European Journal of Operational Research 236 (2014) 51–60

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

A novel Lagrangian relaxation approach for a hybrid flowshop scheduling problem in the steelmaking-continuous casting process

Kun Mao, Quan-ke Pan *, Xinfu Pang, Tianyou Chai

State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, People's Republic of China

ARTICLE INFO

Article history:
Received 14 June 2012
Accepted 11 November 2013
Available online 21 November 2013

Keywords:
Scheduling
Hybrid flowshop
Subgradient optimization
Steelmaking-continuous casting

ABSTRACT

One of the largest bottlenecks in iron and steel production is the steelmaking-continuous casting (SCC) process, which consists of steel-making, refining and continuous casting. The SCC scheduling is a complex hybrid flowshop (HFS) scheduling problem with the following features: job grouping and precedence constraints, no idle time within the same group of jobs and setup time constraints on the casters. This paper first models the scheduling problem as a mixed-integer programming (MIP) problem with the objective of minimizing the total weighted earliness/tardiness penalties and job waiting. Next, a Lagrangian relaxation (LR) approach relaxing the machine capacity constraints is presented to solve the MIP problem, which decomposes the relaxed problem into two tractable subproblems by separating the continuous variables from the integer ones. Additionally, two methods, *i.e.*, the boundedness detection method and time horizon method, are explored to handle the unboundedness of the decomposed subproblems in iterations. Furthermore, an improved subgradient level algorithm with global convergence is developed to solve the Lagrangian dual (LD) problem. The computational results and comparisons demonstrate that the proposed LR approach outperforms the conventional LR approaches in terms of solution quality, with a significantly shorter running time being observed.

Published by Elsevier B.V.

1. Introduction

The hybrid flowshop (HFS) scheduling problem is a complex combinatorial problem encountered in many real-world applications Ruiz, Serifoglu, and Urlings (2008), Ribas, Leisten, and Framinan (2010). A wide variety of solution approaches, such as exact methods Gicquel, Hege, Minoux, and van Canneyt (2012), approximation methods Zhang and van de Velde (2012), heuristic algorithms Oguz, Ercan, Cheng, and Fung (2003) and meta-heuristic algorithms Ruiz and Maroto (2006), have been applied to HFS. An overview of research on HFS scheduling problems, including that investigating processing complexity, scheduling criteria and scheduling approaches, has been provided in a recent review Ruiz and Vazquez-Rodriguez (2010). This paper considers a realistic case of HFS arising from the SCC process, which exhibits the following features: job grouping and precedence constraints, no idle time within the same group of jobs and setup time constraints at the last stage.

Because most HFS scheduling problems are NP-hard and large-scale in practical applications, it is extremely difficult to obtain a high-quality schedule within an acceptable computation time Ruiz and Vazquez-Rodriguez (2010). Lagrangian relaxation (LR) is a decomposition and coordination approach that can yield near-optimal schedules in a reasonable computational time with a quantifiable accuracy. LR has emerged as a practical approach to complex scheduling problems, including single-machine Tang, Xuan, and Liu (2007), parallel-machine Luh, Hoitomt, Max, and Pattipati (1990), practical jobshop Hoitomt, Luh, and Pattipati (1993), Chen, Chu, and Proth (1998), Chen and Luh (2003) and HFS scheduling Tang, Luh, Liu, and Fang (2002, 2006), Xuan and Tang. (2007), Nishi, Hiranaka, and Inuiguchi (2010). In addressing these problems in previous studies, machine capacity constraints or operation precedence constraints were relaxed with Lagrangian multipliers. The relaxed problems were decomposed into a set of subproblems that could be solved by dynamic programming. The Lagrangian dual (LD) problems were often solved by subgradient optimization methods. The multipliers were iteratively adjusted at the "high level" based on the subgradients corresponding to the degrees of constraint violation. At the end of each iteration, simple heuristics were applied to obtain a feasible schedule.

The conventional subgradient algorithm (CSA) is easy to implement and usually provides rapid improvement in early iterations. However, the CSA yields guaranteed convergence to the optimum of the dual problem only if the optimal dual function value is known a priori Bertsekas (1999). In the LR approach, the CSA usually sets a maximal iteration number as the termination criterion, which is selected experientially. To overcome this shortcoming,

* Corresponding author. Tel.: +86 24 83684490.
E-mail address: panquanke@mail.neu.edu.cn (Q.-K. Pan).

0377-2217/$ - see front matter Published by Elsevier B.V.
http://dx.doi.org/10.1016/j.ejor.2013.11.010

Goffin and Kiwiel (1999) proposed a simple subgradient level algorithm (SSLA) by applying a Brannlund's level control strategy to estimate the optimum value Brannlund (1993). On the other hand, the CSAs for addressing LR problems only consider the nonnegativity constraints of the multipliers for primal problems with inequality constraints. However, even though all multipliers satisfy these constraints, some of them might result in unboundedness during the iterations (see Section 5.2.1). Therefore, it is necessary to consider not only the nonnegative constraints of multipliers but also the effects of multipliers on the objective function.

