B. Mathematical Formulation of the SCC Scheduling Problem

This paper uses a time-indexed formulation [31] to model the scheduling problem. In this formulation, the planning horizon is discretized into the periods $1, 2, \ldots, T$, where period t starts at time $t-1$ and ends at time t.

1) Notation: This notation describes the indices, sets, number of elements, fixed parameters, and variables.

Sets and number of elements

M_j number of available machines at stage j;

Ω set of indices of all jobs, $|\Omega|$ is the number of all jobs;

Ω_n set of indices of the nth batch, $n = \{1, 2, \ldots, N\}$, where N is the total number of batches; $\Omega_{n_1} \cap \Omega_{n_2} = \emptyset$, for any $n_1, n_2 \in \{1, 2, \ldots, N\}$, $n_1 \neq n_2$;$\Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_n = \Omega$;

B_k set of indices of all batches on the kth machine at the last stage;

$s(n)$ index of the last job in the nth batch, $s(n) = s(n-1) + |\Omega_n|$, $s(0) = 0$, $s(N) = |\Omega|$;

$b(k)$ index of the last batch on the kth machine at the last stage, $b(k) = b(k-1) + |B_k|$, $b(0) = 0$, $b(M_S) = N$, $1 \leq k \leq M_S$, where S is the total number of stages. Then, $B_k = \{b(k-1)+1, \cdots, b(k)\}$;

R_r the rth route for a job, $r = m_{1,r} + \sum_{j=2}^{S-1} \bar{M}_j (m_{j,r} - 1)$, $R_r = (m_{1,r}, m_{2,r}, \ldots, m_{S-1,r})$, $1 \leq m_{j,r} \leq M_j$, $1 \leq j < S$, $1 \leq r \leq \prod_{j=1}^{S-1} M_j$, $\bar{M}_j = \prod_{s=1}^{j-1} M_s$. For instance, if job i is processed following route R_r, it will be processed on machine $m_{j,r}$ at stage j.

T: total number of time periods in the planning horizon.

Data or fixed parameters

P_{ij} processing time of job i at stage j;

T_{j,k_1}^{j+1,k_2} transportation time from the k_1th machine of stage j to the k_2th machine of stage $j+1$;

$T_{r,j}$ transportation time from stage j to stage $j+1$ in the rth route for a job, $T_{r,j} = T_{j,m_1}^{j+1,m_2}$, $m_1 = m_{j,r}$, $m_2 = m_{j+1,r}$;

Su setup time between two adjacent batches on the same machine at the last stage;

W_j penalty coefficient for the waiting time of each job between stage j and stage $j+1 (1 \leq j \leq S-1)$;

W_S penalty coefficient for the completion time of each job at stage S.

Decision variables

$x_{i,j,t}$ 0/1 variable, which is equal to one if and only if job i at stage j starts at time period t;

$y_{i,r}$ 0/1 variable, which is equal to one if and only if job i is processed following the rth route;

$c_{i,j}$: completion time of job i at stage j.

2) Mathematical Model: With the above symbols, the complex HFS scheduling problem is formulated as follows.

Objective function:

The objective function minimizes the total weighted completion time of each job and the job waiting between two adjacent operations, which is provided as follows:

$$(\mathrm{P}) \min G = \sum_{i=1}^{|\Omega|} \sum_{j=1}^{S-1} W_j (c_{i,j+1} - c_{i,j} - P_{i,j+1}) + \sum_{i=1}^{|\Omega|} W_S c_{i,S}. \quad (1)$$

Constraints:

- Using the time index variables, the completion time of job i at stage j can be written as

$$c_{i,j} = \sum_{t=1}^{T-P_{i,j}+1} (t - 1 + P_{i,j}) x_{i,j,t}, \quad i \in \Omega, \quad 1 \leq j \leq S. \quad (2)$$

- Each job can start at exactly one particular time

$$\sum_{t=1}^{T-P_{i,j}+1} x_{i,j,t} = 1, \quad i \in \Omega, \quad 1 \le j \le S. \tag{3}$$

- For two consecutive operations of the same job, an operation can only start when its preceding operation has been finished. In addition, the transportation times between two machines are different

$$c_{i,j+1} - P_{i,j+1} - c_{i,j} - \sum_{r=1}^{|R|} y_{i,r} T_{r,j} \ge 0, i \in \Omega, \quad 1 \le j < S. \tag{4}$$

- A job is processed following via one route:

$$\sum_{r=1}^{|R|} y_{i,r} = 1, i \in \Omega. \tag{5}$$

- The adjacent jobs in the same batch must be processed consequentially without waiting time at the last stage

$$c_{i+1,S} - P_{i+1,S} = c_{i,S}, \quad \forall i, i+1 \in \Omega_n, 1 \le n \le N. \tag{6}$$

- There is setup time between two adjacent batches to change equipment on the same machine in the last stage

$$c_{i+1,S} - c_{i,S} - P_{i+1,S} \ge Su \tag{7}$$

  where $i = s(b(k-1)+n)$, $1 \le k \le M_S$, $1 \le n \le |B_k| - 1$.
- The number of jobs that are simultaneously processed at time period t cannot be greater than the total number of available units. This constraint is the machine capacity constraint

$$\sum_{i \in \Omega} \sum_{\tau = \max\{t - P_{i,j}+1, 1\}}^{t} x_{i,j,\tau} \le M_j \tag{8}$$

  where $1 \le j < S, 1 \le t \le T$.
- The variables must hold for the following constraints:

$$c_{ij} \ge 0, i \in \Omega, 1 \le j \le S \tag{9}$$

$$x_{ijt} \in \{0,1\}, i \in \Omega, 1 \le j \le S, 1 \le t \le T \tag{10}$$

$$y_{i,r} \in \{0,1\}, i \in \Omega, 1 \le r \le |R|. \tag{11}$$

Remark 1: It is worth noting that our formulation does not introduce the big-M method to describe the machine capacity constraints or operation precedence constraints as in [8], [10], [11], and [27]. The big-M method to formulate this scheduling problem involves two big-M constraints: the operation precedence constraints because of different routes and the machine capacity constraints. In this case, it is not easy to handle the relaxed problem with the relaxation of machine capacity constraints because the big-M constraint remains. According to the computational results in [32], the linear programming relaxation of the time-index formulation usually produces tighter lower bound than the big-M formulations. For the same formulation, the LR always produces tighter lower bound than the linear programming relaxation [2], so the LR of the time-index formulation may produce tighter lower bound than the big-M formulation. Based on these observations and results, we prefer to adopt the time-index formulation.

III. Solution Methodology

A. Lagrangian Relaxation

There are two common relaxation strategies: machine capacity relaxation and operation precedence relaxation [33]. For the machine capacity relaxation in the above model, the machine capacity constraints (6)–(8), which couple different jobs on the same machine, can be relaxed to decompose the LR problem into job-level subproblems. For the operation precedence relaxation, the operation precedence constraints (4) can be relaxed to decompose the LR problem into $S-1$ complicated parallel machine scheduling problems. However, the parallel machine scheduling problem is NP-hard for $m \ge 2$ in general [34], which implies that the subproblem is also NP-hard. Hence, this paper adopts the machine capacity relaxation as follows.

By relaxing constraints (6)–(8) with Lagrangian multipliers $\{\mu_{1,i}\}$, nonnegative multipliers $\{\mu_{2,n}\}$, and nonnegative Lagrangian multipliers $\{\mu_{3,j,t}\}$, respectively, we can obtain the following LR problem:

$$(\mathrm{LR})\ L(\mu) = \min G_L = \min(G + F_1)$$

where G was previously provided and

$$F_1 = F_{11} + F_{12} \tag{12}$$

$$F_{11} = \sum_{j=1}^{S-1} \sum_{t=1}^{T} \mu_{3,j,t} g_{3,j,t} \tag{13}$$

$$F_{12} = \sum_{n=1}^{N} \sum_{i,i+1 \in \Omega_n} \mu_{1,i} g_{1,i} + \sum_{k=1}^{M_S} \sum_{n=b(k-1)+2}^{b(k)} \mu_{2,n} g_{2,n} \tag{14}$$

where $g_{3,j,t} = \sum_{i \in \Omega} \sum_{\tau = \max\{t - P_{i,j}+1, 1\}}^{t} x_{i,j,\tau} - M_j$, $g_{1,i} = c_{i+1,S} - c_{i,S} - P_{i+1,S}$, $g_{2,n} = c_{s(n-1),S} + P_{s(n-1)+1,S} + Su - c_{s(n-1)+1,S}$, subject to (2)–(5), (9)–(11), and

$$\mu_{3,j,t} \ge 0, j \in \{1, 2, \ldots, S-1\}, 1 \le t \le T \tag{15}$$

$$\mu_{1,i} \in \mathbf{R}, s(n-1) < i < s(n), 1 \le n \le N \tag{16}$$

$$\mu_{2,n} \ge 0, b(k-1) + 1 < n \le b(k), 1 \le k \le M_S. \tag{17}$$

Here, μ is a vector of Lagrangian multipliers $\{\mu_{1,i}, \mu_{2,n}, \mu_{3,j,t}\}$. As a function of the multipliers, $L(\mu)$, is the LR problem. The corresponding LD problem is

$$(\mathrm{LD})\ \max L(\mu) = \max \min G_L,$$

subject to (2)–(5), (9)–(11), and (15)–(17).

For a given multiplier μ, the LR problem can be decomposed into the job-level problems $(\mathrm{LR_i})$,

$$L(\mu) = \sum_{i \in \Omega} L_i(\mu). \tag{18}$$

The subproblem for job $i(i \in \Omega)$ is given as follows:

$$(\mathrm{LR_i}):\ L_i(\mu) = \min\left(\sum_{j=1}^{S} f_{i,j}(\mu, x)\right)$$

subject to (2)–(5) and (9)–(11), where

$$f_{i,j}(\mu, x) = w_{ij}c_{ij} + \bar{f}_{i,j}(x) + \bar{f}_{i,j}(P) \tag{19}$$

$$w_{i,j} = \begin{cases} -W_j, & i \in \Omega, j = 1, \\ W_{j-1} - W_j, & i \in \Omega, 1 < j < S \end{cases} \tag{20}$$

$$w_{i,S} = \begin{cases} \hat{W}_S - \mu_{1,i}, & i \in I_1 \\ \hat{W}_S - \mu_{1,i} + \mu_{1,i-1} & i \in I_2 \\ \hat{W}_S + \mu_{1,i-1} + \mu_{2,n}, & (i,n) \in I_3 \\ \hat{W}_S - \mu_{1,i} - \mu_{2,n-1}, & (i,n) \in I_4 \\ \hat{W}_S + \mu_{1,i-1}, & (i,n) \in I_5 \end{cases} \tag{21}$$

$$\bar{f}_{i,j}(x) = \begin{cases} \sum_{t=1}^{T} \sum_{\tau=t}^{t+P_{ij}-1} x_{i,j,t}\mu_{3,j,\tau} & 1 \le j < S \\ 0, & j = S \end{cases} \tag{22}$$

$$\bar{f}_{i,j}(P) = \begin{cases} -W_{i,j}P_{i,j+1}, & (i,j) \in I_6 \\ -\sum_{t=0}^{T} \mu_{3,j,t}M_j, & 1 \le j < S \\ -\mu_{1,i}P_{i+1,j}, & (i,j) \in I_7 \\ \mu_{2,n}(P_{i+1,j} + Su), & (i,j,n) \in I_8 \end{cases} \tag{23}$$

where $\hat{W}_S = W_S + W_{S-1}$, $I_1 = \{i | i = s(n) + 1, n = b(k-1), 1 \le k \le M_S\}$, $I_2 = \{i | s(n-1) + 1 < i < s(n), b(k-1) < n \le b(k), 1 \le k \le M_S\}$, $I_3 = \{(i,n) | i = s(n), b(k-1) < n < b(k), 1 \le k \le M_S\}$, $I_4 = \{(i,n) | i = s(n-1) + 1, b(k-1) + 1 < n \le b(k), 1 \le k \le M_S\}$, $I_5 = \{(i,n) | i = s(n), n = b(k), 1 \le k \le M_S\}$, $I_6 = \{(i,j) | i \in \Omega, 1 \le j < S\}$, $I_7 = \{(i,j) | s(n-1) < i < s(n), 1 \le n \le N, j = S\}$ and $I_8 = \{(i,j,n) | i = s(n), b(k-1) < n < b(k), 1 \le k \le M_j, j = S\}$.

B. Valid Inequalities

In the above relaxation strategy, because of the different transportation times between two machines in constraints (4), the continuous-casting constraints (6)–(7), which adjoin different jobs on the same caster, have to be relaxed to decompose the relaxed problem into job-level scheduling problems. If we did not relax constraints (6) and (7), we may obtain a tighter valid condition for the LR by simplifying constraints (4). One possible method is to replace constraints (4) with the minimum-transportation-time constraints (24) by solving the following linear programming (LP) problem:

$$(\mathrm{LP})\ \min f = W_S \sum_{i \in \Omega} c_{i,S}$$

subject to (6), (7), (9), and

$$c_{i,j+1} - c_{ij} - P_{i,j+1} - T_{j,j+1} \ge 0, i \in \Omega, 1 \le j < S \tag{24}$$

where $T_{j,j+1} = \min_{1 \le k_1 \le M_j, 1 \le k_2 \le M_{j+1}} \{T_{j,k_1}^{j+1,k_2}\}$.

Then, using the solution of the LP problem, we can derive the following valid inequalities.

Proposition 1: Let $\bar{c}_{i,S}(i \in \Omega)$ be the solution of the LP problem, then the solutions $c_{i,S}$ of the primal problem satisfy

$$c_{i,S} \ge \bar{c}_{i,S}, i \in \Omega. \tag{25}$$

Remark 2: Because the relaxed problem does not consider the continuous-casting constraints (6) and (7), multiple jobs in a batch may be processed on the same caster in the same time interval in the relaxed problem, which violates the continuous casting constraints (6) and (7). To reduce this violation, we added valid inequalities (25) by solving an LP problem considering the continuous casting constraints. In other words, the solution of the primal problem satisfies the valid inequalities (25), whereas the solution of the relaxed problem may not.

C. Solution of the Subproblem $(\mathrm{LR_i})$

Although there are only precedence constraints in the subproblem $(\mathrm{LR_i})$, the transportation times between two machines are different. We must compute the optimums of the subproblems corresponding to different routes and choose the best one. Fortunately, the number of machines at each stage is small in practical production. The number of different routes for a job is $|R| = \prod_{j=1}^{S-1} M_j$. For a given route, the subproblem $(\mathrm{LR_i})$ is a tractable job-level problem, which can be solved using a backward dynamic programming [6]. We introduce the following symbols to describe the dynamic programming algorithm.

Let $\hat{f}_{i,j}(\mu, t)$ be the cost to start the operation of job i at stage j at time period t such that

$$\hat{f}_{i,j}(\mu, t) = \begin{cases} f_{i,j}(\mu, x)|_{x_{ijt}=1}, & \underline{T}_{i,j} \le t \le \bar{T}_{i,j} \\ +\infty, & \text{others} \end{cases}$$

where $\underline{T}_{i,1} = 1$, $\underline{T}_{i,j} = \underline{T}_{i,j-1} + P_{i,j} + T_{j-1,j} (1 < j < S)$, $\underline{T}_{i,S} = \bar{c}_{i,S} - P_{i,S} + 1$, $\overline{T}_{i,j} = \overline{T}_{i,j+1} - P_{i,j} - T_{j,j+1} (1 \le j < S)$, $\bar{T}_{i,S} = T - P_{i,s} + 1$.

Let $h_{i,j}(\mu, r, t)$ be the optimal criterion value of state (μ, r, t) for the operation of job i at stage j. Then, for each route $r(y_{i,r} = 1)$, the backward dynamic programming recursion for each job-level subproblem is expressed as

$$h_{i,S}(\mu, r, t) = \min\left\{\hat{f}_{i,S}(\mu, t), h_{i,S}(\mu, r, t+1)\right\} \tag{26}$$

$$h_{i,j}(\mu, r, t) = \min\Big\{h_{i,j}(\mu, r, t+1), \hat{f}_{i,j}(\mu, t) + h_{i,j+1}(\mu, r, t + P_{i,j} + y_{i,r}T_{r,j})\Big\} \tag{27}$$

where $t \in \mathrm{T}_{i,j}$, $\mathrm{T}_{i,j} = \{t | \underline{T}_{i,j} \le t \le \bar{T}_{i,j}\}$, and $h_{i,j}(\mu, r, t) = +\infty$ for $t \notin \mathrm{T}_{i,j}$, $1 \le j \le S$.

Let $L_i(\mu, r) = \min\{h_{i,1}(\mu, r, t) | \underline{T}_{i,1} \le t \le \bar{T}_{i,1}\}$, then

$$L_i(\mu) = \min\left\{L_i(\mu, r) | 1 \le r \le |R|\right\}. \tag{28}$$

From the dynamic programming approach (26)–(27), we know that the complexity to compute $L_i(\mu, r)$ is $O(ST)$. The overall complexity is then $O(ST|\Omega||R|)$.

D. Construction of a Feasible Solution to the Primal Problem

The solution of the dual problem is usually associated with an infeasible schedule because some of the relaxed constraints

cannot be satisfied. Based on a solution of the relaxed problem, a two-phase heuristic is presented to construct a feasible solution.

In the first phase, the jobs that are processed on the same caster follow the same route. Then, the route for each job is determined by minimizing the sum of the transportation time of all jobs, which can be solved using an enumeration method with time complexity $O(\sum_{j=1}^{S-1} M_j M_{j+1})$. This phase can determine the variables $\{y_{i,r}\}$ and the jobs that are processed on each machine.

In the second phase, the solution of the relaxed problem is adjusted to ensure that the precedence constraints among the jobs on each machine are satisfied. In stage $j(1 \le j < S)$, the processing order of the jobs that are processed on the same machine is identical to the ascending order of the starting time $\{t_{ij}\}(t_{ij} = c_{ij} - P_{i,j})$ of the LR problem. For example, because the jobs on each machine $\Omega_{j,k}(1 \le j < S, 1 \le k \le M_j)$ is known according to the first phase, the processing sequence of jobs in $\Omega_{j,k}$ is $[1], [2], \ldots, [|\Omega_{j,k}|]$ according to the ascending order of their starting times of the LR problem. Then, the jobs satisfy the following precedence constraints:

$$c_{[i+1],j} \ge c_{[i],j} + P_{[i+1],j}, 1 \le i < |\Omega_{j,k}|. \tag{29}$$

Thus, we have determined the route $\{y_{i,r}\}$ and the machine capacity constraints (29), so the completion time $\{c_{ij}\}$ of each operation can be obtained by solving a linear programming problem subject to constraints (4), (6), (7), (9), and (29).

IV. Subgradient Optimization Methods

Before providing our improvements, we first introduce the conventional subgradient optimization algorithms for the LD problem as follows:

$$\mu_{m+1} = \mathbf{P}_{\Phi}(\mu_m - \alpha_m g(\mu_m)), \quad m = 1, 2, \ldots \tag{30}$$

where $\mathbf{P}_{\Phi}(x) = \arg\min_{y \in \Phi} \|y - x\|^2$, $g(\mu_m)$ is the subgradient of $F(\mu_m)$ and $F(\mu_m) = -L(\mu_m)$.