This paper considers the relaxation of the machine capacity constraints for the SCC scheduling problem and develops the SSLA to solve the LD problem. Compared with SSLA and CSA, the proposed algorithm, named the improved subgradient level algorithm (ISLA), has the following features: (1) the Brannlund's level control strategy is embedded in the proposed algorithm to estimate the optimal value; (2) three fine-adjusting strategies are introduced to improve the efficiency of SSLA; (3) two efficient approaches considering the sufficient and necessary bounded condition of LR problem are presented to address the unboundedness during iteration; and (4) numerical results show that these strategies improve the efficiency of the algorithm dramatically.

The rest of the paper is organized as follows. First, we provide a description of the SCC process, along with the characterization of SCC scheduling in Section 2. Section 3 gives a brief review of the methods for planning and scheduling in SCC production and LR approaches for the HFS scheduling problem. The SCC scheduling is formulated as an MIP in Section 4. In Section 5, the relaxation approach and the ISLA are presented. Computational experiments are described and comparisons made in Section 6. Finally, Section 7 concludes this study and discusses our future work.

2. The production process of steelmaking-continuous casting

The iron and steel industry provides raw materials for a number of other important industries, which is the most basic industry in the world economy. The manufacturing process is a multi-stage process that can be roughly divided into three phases: ironmaking, steelmaking-continuous casting and steel rolling Kumar, Kumar, Chan, and Tiwari (2006). This paper studies the steelmaking-continuous casting (SCC) phase, which is often a bottleneck in the manufacturing process. The principal processes for primary SCC consist of three stages: steelmaking, refining and continuous casting as shown in Fig. 1.

In the first step, the steelmaking stage (LD process), hot molten iron arrives from a blast furnace in insulated vessels, often by rail, and is poured into converter furnaces or basic oxygen furnaces along with scrap steel. The produced molten iron in the same converter is called a charge (job), a basic unit of SCC production. In the second step, the refining stage (RH process and LF process), molten steel is poured into a ladle, which is transported by a crane to a refining furnace for refining. The operations at this stage aim to produce molten steel of the correct grade or chemistry. At the refining stage, a charge undergo several refining processes (such as the RH process and the LF process). Finally, graded liquid steel from the refining stage is poured into a ladle and transferred to a continuous caster, which can process multiple charges consecutively in the form of a cast (batch). The production process is detailed in Kumar et al. (2006), Tang et al. (2002). The salient features of the SCC process are summarized as follows.

(1) All jobs (charges) follow the same process route and cannot skip stages. A job can be processed at most on one machine and a machine can process at most one job at a time. Job processing is nonpreemptive.
(2) Each machine (caster) of the last stage must process multiple jobs consecutively as a cast (batch). The operation sequence of the jobs in a cast is predefined. The processing machine of a cast at the last stage is known.
(3) A sequence-independent setup time is incurred between two adjacent casts on the same machine. Transportation time is separated from the processing time.

Because all jobs visit three or more stages in the same direction and each stage has multiple parallel identical machines, the SCC process can be viewed as a complex HFS problem with the following features: job grouping and precedence constraints, and a setup time constraint on the machines used in the last stage. The objective of the scheduling problem is to minimize the total weighted earliness/tardiness and the total waiting-time between two operations. Regarding complexity, the single-machine problem with the objective of minimizing the sum of the weighted deviations of the job completion times from a given common due date is already NP-hard Hoogeveen and de Velde (1991). Because this problem is a special case of the SCC scheduling problem, we can conclude that the SCC scheduling problem is also NP-hard.

3. Literature review

Solution methods for solving SCC scheduling problems can be broadly classified into three categories: mathematical programming, heuristics, and artificial intelligence. A previous comprehensive survey on various production planning and scheduling approaches for the steelmaking production have been presented in Tang, Liu, Rong, and Yang (2001). In this paper, we focus on recent advancements and review the relevant literature on SCC scheduling problems.

Regarding mathematical programming methods, Tang et al. (2002, 2007) considered an m-stage HFS problem, which was modeled as an integer programming problem. They integrated the LR and forward dynamic programming to solve the problem. Bellabdaoui and Teghem (2006) provided an MIP model for a three-stage HFS problem with two parallel machines at each stage, and solved it using standard software packages. Recently, Missbauer, Hauberb, and Stadler (2009) applied linear programming and heuristic methods to address a complex case of SCC in an Austrian steel plant where there were four stages and each stage had three parallel machines at most.

With respect to heuristic methods, Pacciarelli and Pranzo (2004) adopted an alternative graph formulation to describe an m-stage HFS problem and solved the problem by a beam search procedure. Kumar et al. (2006) employed a combinatorial auction-based approach to a three-stage HFS problem with four parallel machines at each stage.

With respect to artificial intelligence methods, Atighehchian, Bijari, and Tarkesh (2009) combined ant colony optimization and nonlinear optimization methods to solve a three-stage HFS problem. Similarly, Pan, Wang, Mao, Zhao, and Zhang (2013) proposed an effective artificial bee colony algorithm for a three-stage HFS problem in the steelmaking process.