In LR schemes, only one subgradient of $F(\mu)$ at μ is immediately available, i.e., let $g(\mu) = (g_1(\mu), g_2(\mu)) \in \partial F(\mu)$, where $g_1(\mu)$ corresponds to the inequality constraints and $g_2(\mu)$ corresponds to the equality constraints. $\mu \in \Phi = \{(\gamma_1, \gamma_2) | \gamma_1 \in \Phi_1, \gamma_2 \in \Phi_2\}$, $\Phi_1 = \{\gamma_1 | \gamma_1 \ge 0, \gamma_1 \in \mathbf{R}^{n_1}\}$, $\Phi_2 = \{\gamma_2 | \gamma_2 \in \mathbf{R}^{n_2}\}$, γ_1 is the corresponding multiplier with inequality constraints such as the machine capacity constraints (4) and the setup time constraints (7), γ_2 is the corresponding multiplier with equality constraints such as the continuous casting constraints (6), n_1 is the number of inequality constraints, and n_2 is the number of equality constraints.

The convergence of the conventional subgradient algorithm is usually slow because of two types of zigzagging phenomena: interior zigzagging and boundary zigzagging phenomenon. The former type occurs in the interior area of the feasible region, whereas the other occurs at the boundary of the feasible region. The major cause of the first zigzagging phenomenon is that the current subgradient vector forms an obtuse angle with the previous direction of motion [18] (as shown in Fig. 1), whereas the second zigzagging phenomenon occurs because the subgradient

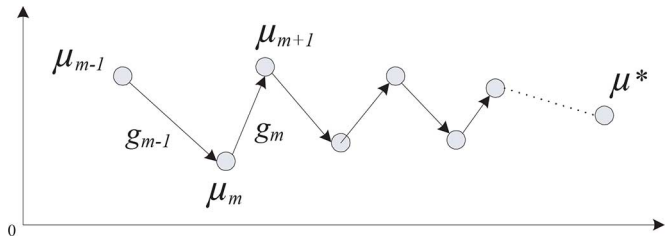


Fig. 1. Interior zigzagging phenomenon.

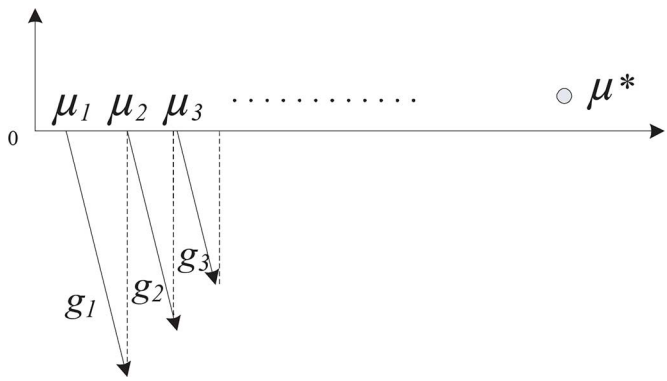


Fig. 2. Boundary zigzagging phenomenon.

are almost perpendicular to the "face" of the feasible region, which is the tangent cone of the feasible region [20] (as shown in Fig. 2). In this case, the iterate points may be almost fixed, which results in notably convergence. Most existing methods such as [18], [20] only consider one type of zigzagging phenomenon. This paper develops a deflected conditional subgradient method to handle the two types of zigzagging phenomena in a unified framework.

A. Deflected Conditional Subgradient

1) Conditional Subgradient: In the conventional subgradient method, the search direction did not consider the feasible region of the multipliers such as non-negativity, which may cause the iterate point to reach beyond the boundaries and may be almost perpendicular to the "face" of the feasible region, which is the tangent cone of the feasible region (as shown in Fig. 2). The subgradient is an inefficient search direction at this moment. To avoid such invalid search, we must generalize the definition of the subdifferential considering the feasible region. The subdifferential of F at μ is defined as

$$\partial F(\mu) = \{g \in \mathbf{R}^n | F(\mu_1) \ge F(\mu) + g^T(\mu_1 - \mu), \ \forall \mu_1 \in \mathbf{R}^n\}$$

where $n = n_1 + n_2$. The generation of subdifferential is the conditional subdifferential

$$\partial^{\Phi} F(\mu) = \{g \in \mathbf{R}^n | F(\mu_1) \ge F(\mu) + g^T(\mu_1 - \mu), \ \forall \mu_1 \in \Phi\}$$

$\mu \in \Phi$, the element of which will be referred to as the conditional subgradients. Obviously, $\partial^{\Phi} F(\mu) \supseteq \partial F(\mu)$ for all $\mu \in \Phi$.

Based on the characterization of the conditional subdifferential [35] and the projection method [20], we can calculate the

conditional subgradient as follows. For convenience, we denote $g = g(\mu), g_1 = g_1(\mu)$ and $g_2 = g_2(\mu)$.

Proposition 2: The conditional subgradient $\hat{g}(\mu)$ of $F(\mu)$ can be given as follows:

$$\hat{g}(\mu) = (\hat{g}_1, \hat{g}_2) = \mathbf{P}_{-T_\Phi(\mu)}(g) \tag{31}$$

where $\hat{g}_1 = \begin{cases} 0, & \text{if } g_{1,i} \geq 0 \text{ and } \mu_{1,i} = 0, \\ g_{1,i}, & \text{others}, \end{cases} \mu_1 \in \Phi_1, \hat{g}_2 = g_2$ and $T_\Phi(\mu)$ is the tangent cone of Φ at μ.

Remark 3: Interestingly, the formulation of the conditional subgradient is identical to the modifications of the projections of subgradients in [36]–[39], which can explain that why their modifications can speed up the convergence of the pure subgradient optimization method.

2) Deflected Subgradient: In the conventional subgradient method, an unpleasant behavior is that the two adjacent subgradients form an obtuse angle in iterations (as shown in Fig. 1), which implies that the search is inefficient. To improve the efficiency, we can deflect the subgradient direction whenever it forms an obtuse angle with the previous stepping direction [18] proposed a modification of the pure subgradient method, where the subgradient g_m is replaced by a deflected subgradient $d_m = g_m + \beta_m d_{m-1}$, where $\beta_m \geq 0$ is called a deflection parameter and $d_{m-1} = 0$ for $m = 0$. Thus, the deflected subgradient method for the LD problem is

$$d_m = g_m + \beta_m d_{m-1} \tag{32}$$

$$\mu_{m+1} = \mathbf{P}_\Phi(\mu_m - \alpha_m d_m),\ m = 0, 1, 2, \ldots \tag{33}$$

where g_m is the subgradient of $F(\mu_m)$ at μ_m.

There are various forms of the choices of the deflection parameter to ensure that two adjacent subgradients form an acute angle. Two deflection strategies are commonly used to determine the deflection parameters: the Camerini-Fratta-Maffioli deflection strategy (CFMDS) [18] and the average direction strategy (ADS) [19]. Then, we can obtain the following proposition.

Proposition 3: Suppose that the deflected subgradient d_m is given by the deflected subgradient optimization method, the step length is $\alpha_m > 0$, and the deflection parameter β_m is given by either

$$\beta_m = \begin{cases} -\tau_m g_m^T d_{m-1} / \|d_{m-1}\|^2, & g_m^T d_{m-1} < 0 \\ 0, & g_m^T d_{m-1} \geq 0 \end{cases} \tag{34}$$

where $1 < \tau_m < 2$, or

$$\beta_m = \|g_m\| / \|d_{m-1}\| \tag{35}$$

where (34) is CFMDS and (35) is ADS, then $d_{m-1}^T d_m \geq 0$, which means that the two consecutive subgradients that are generated by the deflected subgradient optimization method form an acute angle.

This paper selects $\tau_m = 1.5$ for all m according to the suggestion in [18]. It is worth noting that the above strategies have different effects on the subgradient algorithms, of which the details are described in Section V-B1.

3) Deflected Conditional Subgradient: Based on the above analysis and results, we present a new algorithm to address both types of zigzagging phenomena. The algorithm is first to transform the subgradient vector into a feasible direction (or project the subgradient vector into the tangent cone of the feasible region at the current iterative point) as in (31) and subsequently, deflect the feasible direction to form an acute angle with the previous direction of motion as in (32). Because the deflected direction may be infeasible, it should be transformed into a feasible direction. We call this algorithm deflected conditional subgradient (DCS) algorithm, which is given as follows:

$$d_m = \hat{g}_m + \beta_m \hat{d}_{m-1} \tag{36}$$

$$\hat{d}_m = \mathbf{P}_{-T_\Phi(\mu)}(d_m) \tag{37}$$

$$\mu_{m+1} = \mathbf{P}_\Phi(\mu_m - \alpha_m \hat{d}_m),\ m \geq 0 \tag{38}$$

where $\hat{g}_m$ is the conditional subgradient of $F(\mu_m)$ at μ_m as in (31), $\beta_m \geq 0$ is the deflection parameter, and $T_\Phi(\mu)$ is the tangent cone of Φ at μ.

B. Deflected Conditional Subgradient Level Algorithm

Based on the DCS algorithm, an improved simple subgradient level algorithm is developed to replace the strict convergent condition by estimating the optimal dual value dynamically. We call this algorithm the deflected condition subgradient level (DCSL) algorithm. Intuitively, the algorithm always overestimates the optimal dual function and dynamically reduces the overestimation until it converges to the optimum. This adjustment method of the overestimation is essentially identical to the Brannlund's level control strategy [15], [16]. We adopt the overestimation strategy because the underestimation of the optimum dual function will lead to early convergence in the subgradient algorithms [2]. Furthermore, the efficiency of this algorithm can be enhanced by introducing the DC-subgradient.

DCSLA: Let $P(\mu)$ be a primal function value of μ, which is derived using the heuristic that was presented in Section III-D. Let $F(\mu) = -L(\mu), g(\mu)$ be the subgradient of $F(\mu), \hat{g}(\mu)$ be defined as (30) and $d(\mu)$ be defined as (36). Denote $g_m = g(\mu_m), \hat{g}_m = \hat{g}(\mu_m)$ and $\hat{d}_m = \hat{d}(\mu_m)$.