In recent decades, the LR approach has emerged as a practical approach for HFS scheduling. Different relaxations or decoupling strategies have been proposed to solve these scheduling problems. Tang et al. (2002) relaxed the machine capacity constraints of the HFS problem and used dynamic programming to solve the relaxed problem. Later, the same authors developed an operation-precedence constraints relaxation method for HFS to minimize the total weighted completion time Tang et al. (2006). Similarly, Xuan and Tang. (2007) proposed a batch decoupling approach for HFS with the objective of minimizing the completion time. Recently, Nishi

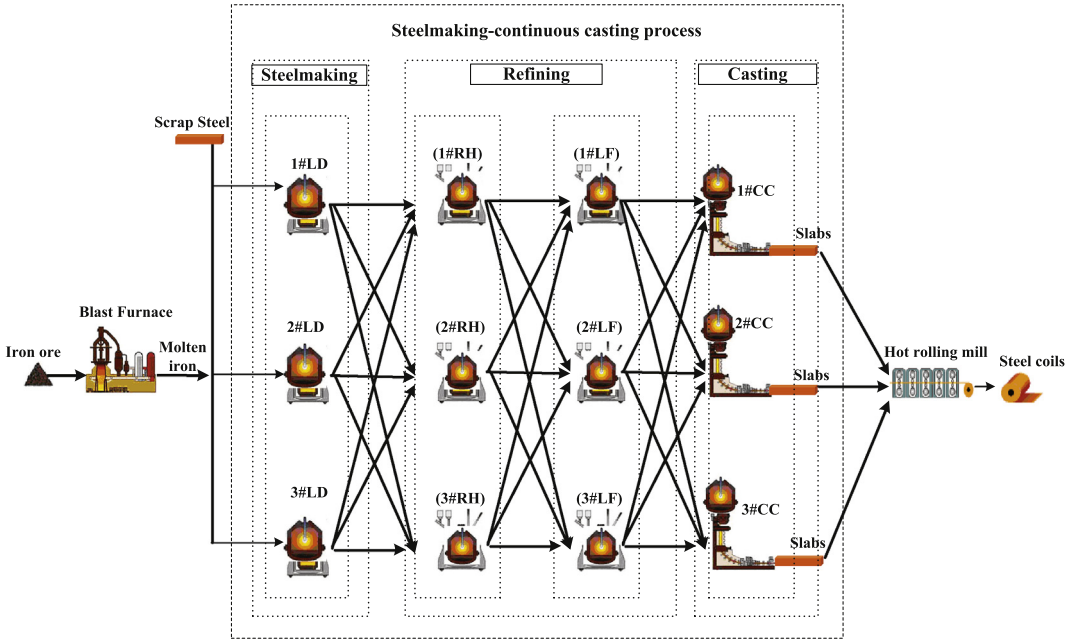

Fig. 1. Steelmaking-continuous casting process.

et al. (2010) proposed an LR approach with cut generation to improve the Lagrangian bounds for the LR problem by relaxing machine capacity constraints. It is worth noting that all LD problems addressed in the aforementioned approaches were solved by the CSA.

4. Mathematical formulation

4.1. Notation

The following notation describes indices, sets, and number of elements, fixed parameters and variables.

Indices, sets, and number of elements

i: charge index (also called job index);
j: stage index;
n: cast index (also called batch index);
M_j: number of machines available in 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, $n \in \{1,2,\ldots,N\}$, where N is the total number of batches;
$\Omega_{n_1} \cap \Omega_{n_2} = \emptyset$, for any $n_1 \neq n_2 \in \{1,2,\ldots,N\}$; $\Omega_1 \cup \Omega_2 \cup \cdots \cup \Omega_n = \Omega$ (Ω_n is a given set);
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, $s(n) = s(n-1) + |\Omega_n|$, $s(0) = 0$, $s(N) = |\Omega|$, $\Omega_n = \{s(n-1)+1,\ldots, s(n)\}$;
$b(k)$: index of the last batch on the kth machine at the last stage, $b(k) = b(k-1) + |B_k|$, $b(0) = 0$, $b(M_S) = N$, where S is the total number of stages. Then, $B_k = \{b(k-1)+1,\ldots,b(k)\}$.

Data or fixed parameters

P_{ij}: processing time of job i in stage j;
T_{jj+1}: transportation time from stage j to stage $j+1$;
d_n: planning time of batch n starting;
Su: setup time between two adjacent batches on the same machine in the last stage;
C_j: penalty coefficient for the waiting time of each job between stage j and stage $j+1$;
D_1: penalty coefficient for each batch starting before due date;
D_2: penalty coefficient for each batch starting after due date;
U: sufficient large positive integer.

Decision variables

x_{ijk}: 0/1 variable that is equal to one if and only if job i is processed on the kth machine in stage j;
$y_{i_1,i_2,j}$: 0/1 variable that is equal to one if and only if charge i_1 is processed before charge i_2 in stage j;
t_{ij} starting time of job i in stage j.