Step 1. (Initialization) Select $\varepsilon_1 > 0, \varepsilon_2 > 0, \beta \in (0, 1), \delta_1 > 0, \sigma_{\max} > 0, W > 2$ (W is an even number), and $t \in (0, 1)$. Set $\mu_0 = 0, \mu_{\text{best}} = \mu_0, r = 0, \sigma_1 = 0, F_{\text{rec}}^{-1} = \infty, m = 0, l = 1, s = 1, \hat{d}_{-1} = 0, M[l] = m$, and $h(l) = 1/(l + 1)$. Calculate $P(\mu_0)$ and set $P_{\text{best}} = P(\mu_0)$.

Step 2. (Function evaluation)

Step 2.1: If $F(\mu_m) < F_{\text{rec}}^{m-1}$, set $F_{\text{rec}}^m = F(\mu_m)$ and $\mu_{\text{best}} = \mu_m$; Otherwise, set $F_{\text{rec}}^m = F_{\text{rec}}^{m-1}$ (so that $F_{\text{rec}}^m = \min\{F(\mu_j) | 0 \leq j \leq m\}$).

Step 2.2: If $P_{\text{best}} < P(\mu_m)$, then set $P_{\text{best}} = P(\mu_m)$.

Step 2.3: If $\bar{P} + F(\mu_m) < \delta_l$, then set $\delta_l = P(\mu_m) + F(\mu_m), \hat{d}_{m-1} = 0$.

Step 3. (Sufficient descent) If $F(\mu_k) \leq F_{\text{rec}}^m - 0.5\delta_l$, then set $M[l + 1] = m, \sigma_m = 0, \hat{d}_{m-1} = 0, \delta_{l+1} = \delta_l, h(l + 1) = h(l), l = l + 1$.

Step 4. (Small overestimation detection) Set $r = r + 1, D[r] = F(\mu_m)$.

If $r > W$ and $\sum_{n=1}^{W-2} |2(D[n + 2] - D[n])| / |WD[n]| < \varepsilon_2$, set $m_1 = m + \lfloor (h(l)\sigma_{\max} - \sigma_m + 1) / \|\hat{d}_m\| \rfloor + 1, \sigma_{m_1} =$

$h(l)\sigma_{\max}+1, F_{\text{rec}}^{m_1}=F_{\text{rec}}^{m}, \mu_{m_1}=\mu_m, \hat{d}_{m_1}=\hat{d}_m$, and $m=m_1$.

If $r>W$, set $r=0$.

Step 5. (Large overestimation detection) If $\sigma_m > h(l)\sigma_{\max}$, then $M[l+1]=m, \mu_m=\bar{\mu}, \sigma_m=0, \hat{d}_{m-1}=0, \delta_{l+1}=\beta\delta_l, h(l)=1/(l+1), l=l+1, s=s+1$.

Step 6. (Calculation of the subgradient and multipliers) Let $F_{\text{lev}}^{m}=F_{\text{rec}}^{M[l]}-\delta_l$. Update the multipliers as (36)–(38), where $\alpha_m=t(F(\mu_m)-F_{\text{lev}}^{m})/\|\hat{d}_m\|^2$.

Step 7. (Accumulation of iterative paths) Set $\sigma_{m+1}=\sigma_m+\|\alpha_m\hat{d}_m\|, m=m+1$.

Step 8. (Termination check) If $|\delta_l/F_{\text{rec}}^{M[l]}|<\varepsilon_1$ or $\|\hat{d}_m\|<\varepsilon_1$ is satisfied, then stop the iteration; otherwise, go to Step 2.

Remark 4: In the above algorithm, δ_1 is the estimation of $F(\mu)-F^*$ and $\sigma_{\max}$ is the estimation of $\|\mu_0-\mu^*\|$. If we use δ_1 to estimate $F(\mu)-F^*$, then set $\sigma_{\max}=\delta_1/\|g(\mu_0)\|$ according to the accumulation of iterative paths (Step 7). This condition is realizable because $P(\mu)$ can provide a lower bound of F^* according to the dual theory [2]. Hence, we can set $\delta_1=\tau(F(\mu_0)+P(\mu))(\tau\in(0,1])$.

Remark 5: In the late iterations of the DCSLA, $F(\mu_m)$ will weakly oscillate between two points or within a local area because of the overestimation strategy. This phenomenon cannot be eliminated by the deflected-conditional subgradient because it only changes the search direction. To improve the efficiency, Step 4 detects the weak oscillation by recording the history information and directly accumulating sufficient number of paths to adjust the estimation level. To guarantee the global convergence, Step 5 adjusts the estimation level F_{lev}^{m} under the large-overestimation condition. The detailed convergence analysis is provided in the Appendix.

Compared with other step-length methods [14], [15], [17], this method is simpler and has fewer limitations such as the *priori* distance between the initial point and the optimal point. Moreover, this method introduces a new subgradient to improve the convergence. The numerical results can confirm this improvement in Section V. Hence, the DCSLA inherits the simplicity and low computational complexity of the conventional subgradient algorithm and makes up for its shortcomings such as the strict convergence condition and strong zigzagging phenomena.

Proposition 4: For the DCSLA, we have $\inf_{m\geq 0} F(\mu_m)=F^*$.

The detailed proof is provided in the Appendix.

V. Computational Results

A. Algorithm Parameters

To analyze the performance of the presented algorithm for the SCC scheduling problem, our computational study compares six methods. For simplicity, we denote the subgradient level algorithm as L, the standard subgradient as S, the conditional subgradient as C, the Camerini-Fratta-Maffioli deflection strategy as F, and the average direction strategy as A,

LC	the algorithm L that adopts the method C;
LCF	the algorithm L that adopts the methods C and F;
LCA	the algorithm L that adopts the methods C and A.

The other three methods LS, LSF, and LSA have similar meanings.

The initial parameters of the algorithms are provided as follows: $\varepsilon_1=1e-3, \varepsilon_2=1e-5, W=4, t=0.8, \beta=0.8, \delta_1=(P(\mu_0)+F(\mu_0))/5, \sigma_{\max}=(P(\mu_0)+F(\mu_0))/\|g(\mu_0)\|$.

Remark 6: Because the gaps of real scheduling problems are larger than 1% and $\delta_l\to 0$ according to the convergence proof, we set $|\delta_l/F_{\text{rec}}^{M[l]}|<\varepsilon_1=10^{-3}$ as the stopping criterion. In the late iterations of the DCSLA, the oscillation range of $F(\mu_m)$ is small, so we set $\varepsilon_2=10^{-5}\ll\varepsilon_1$ as the criterion of the small overestimation. Here we set $t=\beta=0.8, W=4$ and $\delta_1=(P(\mu_0)+F(\mu_0))/5$ based on our experimental experiences. Correspondingly, we set $\sigma_{\max}=(P(\mu_0)+F(\mu_0))/\|g(\mu_0)\|$ to estimate the distance between μ_0 and μ^*.

All algorithms are evaluated using the following performance measures: relative duality gap, iteration numbers, and computational time. The relative duality gap $g=(UB-LB)/LB\times 100\%$ is used as a criterion to measure the suboptimality, where UB is the upper bound, which is derived from the modified feasible solution, and LB is the lower bound, which is obtained from the solution of the dual problem. All algorithms are implemented in C# language and run on a PC with Intel Core i7-2600 3.4 GHz CPU using the Windows 7 operative system (64 bit).

B. Problem Instances and Computational Results

1) Problem Instances: By carefully analyzing the actual production data from Baosteel Complex of China, we generate a total of 1440 test instances. The problem structures are described as follows.

(1) The number of stages varies at two levels: 3 and 4. The number of machines at each stage varies at three levels: 3, 4, and 5. The number of batches on each machine at the last stage varies at four levels: 2, 3, 4, and 5. The number of jobs in each batch varies at six levels: 3, 4, 5, 6, 7, and 8.

(2) The objective function coefficients $W_j=10+30(j-1)(1\leq j\leq S)$. The setup time is $Su=80$. The integer of the transportation time is uniformly generated in the range of [3, 10]. The integer of the processing time P_{ij} is uniformly generated in the range of [36, 50].

We consider six combinations of the stage and machine levels (Stages versus Machines): 3 versus 3, 3 versus 4, 3 versus 5, ..., 4 versus 5. There are $4\times 6=24$ pairs of the batch and job levels in each combination of the stage and machine levels. For each pair of the batch and job levels in the above combination configurations, we randomly generate ten instances, which results in a total of $2\times 3\times 4\times 6\times 10=1440$ test instances.

2) Computational Results: Because the full details of the test results are notably substantial, we will summarize these results in more compact tables and statistic results. For simplicity, we denote the Stages as S, Machines as M, Batches as B, and Jobs as J. First, we consider a representative combination of the stage and machine level (i.e., S versus M = 3 versus 4).

Tables I and II provide the gaps and the lower bounds of six algorithms, respectively. Table III presents the iteration numbers and running times of the six algorithms. From the compu-

TABLE I
GAPS (%) OF SIX ALGORITHMS (S VERSUS M = 3 VERSUS 4)

B vs. J	LS	LSA	LSF	LC	LCA	LCF
2 vs. 3	3.00	3.02	3.00	2.51	2.47	2.49
2 vs. 4	3.00	3.01	3.00	2.71	2.69	2.69
2 vs. 5	3.32	3.33	3.32	3.04	3.04	3.02
2 vs. 6	4.76	4.78	4.76	4.49	4.41	4.45
2 vs. 7	3.29	3.30	3.29	3.10	3.10	3.11
2 vs. 8	3.89	3.90	3.89	3.75	3.73	3.74
3 vs. 3	1.86	1.87	1.86	1.63	1.62	1.61
3 vs. 4	1.95	1.95	1.95	1.78	1.78	1.77
3 vs. 5	2.47	2.48	2.47	2.34	2.34	2.33
3 vs. 6	2.48	2.49	2.48	2.38	2.38	2.38
3 vs. 7	3.11	3.12	3.11	3.00	3.01	2.99
3 vs. 8	2.73	2.74	2.73	2.64	2.65	2.64
4 vs. 3	1.72	1.72	1.72	1.56	1.56	1.55
4 vs. 4	1.79	1.80	1.79	1.69	1.69	1.69
4 vs. 5	1.98	1.98	1.98	1.89	1.88	1.88
4 vs. 6	1.75	1.76	1.75	1.70	1.70	1.70
4 vs. 7	2.10	2.10	2.10	2.04	2.05	2.04
4 vs. 8	2.34	2.34	2.34	2.29	2.30	2.29
5 vs. 3	1.46	1.46	1.46	1.36	1.36	1.35
5 vs. 4	1.55	1.55	1.55	1.45	1.46	1.45
5 vs. 5	1.51	1.51	1.51	1.45	1.46	1.46
5 vs. 6	1.63	1.63	1.63	1.57	1.59	1.58
5 vs. 7	1.81	1.81	1.81	1.76	1.77	1.75
5 vs. 8	2.02	2.02	2.02	1.98	1.99	1.98
Average	2.40	2.40	2.40	2.25	2.25	2.25

TABLE II
LOWER BOUNDS OF SIX ALGORITHMS (S VERSUS M = 3 VERSUS 4)

B vs. J	LS	LSA	LSF	LC	LCA	LCF
2 vs. 3	475985	475919	475985	478224	478457	478342
2 vs. 4	731231	731159	731231	733322	733487	733449
2 vs. 5	1040481	1040370	1040481	1043245	1043242	1043506
2 vs. 6	1380007	1379849	1380007	1383638	1384608	1384173
2 vs. 7	1780481	1780286	1780481	1783588	1783613	1783585
2 vs. 8	2222127	2221876	2222127	2225086	2225445	2225316
3 vs. 3	978479	978439	978479	980738	980801	980876
3 vs. 4	1520100	1520022	1520100	1522648	1522650	1522815
3 vs. 5	2164379	2164273	2164379	2167196	2167069	2167341
3 vs. 6	2922329	2922147	2922329	2925194	2925243	2925287
3 vs. 7	3760583	3760297	3760583	3764675	3764174	3764900
3 vs. 8	4731805	4731451	4731805	4735863	4735715	4736079
4 vs. 3	1651004	1650926	1651004	1653586	1653631	1653743
4 vs. 4	2579745	2579637	2579745	2582344	2582294	2582481
4 vs. 5	3693037	3692856	3693037	3696452	3696460	3696549
4 vs. 6	5021266	5021120	5021266	5024103	5023810	5024088
4 vs. 7	6513710	6513365	6513710	6517482	6517044	6517392
4 vs. 8	8203542	8203126	8203542	8207198	8206540	8207015
5 vs. 3	2496160	2496093	2496160	2498575	2498510	2498773
5 vs. 4	3917616	3917457	3917616	3921261	3921040	3921416
5 vs. 5	5691064	5690869	5691064	5694185	5693779	5693966
5 vs. 6	7669578	7669302	7669578	7673786	7672890	7673546
5 vs. 7	9991441	9991002	9991441	9996271	9995027	9996559
5 vs. 8	12606384	12605797	12606384	12610596	12609995	12610843
Average	3905939	3905735	3905939	3909136	3908980	3909252

TABLE III
ITERATION NUMBERS AND RUNNING TIMES OF SIX ALGORITHMS (S VERSUS M = 3 VERSUS 4)

B vs. J	Iteration Numbers						Running times (s)					
	LS	LSA	LSF	LC	LCA	LCF	LS	LSA	LSF	LC	LCA	LCF
2 vs. 3	78	94	78	46	72	45	1.8	2.2	1.8	1.0	1.6	1.1
2 vs. 4	74	90	74	39	51	40	2.8	3.4	2.8	1.5	1.9	1.5
2 vs. 5	77	95	77	41	57	45	4.3	5.3	4.2	2.3	3.2	2.5
2 vs. 6	90	113	90	71	106	70	6.8	8.8	6.9	5.5	8.1	5.4
2 vs. 7	78	93	78	48	61	50	7.6	9.3	7.6	4.7	6.0	5.0
2 vs. 8	82	102	82	68	88	72	10.1	12.8	10.2	8.4	11.0	9.0
3 vs. 3	59	72	59	25	26	24	2.9	3.6	2.9	1.3	1.3	1.2
3 vs. 4	62	75	62	25	27	24	5.1	6.3	5.1	2.1	2.2	2.0
3 vs. 5	67	81	67	32	32	31	8.0	9.8	8.1	3.8	3.9	3.7
3 vs. 6	68	84	68	31	36	33	11.2	13.9	11.5	5.2	6.1	5.6
3 vs. 7	72	89	72	41	50	43	15.7	19.2	15.7	8.9	10.8	9.3
3 vs. 8	71	87	71	34	50	37	19.5	23.9	19.6	9.3	13.7	10.1
4 vs. 3	58	69	58	26	28	26	5.2	6.1	5.1	2.2	2.5	2.3
4 vs. 4	60	72	60	23	26	25	8.4	10.5	8.7	3.4	3.7	3.5
4 vs. 5	62	75	62	26	31	28	12.9	15.7	13.0	5.4	6.5	5.8
4 vs. 6	61	73	61	24	25	23	17.7	21.1	17.9	6.8	7.3	6.6
4 vs. 7	64	78	64	30	34	30	24.6	30.0	24.6	11.5	13.2	11.5
4 vs. 8	66	79	66	31	40	34	32.0	38.8	32.2	15.4	19.7	16.9
5 vs. 3	52	63	52	21	21	22	7.2	8.7	7.1	3.0	2.8	3.0
5 vs. 4	55	66	55	22	25	24	12.3	14.8	12.1	5.0	5.5	5.3
5 vs. 5	55	67	55	22	23	22	18.5	21.9	18.4	7.3	7.6	7.3
5 vs. 6	60	72	60	25	25	24	25.6	30.9	25.3	10.6	10.4	10.3
5 vs. 7	61	74	61	31	35	30	33.0	40.8	33.2	16.8	19.1	16.0
5 vs. 8	64	77	64	31	35	30	44.3	53.6	44.7	21.8	24.7	20.6
Average	66	81	66	34	42	35	14.1	17.1	14.1	6.8	8.0	6.9

tational results of the three tables, we can obtain the following observations.

(1) The LR approach for the SCC scheduling problems can obtain high-quality solutions within acceptable computational times. The overall average gaps of the six algorithms are less than 2.40%, whereas the overall average running times are shorter than 17.1s.

(2) LC, LCA, and LCF perform better than LS, LSA, and LSF in terms of gaps and running times. In particular, the overall average gaps of the former group are all 2.25%, which are less than those of the latter group that are 2.40%. The longest overall average running time of the former group is 8.0 s, whereas the shortest overall running times of the latter group is 14.1 s. Because the only difference between the two groups is the conditional subgradient, we can conclude that the conditional subgradient plays an important role in improving the efficiency.

(3) The deflected subgradient slightly affect the efficiency of the DCSLA compared with the conditional subgradient. The average direction strategy A even worsens the DCSLA. In particular, the running times of the LSA are longer than those of LS and LSF. Similar observations can also be found in LC and LCF. Because the deflected subgradient mainly works in the interior feasible region, the computational comparisons reveal that the entire optimizing process appears occur at boundary of feasible region. After carefully analyzing the multiplier μ_m, we found that $\{\mu_{3,j,t}\}$ contains many zero elements. Moreover, the feasible region of nonnegative multipliers is a polyhedral, which implies that the conditional subgradient direction is parallel to the face at the iterative point. This result can explain why the deflected subgradient does not work in the optimizing process.

Furthermore, the deflection strategy F only deflects the conditional subgradient direction when it forms an obtuse angle with the previous deflected subgradient vector, whereas the deflection strategy A always bisects the current subgradient direction