4.2. The model

With the aforementioned symbols, the complex HFS scheduling problem is formulated as follows.

4.2.1. Objective function

To ensure continuity of the production process and just-in-time delivery of final products, this paper aims to minimize a cost function consisting of the following terms.

(1) Job waiting penalty between adjacent operations to reduce the temperature drop for molten steel as much as possible.
(2) Earliness/tardiness penalty to ensure that blooms or billets in each charge are delivered as punctually as possible.

The objective function is formulated as follows.

$$(\mathrm{P}) \min G = \min (F_1 + F_2 + F_3), \tag{1}$$

where

$$F_1 = \sum_{i=1}^{|\Omega|} \sum_{j=1}^{S-1} C_j (t_{ij+1} - t_{ij}), \tag{2}$$

$$F_2 = D_1 \sum_{n=1}^{N} \max (0, d_n - t_{s(n-1)+1,S}), \tag{3}$$

$$F_3 = D_2 \sum_{n=1}^{N} \max (0, t_{s(n-1)+1,S} - d_n), \tag{4}$$

F_1, F_2 and F_3 are the penalty functions of job waiting between adjacent operations, batch earliness, and batch tardiness, respectively.

It can be easily observed that the nonlinearity of the model (P) derives from F_2 and F_3. To address this nonlinearity, two sets of new variables are introduced: $t^l_{s(n-1)+1,S} = \max(0, d_n - t_{s(n-1)+1,S})$, $t^u_{s(n-1)+1,S} = \max(0, t_{s(n-1)+1,S} - d_n)$. Therefore $t_{s(n-1)+1,S} = t^u_{s(n-1)+1,S} - t^l_{s(n-1)+1,S} + d_n \quad (\forall k \in \{1,2,\ldots,M_S\},\ \forall n \in \{1,2,\ldots,|B_k|-1\})$ and $F_2 = D_1 \sum_{n=1}^{N} t^l_{s(n-1)+1,S}$, $F_3 = D_2 \sum_{n=1}^{N} t^u_{s(n-1)+1,S}$.

4.2.2. Constraints about continuous variables

(1) For the two consecutive operations of the same job, only when the preceding operation has terminated can the immediate next one be started. We call this constraint operation precedence constraint.

$$t_{i,j+1} - t_{ij} - P_{ij} \geqslant T_{jj+1}, \quad \forall i \in \Omega,\ \forall j \in \{1,2,\ldots,S-1\}, \tag{5}$$

(2) The adjacent jobs in the same batch must be processed consecutively without any waiting time in the last stage.

$$t_{i+1,S} = t_{i,S} + P_{i,S}, \quad \forall i, i+1 \in \Omega_n,\ \forall n \in \{1,2,\ldots,N\}. \tag{6}$$

(3) There exists a setup time between two adjacent batches for changing equipment on the same machine in the last stage.

$$t_{i+1,S} - t_{i,S} - P_{i,S} \geqslant Su, \tag{7}$$

where $i = s(b(k-1)+n),\ \forall k \in \{1,2,\ldots,M_S\},\ \forall n \in \{1,2,\ldots,|B_k|-1\}$.

(4) The starting time must be nonnegative.

$$t_{ij} \geqslant 0, \quad \forall i \in \Omega,\ \forall j \in \{1,2,\ldots,S\}. \tag{8}$$

4.2.3. Constraints about discrete variables

(1) A job can be processed on exactly one machine at each stage.

$$\sum_{k=1}^{M_j} x_{ijk} = 1, \quad \forall i \in \Omega,\ \forall j \in \{1,2,\ldots,S-1\}. \tag{9}$$

(2) There exists an operation precedence relationship between any two different jobs at each stage.

$$y_{i_1,i_2,j} + y_{i_2,i_1,j} = 1, \quad \forall i_1 \neq i_2, i_1, i_2 \in \Omega, \\ \forall j \in \{1,2,\ldots,S-1\}. \tag{10}$$

(3) The discrete variable must hold for the following constraints.

$$x_{ijk} \in \{0,1\},\ \forall i \in \Omega, \forall j \in \{1,2,\ldots,S-1\}, \forall k \in \{1,2,\ldots,M_j\}. \tag{11}$$

$$y_{i_1,i_2,j} \in \{0,1\},\ \forall i_1 \neq i_2, i_1, i_2 \in \Omega, \forall j \in \{1,2,\ldots,S-1\}. \tag{12}$$

4.2.4. Constraints regarding continuous and discrete variables

A machine can process at most one job at a time, which is called the machine capacity constraint.

$$t_{i_2,j} - t_{i_1,j} - P_{i_1,j} + (3 - x_{i_1,j,k} - x_{i_2,j,k} - y_{i_1,i_2,j})U \geqslant 0, \\ \forall i_1 \neq i_2,\ i_1, i_2 \in \Omega,\ \forall j \in \{1,2,\ldots,S-1\}, \\ \forall k \in \{1,2,\ldots,M_j\}. \tag{13}$$

4.2.5. Transformation constraints

To transform the objective function into a linear function, some extra constraints have to be added.

$$t_{s(n-1)+1,S} = t^u_{s(n-1)+1,S} - t^l_{s(n-1)+1,S} + d_n, \quad \forall n \in \{1,2,\ldots,N\}, \tag{14}$$

$$t^u_{s(n-1)+1,S} \geqslant 0, \quad t^l_{s(n-1)+1,S} \geqslant 0, \quad \forall n \in \{1,2,\ldots,N\}. \tag{15}$$

5. Solution methodology

5.1. Lagrangian relaxation

In the model formulated above, only machine capacity constraints (13) couple different types of variables or different jobs on the same machine; therefore, we consider relaxing the constraints (13) to decompose the LR problem into job-level subproblems or two tractable subproblems for each type of variable. There exist other relaxation strategies such as the relaxation of the operation precedence constraints. However, for the considered problem, after relaxing the operation precedence constraints, the subproblems of the corresponding LR problem are also NP-hard Tang et al. (2006). For example, if we relax the operation precedence constraints (5), the LR problem can be decomposed into $S-1$ parallel machine scheduling problems, each for a different stage. The subproblem is to minimize total weighted completion time on parallel identical machines, which is NP-hard for $M_j \geqslant 2$ Lenstra, Rinnoy, and Brucker (1977).

By relaxing the constraints (13) with nonnegative Lagrangian multipliers $\{\mu_{i_1,i_2,j,k}\}$, we can obtain the following LR problem

$$(\mathrm{LR}) L(\mu) = \min G_L = \min(F_1 + F_2 + F_3 + F_4),$$

where F_1, F_2 and F_3 are as defined previously and

$$F_4 = -\sum_{j=1}^{S-1}\sum_{k=1}^{M_j}\sum_{i_1=1}^{|\Omega|}\sum_{i_2=1,i_2\neq i_1}^{|\Omega|} \mu_{i_1,i_2,j,k}(t_{i_2,j} - t_{i_1,j} - P_{i_1,j} \\ + (3 - x_{i_1,j,k} - x_{i_2,j,k} - y_{i_1,i_2,j})U), \tag{16}$$

subject to (5)–(12), (14) and (15) and

$$\mu_{i_1,i_2,j,k} \geqslant 0, \quad j \in \{1,2,\ldots,S-1\}, \quad k \in \{1,2,\ldots,M_j\}, \\ i_1 \neq i_2 \in \Omega. \tag{17}$$

Here, μ is a vector of nonnegative Lagrangian multipliers $\{\mu_{i_1,i_2,j,k}\}$ associated with the constraints in (13). $L(\mu)$, as a function of the multipliers, is the LR problem. The corresponding LD problem is

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

subject to (5)–(12), 14,15 and 17. With multipliers $\{\mu_{i_1,i_2,j,k}\}$, the LR problem can be decomposed into the following two subproblems, *i.e.*, a linear programming problem (LR_1) and a 0-1 integer programming problem (LR_2).

$$(\mathrm{LR}_1) \quad \min\ G_{L1} = \min\,(F_1 + F_2 + F_3 + F_{41}),$$

subject to (5)–(8) (14) and (15), where

$$F_{41} = \sum_{j=1}^{S-1}\sum_{k=1}^{M_j}\sum_{i_1=1}^{|\Omega|}\sum_{i_2=1,i_1\neq i_2}^{|\Omega|} \left(t_{i_1,j}\left(\mu_{i_1,i_2,j,k} - \mu_{i_2,i_1,j,k}\right) + \mu_{i_1,i_2,j,k}P_{i_1,j}\right). \tag{18}$$

$$(\mathrm{LR}_2) \quad \min\ G_{L2} = \min F_{42},$$

subject to (9)–(12), where

$$F_{42} = -\sum_{j=1}^{S-1}\sum_{k=1}^{M_j}\sum_{i_1=1}^{|\Omega|}\sum_{i_1=1,i_1\neq i_2}^{|\Omega|} (3 - x_{i_1,j,k} - x_{i_2,j,k} - y_{i_1,i_2,j})\mu_{i_1,i_2,j,k}U. \tag{19}$$

5.2. Subproblem solution

5.2.1. Solution of subproblem (LR_1)

From the definition of $F_1,\ F_2,\ F_3$ and F_{41}, it can be determined that

$$G_{L1} = \sum_{j=1}^{S}\sum_{i=1}^{|\Omega|} c_{ij}t_{ij} + \sum_{n=1}^{N}\left(D_1 t^l_{s(n-1)+1,S} + D_2 t^u_{s(n-1)+1,S}\right) + \bar{P}, \tag{20}$$

where $\bar{P} = \sum_{j=1}^{S-1}\sum_{k=1}^{M_j}\sum_{i_1=1}^{|\Omega|}\sum_{i_2=1,i_2\neq i_1}^{|\Omega|}\mu_{i_1,i_2,j,k}P_{i_1,j}$,

$$c_{ij} = \begin{cases} \sum_{k=1}^{M_j}\sum_{i_2=1,i_2\neq i}^{|\Omega|}\left(\mu_{i,i_2,j,k} - \mu_{i_2,i,j,k}\right) - C_j, & i\in\Omega,\ j=1; \\ \sum_{k=1}^{M_j}\sum_{i_2=1,i_2\neq i}^{|\Omega|}\left(\mu_{i,i_2,j,k} - \mu_{i_2,i,j,k}\right) + C_j - C_{j+1}, & i\in\Omega,\ 1<j<S; \\ C_j, & i\in\Omega,\ j=S. \end{cases}$$

The subproblem (LR_1) is a typical linear programming model which can be solved by a standard software package directly such as CPLEX. It is worth noting that the coefficients of the objective function G_L may sometimes be negative in the iterations of subgradient algorithms. Moreover, the negativity of some coefficients might result in the unboundedness of the linear programming problem (LR_1) in some cases. The following proposition provides the bounded cases.

Proposition 1. *For given multipliers* $\{\mu_{i_1,i_2,j,k}\}$, *the sufficient and necessary condition of the boundedness of linear programming* (LR_1) *is* c_{ij} ($i\in\Omega, 1\leqslant j\leqslant S$), *which satisfies following conditions*

$$\sum_{j=s}^{S}\sum_{i\in\Omega}c_{ij} + (b(k)-n)D_2 \geqslant 0, \tag{21}$$

where $\bar{\Omega} = \Omega_n\cup\Omega_{n+1}\cup\cdots\cup\Omega_{b(k)}, b(k-1)+1\leqslant n\leqslant b(k), k=1,2,\ldots, M_S, 1\leqslant s\leqslant S$.

Proof. Because the machine capacity constraints are relaxed, it is not difficult to obtain the proposition based on the operation precedence constraints of each job. □

The boundedness of the linear programming problem (LR_1) can also be checked via the interior point method or simplex method. If the linear programming problem (LR_1) is unbounded, according to the Proposition 1, the boundedness of (LR_1) can be guaranteed by decreasing each multiplier of $\left\{\mu_{i_1,i_2,j,k}\right\}$ by a constant factor. A simple strategy can also be applied to replace the boundedness check, *e.g.*, we set a time horizon $UT = \sum_{i\in\Omega}(\sum_{j=1}^{S}p_{ij} + \sum_{j=1}^{S-1}T_{j,j+1})$ as the upper bound of the starting time of each job. This method is called the time horizon method. However, the two methods have different effects on subgradient algorithms, the details of which are discussed in Section 6.2.

5.2.2. Solution of subproblem (LR_2)

The subproblem (LR_2) contains integer and continuous variables and therefore is not easily solvable. Therefore, we further decompose the problem (LR_2) into tractable subproblems. In order to do so, we present the following proposition.

Proposition 2. *For given multipliers* $\{\mu_{i_1,i_2,j,k}\}$, *the optimal solution of subproblem* (LR_2) *is*

$$x^*_{ijk} = \begin{cases} 1, & k = \arg\min_{1\leqslant k\leqslant M_j}\bar{\mu}_{ijk} \\ 0, & otherwise \end{cases}, \quad i\in\Omega,\ j=\{1,2,\ldots,S-1\},$$

where $\bar{\mu}_{ijk} = \sum_{r=1,r\neq i}^{|\Omega|}\mu_{irjk} + \sum_{r=1,r\neq i}^{|\Omega|}\mu_{rijk}$, *and*

$$y^*_{i_1,i_2,j} = \begin{cases} 1, & \hat{\mu}_{i_1,j,k} < \hat{\mu}_{i_2,j,k} \\ 0, & otherwise \end{cases}, \quad i_1\neq i_2,\ i_1,i_2\in\Omega,\ j=\{1,2,\ldots,S-1\},$$

where $\hat{\mu}_{ijk} = \sum_{i_2=1,i_2\neq i}^{|\Omega|}\sum_{k=1}^{M_j}\mu_{i_1,i_2,j,k}$.

5.3. Construction of a feasible solution to the primal problem

In general, the solution of dual problem is associated with an infeasible schedule because the relaxed constraints may not be satisfied. The following heuristic, which similar to list scheduling, is used to generate a feasible schedule from the solution of the relaxed problem.