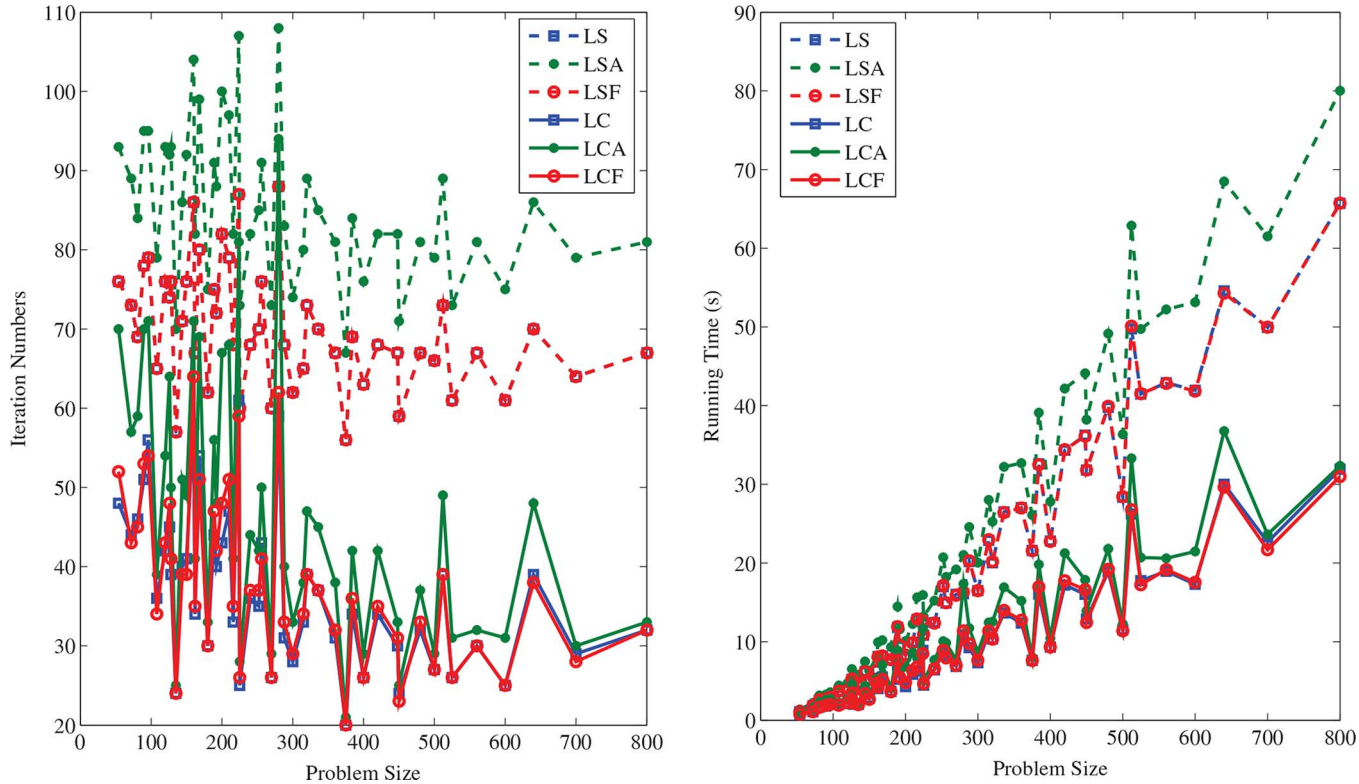


Fig. 3. Comparisons of the iteration numbers and running times of six algorithms.

and the previous deflected direction. The computational comparisons show that the algorithms that adopt the strategy A perform worse than the algorithms that adopt the strategy F. This result indicates that the choices of the deflection strategy are also essential to improve the efficiency.

(4) The LCF has better lower bounds than the other algorithms. In addition, the LCF performs better than the LC for some large-scale problems. Specifically, the running times of LCF for the problems (B versus J: 5 versus 6, 5 versus 7 and 5 versus 8) are shorter than those of LC in Table III.

To compare the computational results of the three combinations of the stage and machine levels, we present the average iteration numbers and running times of six algorithms at various problem sizes (problem size = Machine × Stage × Batch × Job) in Fig. 3. In addition, we provide the minimum, maximum and average value of the duality gaps and iteration numbers of the six algorithms in Table IV. The previous observations also hold in these computational results. It is worth noting that LCF performs best among the six algorithms in terms of gaps and running times.

TABLE IV
DUALITY GAPS (%) AND ITERATION NUMBERS OF SIX ALGORITHMS

S vs. M		Gaps (%)						Iteration Numbers					
		LS	LSA	LSF	LC	LCA	LCF	LS	LSA	LSF	LC	LCA	LCF
3 vs. 3	Min	1.08	1.08	1.08	1.00	1.00	1.00	45	57	45	20	19	19
	Max	4.26	4.28	4.26	3.88	3.83	3.83	88	108	88	70	107	73
	Avg.	2.38	2.38	2.38	2.22	2.21	2.21	65	79	65	35	43	35
3 vs. 4	Min	1.46	1.46	1.46	1.36	1.36	1.35	52	63	52	21	21	22
	Max	4.76	4.78	4.76	4.49	4.41	4.45	90	113	90	71	106	72
	Avg.	2.40	2.40	2.40	2.25	2.25	2.25	67	81	67	34	42	35
3 vs. 5	Min	1.37	1.37	1.37	1.31	1.32	1.31	54	63	54	20	20	20
	Max	4.49	4.50	4.49	4.31	4.29	4.29	88	109	88	76	109	86
	Avg.	2.29	2.29	2.29	2.16	2.16	2.15	66	80	66	33	40	33
4 vs. 3	Min	1.58	1.58	1.58	1.54	1.54	1.54	57	69	57	23	25	24
	Max	4.34	4.36	4.34	3.85	3.80	3.82	92	109	92	90	119	84
	Avg.	2.81	2.82	2.81	2.64	2.63	2.63	71	85	71	39	47	39
4 vs. 4	Min	1.64	1.64	1.64	1.53	1.53	1.53	56	67	56	22	22	21
	Max	4.61	4.63	4.61	4.24	4.22	4.22	87	107	87	62	81	64
	Avg.	2.88	2.89	2.88	2.72	2.72	2.71	71	87	71	38	45	38
4 vs. 5	Min	1.40	1.40	1.40	1.31	1.32	1.31	52	62	52	20	19	19
	Max	4.74	4.76	4.74	4.51	4.51	4.50	93	115	93	73	99	76
	Avg.	2.89	2.90	2.89	2.72	2.73	2.72	72	88	72	37	47	39
Average		2.64	2.64	2.64	2.47	2.47	2.46	69	83	69	36	44	36

VI. CONCLUSION

This paper developed a relaxation approach for the machine capacity constraints to solve the SCC scheduling problem. Some valid inequalities were proposed to strengthen the Lagrangian relaxation. The LD problem was solved using the DCSLA, which improves the efficiency by weakening the phenomena and guarantees global convergence using the Brannlund's level control strategy. The effects of different deflection strategies on the algorithm were also studied. The computational comparisons and careful analysis showed that the optimizing process for the LD problem appears occur at the boundary of the feasible region. The algorithms that adopt the conditional subgradient perform better than the algorithms that adopt the deflected subgradient. Furthermore, the computational results demonstrated that the algorithm LCF performs best among the six algorithms in terms of duality gaps, iteration

numbers and running times. The method can also be applied to other similar scheduling problems. Further research may focus on the improvement of the proposed methods and their applications in the future.

APPENDIX
CONVERGENCE PROOF

Before providing the proof of the global convergence, we introduce several lemmas and notations. Let $F^* = \min_{\mu\in\Phi} F(\mu)$ and $\Phi^* = \{\mu \in \Phi | F(\mu) = F^*\}$. According to (31) and (37), it is easy to see that the deflected conditional subgradient d_m is bounded because the subgradient g_m is bounded. For convenience, we suppose $\|\hat{d}_m\| \le G$ (G is a large constant). It is easy to obtain the following lemma.

Lemma 1: $L(\mu)$ is concave over Φ and $F(\mu) = -L(\mu)$ is convex over Φ.

Proof: This proof can be obtained from [2, Prop. 5.1.2]. □

Lemma 2: (Projection theorem) Given some $\mu \in R^n$, a vector $\bar{\mu} \in \Phi$ is equal to $\mathbf{P}_\Phi(\mu)$ if and only if

$$(\mu_1 - \bar{\mu})^T(\mu - \bar{\mu}) \le 0, \forall \mu_1 \in \Phi. \tag{39}$$

Proof: This proof can be obtained from [2, Prop. B.11]. □

From Lemma 2, it is not difficult to obtain the following lemma.

Lemma 3: Given some vector $\mu_0 \in \Phi, d \in R^n, \alpha$ is positive scalar and $\bar{\mu}_1 = \mu_0 - \alpha d$. Let $\mu_1 = P_\Phi(\bar{\mu}_1)$ and $\bar{g} = \mu_1 - \bar{\mu}_1$, then

$$\bar{g}^T d \ge 0. \tag{40}$$

Lemma 4: Suppose that $\hat{g}_m, d_m$ and $\hat{d}_m$ is given by (30), (36), and (37). Let $\hat{\mu}$ and μ_m be such that $F(\mu_m) > F(\hat{\mu})$. If

$$0 < \alpha_m < (F(\mu_m) - F(\hat{\mu}))/\|\hat{d}_m\|^2, \forall m = 0, 1, 2, \ldots$$

Then

$$\hat{g}_m^T(\hat{\mu} - \mu_m) \ge d_m^T(\hat{\mu} - \mu_m) \ge \hat{d}_m^T(\hat{\mu} - \mu_m) \tag{41}$$

$$\|\mu_{m+1} - \hat{\mu}\|^2 \le \|\mu_m - \hat{\mu}\|^2 - 2\alpha_m(F(\mu_m) - F(\hat{\mu})) + \alpha_m^2\|\hat{d}_m\|^2 \tag{42}$$

for all m.

Proof: According to Proposition 2 and the definition of $\hat{d}_m$, it is not difficult to prove that

$$d_m^T(\hat{\mu} - \mu_m) \ge \hat{d}_m^T(\hat{\mu} - \mu_m), m \ge 0.$$

Therefore, we need only to show that $\hat{g}_m^T(\hat{\mu} - \mu_m) \ge d_m^T(\hat{\mu} - \mu_m)$. We will prove the lemma by induction on m. Clearly, (41) is valid for $m = 0$ with an equal sign. Suppose that the assertion of the theorem is true for some $k = m$. To prove it for $k = m+1$, observe that by

$$d_{m+1}^T(\hat{\mu} - \mu_{m+1}) = \hat{g}_{m+1}^T(\hat{\mu} - \mu_{m+1}) + \beta_{m+1}\hat{d}_m^T(\hat{\mu} - \mu_{m+1}). \tag{43}$$

Hence, we need only to show $\beta_{m+1}\hat{d}_m^T(\hat{\mu} - \mu_{m+1}) \le 0$.

Let $\bar{g}_m = P_\Phi(\mu_m - \alpha_m\hat{d}_m) - (\mu_m - \alpha_m\hat{d}_m)$. Then

$$\begin{aligned} \hat{d}_m^T(\hat{\mu} - \mu_{m+1}) &= \hat{d}_m^T(\hat{\mu} - P_\Phi(\mu_m - \alpha_m\hat{d}_m)) \\ &= \hat{d}_m^T(\hat{\mu} - \mu_m + \alpha_m\hat{d}_m - \bar{g}_m) \\ &= \hat{d}_m^T(\hat{\mu} - \mu_m) + \alpha_m\|\hat{d}_m\|^2 - \hat{d}_m^T\bar{g}_m \\ &\le \hat{d}_m^T(\hat{\mu} - \mu_m) + \alpha_m\|\hat{d}_m\|^2. \end{aligned} \tag{44}$$

The last inequality can be obtained from Lemma 3. Because $F(\mu)$ is convex over Φ, from the definition of conditional subgradient, we can obtain $F(\hat{\mu}) - F(\mu_m) \ge \hat{g}_m^T(\hat{\mu} - \mu_m)$. From $0 < \alpha_m < (F(\mu_m) - F(\hat{\mu}))/\|\hat{d}_m\|^2$, we know that

$$\alpha_m\|\hat{d}_m\|^2 \le F(\mu_m) - F(\hat{\mu}) \le \hat{g}_m^T(\mu_m - \hat{\mu}) \le \hat{d}_m^T(\mu_m - \hat{\mu}).$$

The last inequality follows from the induction hypothesis (41). Hence, it holds that

$$\alpha_m\|\hat{d}_m\|^2 + \hat{d}_m^T(\hat{\mu} - \mu_m) \le 0. \tag{45}$$

This equation and (43) yield

$$\beta_{m+1}\hat{d}_m^T(\hat{\mu} - \mu_{m+1}) \le 0. \tag{46}$$

Because $\beta_{m+1} \ge 0$, from (43) and (46), it follows that $\hat{g}_{m+1}^T(\hat{\mu} - \mu_{m+1}) \ge d_{m+1}^T(\hat{\mu} - \mu_{m+1})$, which implies that (41) holds for all m.

Based on the above results, we have

$$\begin{aligned} &\|\mu_{m+1} - \hat{\mu}\|^2 = \|\mathbf{P}_\Phi(\mu_m - \alpha_m\hat{d}_m) - \hat{\mu}\|^2 \\ &\le \|\mu_m - \hat{\mu}\|^2 - 2\alpha_m\hat{d}_m^T(\mu_m - \hat{\mu}) + \alpha_m^2\|\hat{d}_m\|^2 \\ &\le \|\mu_m - \hat{\mu}\|^2 - 2\alpha_m\hat{g}_m^T(\mu_m - \hat{\mu}) + \alpha_m^2\|\hat{d}_m\|^2 \\ &\le \|\mu_m - \hat{\mu}\|^2 - 2\alpha_m(F(\mu_m) - F(\hat{\mu})) + \alpha_m^2\|\hat{d}_m\|^2. \end{aligned}$$

□

Lemma 5: Assume that the stepsize α_m in (38) is such that

$$\alpha_m > 0, \sum_{m=0}^{\infty}\alpha_m = \infty, \sum_{m=0}^{\infty}\alpha_m^2 < \infty.$$

Then, the sequence $\{\mu_m\}$, which is generated by the deflected conditional subgradient algorithm, converge to an element of Φ^*.

Proof: This proof is similar to the proof of [20] and [40]. It is omitted here. □

Lemma 6: For the DCSL algorithm, we have $l \to \infty$, and either $\inf_{m\ge 0} F(\mu_m) = -\infty$ or $\lim_{l\to\infty}\delta_l = 0$.

Proof: Assume that l takes only a finite number of values, say $l = 1, 2, \ldots, \bar{l}, s = 1, 2, \ldots, \bar{s}$. In this case, from Steps 5 and 7 it can be known that $\sigma_{m+1} = \sigma_m + \|\alpha_m\hat{d}_m\| < \sigma_{\max}/(\bar{s}+1)$ for all $m \ge M[\bar{l}]$, which implies that

$$\lim_{m\to\infty}\|\alpha_m\hat{d}_m\| = t\lim_{m\to\infty}(F(\mu_m) - F_{\text{lev}}^m)/\|\hat{d}_m\| = 0.$$

However, this result is impossible, because for all $m \ge M[\bar{l}]$, we have

$$\alpha_m = t\left[F(\mu_m) - F_{\text{lev}}^m\right]/\|\hat{d}_m\|^2 > t\delta_{\bar{l}}/G^2 > 0.$$

Hence, $l \to \infty$.

Let $\delta = \lim_{l\to\infty} \delta_l$. If $\delta > 0$, then from Steps 3 and 6 it follows that for all sufficiently large l, we have $\delta_l = \delta$ and $F_{\text{rec}}^{M[l+1]} - F_{\text{rec}}^{M[l]} \le -0.5\delta$, which implies that $\inf_{m\ge 0} F(\mu_m) = -\infty$. □

Proposition: For the DCSLA, we have $\inf_{m\ge 0} F(\mu_m) = F^*$.

Proof: If $\lim_{l\to\infty} \delta_l > 0$, according to Lemma 6, it is not difficult to know that $\inf_{m\ge 0} F(\mu_m) = -\infty$ and it is proved. Thus, assume that $\lim_{l\to\infty} \delta_l = 0$. Let L be given by $L = \{l \in \{1, 2, \ldots\} | \delta_{l+1} = \beta\delta_l\}$. Then, from Steps 4, 5, and 7, we obtain

$$\sigma_{m+1} = \sigma_m + \|\alpha_m \hat{d}_m\| = \sum_{i=M[l]}^{m} \|\alpha_i \hat{d}_i\|$$

so that $M[l+1] = m+1$ and $l+1 \in L$ whenever $\sum_{i=M[l]}^{m+1} \alpha_i \|\hat{d}_i\| > \sigma_{\max} h(l)$ at Step 6. Hence

$$\sum_{i=M[l]}^{M[l+1]-1} \alpha_i > h(l)\sigma_{\max}/G.$$

Because the cardinality of L is infinite and $\sum_{l\in L} h(l) = \sum_{s=1}^{\infty} 1/(s+1)$, we have

$$\sum_{i=M[\hat{l}]}^{\infty} \alpha_i \ge \sum_{l\ge\hat{l}, l\in L} \sum_{i=M[\hat{l}]}^{M[l+1]-1} \alpha_i > \sum_{s=\hat{s}}^{\infty} \frac{\sigma_{\max}}{G(s+1)} \to \infty.$$

Now, to arrive at a contradiction, assume that $\inf_{m\ge 0} F(\mu_m) > F^*$, so that for some $\hat{\mu} \in \Phi$ and some $\varepsilon > 0$, $\inf_{m\ge 0} F(\mu_m) - \varepsilon \ge F(\hat{\mu})$. Because $F(\mu)$ is continuous over Φ and $\delta_l \to 0$, there is a sufficiently large $\hat{l}$ such that $\delta_{\hat{l}} \le \varepsilon$ for all $l \ge \hat{l}$, so that for all $m \ge M[\hat{l}], F_{\text{lev}}^m = F_{\text{rec}}^{M[l]} - \delta_l \ge \inf_{m\ge 0} F(\mu_m) - \varepsilon \ge F(\hat{\mu})$. Using this relation and Lemma 4, for $\mu = \hat{\mu}$ and $\alpha_m = t[F(\mu_m) - F_{\text{lev}}^m]/\|\hat{d}_m\|^2$, we obtain

$$\begin{aligned}
&\|\mu_{m+1} - \hat{\mu}\|^2 \\
&\quad \le \|\mu_m - \hat{\mu}\|^2 - 2\alpha_m [F(\mu_m) - F(\hat{\mu})] + \alpha_m^2 \|\hat{d}_m\|^2 \\
&\quad \le \|\mu_m - \hat{\mu}\|^2 - 2\alpha_m [F(\mu_m) - F_{\text{lev}}^m] + \alpha_m^2 \|\hat{d}_m\|^2 \\
&\quad \le \|\mu_m - \hat{\mu}\|^2 - t(2-t)[F(\mu_m) - F_{\text{lev}}^m]^2/\|\hat{d}_m\|^2. \\
&\quad \le \|\mu_m - \hat{\mu}\|^2 - (2-t)\alpha_m^2 \|\hat{d}_m\|^2/t.
\end{aligned}$$

By summing these inequalities over $m \ge M[\hat{l}]$, we have

$$(2-t)/t \sum_{m=M[\hat{l}]}^{\infty} \alpha_m^2 \|\hat{d}_m\|^2 \le \|\mu_{M[\hat{l}]} - \mu\|^2. \quad (47)$$

If $\|\hat{d}_m\|^2 \to 0 (m \to \infty)$, according to the definition of $\hat{d}_m$, we know that $F(\mu_m) \to F^* (m \to \infty)$ because $F(\mu)$ is a convex function; Otherwise, $\sum_{m=0}^{\infty} \alpha_m^2 < \infty$ according to (47). Because $\sum_{m=0}^{\infty} \alpha_m = \infty$ and $\alpha_m > 0$, according to Lemma 5, the following relation can be obtained: $\liminf_{m\to\infty} F(\mu_m) = F^*$.

Therefore, $\inf_{m\ge 0} F(\mu_m) = F^*$, which contradicts the assumption.

REFERENCES

[1] H. Missbauer, W. Hauberb, and W. Stadler, "A scheduling system for the steelmaking-continuous casting process: A case study from the steelmaking industry," *Int. J. Prod. Res.*, vol. 47, pp. 4147–4172, 2009.

[2] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmon, MA, USA: Athena, 1999.

[3] P. B. Luh, D. J. Hoitomt, E. Max, and K. R. Pattipati, "Scheduling generation and reconfiguration for parallel machines," *IEEE Trans. Robot. Autom.*, vol. 6, pp. 687–696, Dec. 1990.

[4] P. B. Luh and D. J. Hoitomt, "Scheduling of manufacturing systems using the Lagrangian relaxation technique," *IEEE Trans. Autom. Control*, vol. 38, pp. 1066–1079, Jul. 1993.