Step 1: Take the solution $\{t_{ij}\}$ of LR problem as the initial list. Set $x_{i,j,k} = 0 (i\in\Omega, 1\leqslant k\leqslant M_j, 1\leqslant j<S)$, $y_{i_1,i_2,j} = 0 (i_1\neq i_2, i_1,i_2\in\Omega, 1\leqslant j<S), j=1$.
Step 2: Let the list T_j be an ascending order list of all elements in $\{t_{ij}|i\in\Omega\}$. Set $n=1, J_k=\Phi, |J_k|=0,\ k=1,2,\ldots,M_j$.
Step 3: $T_j(n)$ denotes the nth element of T_j. Let $i=\arg_{i\in\Omega}\{t_{ij}=t_{T_j(n),j}\}$. Set $x_{i,j,k^*}=1$, where $k^*=\arg\min_{1\leqslant k\leqslant M_j}|J_k|$. If $|J_{k^*}|>0$, set $y_{i,r,j}=1, \forall r\in J_{k*}$.
Step 4: Set $J_{k^*}=J_{k^*}\cup\{i\}, |J_{k^*}|=|J_{k^*}|+1, n=n+1$. If $n\leqslant|\Omega|$, then go to step 3; otherwise, set $j=j+1$, and go to the next step.
Step 5: If $j<S$, then go to step 2; otherwise, go to the next step.
Step 6: Substitute $\{x_{i,j,k}\}$ and $\left\{y_{i_1,i_2,j}\right\}$ into the primal model (P), which becomes a linear programming model. Thus, a feasible solution can be obtained. Finally, terminate the algorithm.

5.4. Improved subgradient level algorithm

The LR function is concave (or convex) and nondifferentiable; such functions are commonly solved by bundle methods or subgradient methods. The proximal bundle method, rather than the classical bundle method, is often used to solve address nondiffierentiable problems Markela (2002). However, a quadratic programming problem must be addressed per iteration. The computational effort required for the quadratic programming increases very rapidly as the problem size increases. Hence, the bundle method is not applicable in our problem.

As mentioned in the introduction, CSA yields guaranteed convergence to the optimum value if the optimal dual function value is known a priori (which is obviously not the case in our scheduling problem). To overcome this deficiency, Goffin and Kiwiel (1999) proposed SSLA with Brannlund's level control strategy for estimating the optimum value Brannlund (1993). A more advanced level algorithm proposed by Brannlund, Kiwiel, and Lindberg (1995) is a branch of proximal bundle method, which has to solve a quadratic programming problem per iteration. To further enhance the efficiency of SSLA, this paper improves it by introducing three fine-adjusting strategies, collectively referred to as the improved subgradient level algorithm (ISLA).

Improved subgradient level algorithm:

Let $P(\mu)$ be a primal function value of μ, which is derived by using the heuristic presented in Section 5.3. Let $F(\mu) = -L(\mu), g_F(\mu)$ is the subgradient of $F(\mu)$, $g_F(\mu) = \left[v_{1,2,1,1}, v_{1,3,1,1}, \ldots, v_{|\Omega|,|\Omega-1|,S-1,M_{S-1}}\right]$, $v_{i_1,i_2,j,k} = t_{i_2,j} - t_{i_1,j} - P_{i_1,j} + \left(3 - x_{i_1,j,k} - x_{i_2,j,k} - y_{i_1,i_2,j}\right)U$.

Step 0: **Preparing for initialization.**
Select $\mu_1\geqslant 0, \alpha\in(0,1)$.
Step 1: **Boundedness detection of subproblem.**
Calculate $F(\mu_1)$. If the subproblem (LR_1) is not bounded, set $\mu_1=\alpha\mu_1$ and go to Step 1; otherwise, go to the next step.
Step 2: **Initialization.**
Select $\varepsilon_1>0,\ \varepsilon_2>0,\ \varepsilon_3>0,\ \beta\in(0,1),\ \delta_1>0$ (δ_1 denotes the reduction corresponding to the estimation level F^m_{lev}), $\sigma_{max}>0$ (σ_{max} denotes the threshold for adjusting δ_l), $W>0$ (W is an

even number), $t \in (0,2)$. Set $F^0_{\text{best}} = \infty$, $P_{\text{best}} = \infty$, the initial accumulated path $\sigma_1 = 0$, the iteration number $m = 1$, the weak-oscillation number $r = 0$, the strong-oscillation number $s = 0$, $l = 1$, $M[l] = 1$($M[l]$ will denote the iteration number when the lth update of F^m_{best} occurs). Denote $h(s) = 1/(s+1)$, which is used to adjust the threshold $\sigma_{\max}$.

Step 3: **Function evaluation.**

Step 3.1: Calculate $F(\mu_m)$ and $g_F(\mu_m)$. Denote $g_m = g_F(\mu_m)$. If $F(\mu_m) < F^{m-1}_{\text{best}}$, set $F^m_{\text{best}} = F(\mu_m)$; otherwise set $F^m_{\text{best}} = F^{m-1}_{\text{best}}$.

Step 3.2: Obtain a feasible solution and the corresponding objective function value $P(\mu_m)$ using the method described in Section 5.3. If $P(\mu_m) < P_{\text{best}}$, then set $P_{\text{best}} = P(\mu_m)$.

Step 3.3: If $P_{\text{best}} + F(\mu_m) < \delta_l$, then set $\delta_l = P_{\text{best}} + F(\mu_m)$.

Step 4: **Sufficient descent.**

If $F(\mu_k) \leqslant F^m_{\text{best}} - 0.5\delta_l$, then set $M[l+1] = m$, $\sigma_m = 0$, $\delta_{l+1} = \delta_l$, $l = l+1$.

Step 5: **Weak oscillation detection.**

Set $D[r] = F(\mu_m)$, $r = r+1$. If $r > W$ and $\sum_{n=1}^{W-2} \left|\frac{D[n+2]-D[n]}{D[n]}\right| \frac{2}{W} < \varepsilon_3$, set $m = m + \lfloor (h(s)\sigma_{\max} - \sigma_m + 1)/\|d_m\| \rfloor + 1$, $\sigma_m = h(s)\sigma_{\max} + 1$, where $\lfloor x \rfloor$ is the nearest integer less than or equal to x. If $r > W$, set $r = 0$.

Step 6: **Strong oscillation detection.**

If $\sigma_m > h(s)\sigma_{\max}$, then $M[l+1] = m$, $\sigma_m = 0$, $\delta_{l+1} = \beta\delta_l$, $l = l+1$, $s = s+1$.

Step 7: **Projection of multipliers.**

Let $F^m_{\text{lev}} = F^{M[l]}_{\text{best}} - \delta_l$. Set $z_m = \mu_m - d_m$, $\mu_{m+1} = \mathbf{P}_{R^+}(z_m)$, $d_m = t[F(\mu_m) - F^m_{\text{lev}}]g_m/\|g_m\|^2$.

Step 8: **Boundedness detection of subproblem.**

Calculate $F(\mu_{m+1})$. If the subproblem (LR_1) is not bounded, set $d_m = \alpha d_m$ and go to step 7; otherwise, go to the next step.

Step 9: **Path update.**

Set $\sigma_{m+1} = \sigma_m + \|d_m\|$, $m = m+1$.

Step 10: **Termination check.**

If $\|g_m\| < \varepsilon_1$ or $\left|\delta_l / F^{M[l]}_{\text{best}}\right| < \varepsilon_2$ is satisfied, then stop the iteration; otherwise, go to Step 3.

As mentioned in Section 5.2.1, one can also use the time horizon method to replace the boundedness detection (Steps 1 and 8) in the algorithm. To further improve the efficiency of the algorithm, some modifications of the projection of the multipliers (Step 7) suggested by Wang (2003) can be made as follows.

Step 7': **Improved projection of multipliers.**

$$d_m = t[F(\mu_m) - F^m_{\text{lev}}]\tilde{g}_m/\|\tilde{g}_m\|^2,$$

$$\tilde{g}_{m,i} = \begin{cases} 0, & \text{if } \mu_{m,i} = 0 \text{ and } g_{m,i} > 0; \\ g_{m,i}, & \text{otherwise.} \end{cases}$$

where $g_{m,i}$ denotes the ith element of g_m.

Compared with SSLA, ISLA represents an improvement by introducing three fine-adjusting strategies: weak oscillation detection (Step 5), adjustment of the threshold $\sigma_{\max}$ ($h(s)\sigma_{\max}$ in Step 6) and improved projection of multipliers (Step 7'). Numerical experiments (Section 6.2.2) demonstrate that these strategies are essential to improve the efficiency of ISLA.

In ISLA, $F(\mu_m)$ oscillates between two function values or within a local area. To escape the local area, ISLA must accumulate enough paths ($\|d_m\|$ in Step 9) to adjust δ_l (Step 6). Due to the tiny steplengths in late iterations, the accumulation is very time-consuming. To avoid running useless iterations, Step 5 detects the weak oscillation by recording the precious function values and changing σ_m directly, whereas Step 6 reduces the threshold $\sigma_{\max}$ by $h(s)$.

The convergence of the algorithm is guaranteed by adjusting the estimation of the dual function value (F^m_{lev}). The adjustment is realized via the weak oscillation and strong oscillation detection methods. Weak oscillation detection is a fine adjustment which improves the efficiency, whereas strong oscillation is a rough adjustment guaranteeing convergence. The details of the analysis are provided in Section 5.5.

5.5. Convergence analysis

Let $x = \{x_{ijk}, y_{i_1,i_2,j}, t_{ij}\}$, $X = \{x \in Z^{r_1} \cup R^{r_2} : x$ satisfies the constraints (5)–(12) and (14) and (15)$\}$, $r_1 = (S-1)|\Omega|(|\Omega|-1) + |\Omega|\sum_{j=1}^{S-1} M_j$, $r_2 = (S-1)|\Omega|$, $F(\mu) = -L(\mu)$. Based on Prop. 5.1.2. in Bertsekas (1999), we can determine that $L(\mu)$ is a concave function and $F(\mu)$ is a convex function. Hence, $F(\mu)$ and $L(\mu)$ are continuous over $\Psi = \{\mu | \mu \geqslant 0\}$. Let $F^* = \min_{\mu\in\Psi} F(\mu)$ and $\Psi^* = \{\mu | F(\mu) = F^*, \mu \in \Psi\}$.

Assumption 1. $g_F(\mu)$, the subgradient of $F(