[5] T. Nishi, Y. Isoya, and M. Inuiguchi, "An integrated column generation and LAgrangian relaxation for flowshop scheduling problems," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, 2009, pp. 299–304.

[6] H. Chen, C. Chu, and J. M. Proth, "An improvement of the Lagrangian relaxation approach for job shop scheduling: A dynamic programming method," *IEEE Trans. Robot. Autom.*, vol. 14, pp. 786–795, Oct. 1998.

[7] P. Baptiste, M. Flamini, and F. Sourd, "Lagrangian bounds for just-int-ime job-shop scheduling," *Comput. Oper. Res.*, vol. 35, pp. 905–915, 2008.

[8] L. Tang, H. Xuan, and J. Liu, "A new Lagrangian relaxation algorithm for hybrid flowshop scheduling to minimize total weighted completion time," *Comput. Oper. Res.*, vol. 33, pp. 3344–3359, 2006.

[9] T. Nishi, Y. Hiranaka, and M. Inuiguchi, "Lagrangian relaxation with cut generation for hybrid flowshop scheduling problems to minimize the total weighted tardiness," *Comput. Oper. Res.*, vol. 37, pp. 189–198, 2010.

[10] L. X. Tang, P. B. Luh, J. Y. Liu, and L. Fang, "Steelmaking process scheduling using Lagrangian relaxation," *Int. J. Prod. Res.*, vol. 40, pp. 55–70, 2002.

[11] H. Xuan and L. Tang, "Scheduling a hybrid flowshop with batch production at the last stage," *Comput. Oper. Res.*, vol. 34, pp. 2178–2733, 2007.

[12] M. Held and R. M. Karp, "The travelling salesman problem and minimization spanning tress: Part II," *Math. Program.*, vol. 1, pp. 6–25, 1971.

[13] B. Guta, "Subgradient optimization methods in integer programming with an application to a radiation therapy problem," Ph.D. dissertation, Teknishe Universitat Kaiserlautern, Kaiserlautern, Germany, 2003.

[14] S. Kim, H. Ahn, and S. C. Cho, "Variable target value subgradient method," *Math. Program.*, vol. 49, pp. 359–369, 1991.

[15] U. Brannlund, "On relaxation methods for nonsmooth convex optimization," Ph.D. dissertation, Royal Inst. Technol., Stockholm, Sweden, 1993.

[16] J. L. Goffin and K. C. Kiwiel, "Convergence of a simple subgradient level method," *Math. Program.*, vol. 85, pp. 207–211, 1999.

[17] H. D. Sherali, G. Choi, and C. H. Tuncbilek, "A variable target value method for nondifferentiable optimization," *Oper. Res. Lett.*, vol. 26, pp. 1–8, 2000.

[18] P. M. Camerini, L. Fratta, and F. Maffioli, "On improving relaxation methods by modified gradient techniques," *Math. Program. Study*, vol. 3, pp. 26–34, 1975.

[19] H. D. Sherali and O. Ulular, "A primal-dual conjugate subgradient algorithm for specially strucutured linear and convex programming problems," *Appl. Math. Optim.*, vol. 20, pp. 193–221, 1989.

[20] T. Larsson, M. Patriksson, and A. B. Stromberg, "Conditional subgradient optimization: Theory and applications," *Eur. J. Oper. Res.*, vol. 88, pp. 382–403, 1996.

[21] X. Zhao, P. B. Luh, and J. Wang, "Surrogate gradient algorithm for Lagrangian relaxation," *J. Optim. Theory Appl.*, vol. 100, pp. 699–712, 1999.

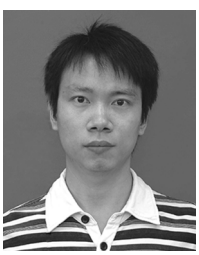
[22] T. S. Chang, "Comments on surrogate gradient algorithm for Lagrangian relaxation," *J. Optim. Theory Appl.*, vol. 137, pp. 691–697, 2008.

[23] R. Buil, M. A. Piera, and P. B. Luh, "Improvement of Lagrangian relaxation convergence for production scheduling," *IEEE Trans. Autom. Sci. Eng.*, vol. 9, no. 1, pp. 137–147, Jan. 2012.

[24] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, pp. 117–129, 1976.

[25] L. Tang, J. Liu, A. Rong, and Z. Yang, "A review of planning and scheduling systems and methods for integrated steel production," *Eur. J. Oper. Res.*, vol. 133, pp. 1–20, 2001.

[26] G. Dutta and R. Fourer, "A survey of mathematical programming applications in integrated steel plants," *Manuf. Service Oper. Manage.*, vol. 3, pp. 387–400, 2001.
[27] Q.-K. Pan, L. Wang, K. Mao, J.-H. Zhao, and M. Zhang, "An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process," *IEEE Trans. Autom. Sci. Eng.*, vol. 10, no. 2, pp. 307–322, Apr. 2013.
[28] V. Kumar, S. Kumar, F. T. S. Chan, and M. K. Tiwari, "Auction-based approach to resolve the scheduling problem in the steelmaking process," *Int. J. Prod. Res.*, vol. 44, pp. 1503–1522, 2006.
[29] J. Li, X. Xiao, Q. Tang, and C. A. Floudas, "Production scheduling of a large-scale steelmaking continuous casting process via unit specific event-based continuous-time models: Short-term and medium term scheduling," *Ind. Eng. Chem. Res.*, vol. 51, pp. 7300–7319, 2012.
[30] A. Atighehchian, M. Bijari, and H. Tarkesh, "A novel hybrid algorithm for scheduling steelmaking continuous casting production," *Comput. Oper. Res.*, vol. 36, pp. 2450–2461, 2009.
[31] J. P. Sousa and L. A. Wolsey, "A time indexed formulation of nonpreemptive single machine scheduling problems," *Math. Program.*, vol. 54, pp. 353–367, 1992.
[32] A. B. Keha, K. Khowala, and J. W. Fowler, "Mixed integer programming formulations for single machine scheduling problems," *Comput. Ind. Eng.*, vol. 56, pp. 357–367, 2009.
[33] H. Chen and P. B. Luh, "An alternative framework to Lagrangian relaxation approach for job shop scheduling," *Eur. J. Oper. Res.*, vol. 149, pp. 499–512, 2003.
[34] J. K. Lenstra, K. A. Rinnooy, and P. Bruker, "Complexity of machine scheduling problems," *Ann. Discrete Math.*, vol. 1, pp. 343–362, 1977.
[35] J. V. Demyanov and V. K. Somesova, "Conditional subdifferentials of convex functions," *Soviet Mathematics Doklady*, vol. 19, pp. 1181–1185, 1978.
[36] K. Mao, Q.-K. Pan, X. Pang, and T. Chai, "A novel Lagrangian relaxation approach for a hybrid flowshop scheduling problem in the steel making continuous casting process," *Eur. J. Oper. Res.*, vol. 236, pp. 51–60, 2014.
[37] J. E. Beasley, "Lagrangian heuristic for location problem," *Eur. J. Oper. Res.*, vol. 65, pp. 383–399, 1993.
[38] S. Tragantalerngsak, J. Holt, and M. Bonnqvist, "Lagrangian heuristics for the two-echelon, single source, capacitated facility location problem," *Eur. J. Oper. Res.*, vol. 102, pp. 611–625, 1997.
[39] S.-H. Wang, "An improved stepsize of the subgradient algorithm for solving the lagrangian relaxation problem," *Comput. Electr. Engineering*, vol. 29, pp. 245–249, 2003.
[40] S. Boyd and A. Mutapcic, "Subgradient methods," in *Lecture Notes of EE364b*. Stanford, CA, USA: Stanford Univ., 2006–2007, Winter.



Kun Mao received the B.S. and M.S. degrees from Northeastern University, Shenyang, China, in 2008 and 2010, respectively. Since 2010, he has been working toward the Ph.D. degree in control theory and control engineering at Northeastern University.

His current research interests include discrete optimization and scheduling.

Quan-Ke Pan received the B.Sc. and Ph.D. degrees from Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1993 and 2003, respectively.

Since 2003, he has been with the Computer Science Department, Liaocheng University, where he became a Full Professor in 2006. He has been with the State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang, China, since 2011. His current research interests include intelligent optimization and scheduling. He has authored more than 200 refereed papers.

Tianyou Chai (M'90–SM'97–F'08) received the Ph.D. degree in control theory and engineering from Northeastern University, Shenyang, China, in 1985.

Since 1988, he has been a Professor with Northeastern University, where he is the Founder and the Director of the Research Center of Automation, which became a National Engineering and Technology Research Center and a State Key Laboratory. He has developed control technologies with applications to various industrial processes. He has authored/coauthored 144 peer-reviewed international journal papers. His current research interests include modeling, control, optimization, and integrated automation of complex industrial processes. M

Dr. Chai is a member of the Chinese Academy of Engineering, a Fellow of the International Federation of Automatic Control, and the Director of the Department of Information Science of the National Natural Science Foundation of China. For his contributions, he has been a recipient of four prestigious awards of the National Science and Technology Progress and National Technological Innovation and the 2007 Industry Award for Excellence in Transitional Control Research from the IEEE Multi-Conference on Systems and Control.

Peter B. Luh (S'76–M'80–SM'91–F'95) received the B.S. degree from National Taiwan University, Taipei, Taiwan, the M.S. degree fromthe Massachusetts Institute of Technology, Cambridge, MA, USA, and the Ph.D. from Harvard University, Cambridge.

He is the SNET Professor of Communications and Information Technologies with the Department of Electrical and Computer Engineering, University of Connecticut, Storrs, CT, USA. He is a member of the Chair Professors Group at CFINS in the Department of Automation, Tsinghua University, Beijing, China. His research interests include smart power systems and auction methods, smart, green and safe buildings, and intelligent manufacturing systems.

Prof. Luh was the founding Editor-in-Chief of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, and an Editor-in-Chief of the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION.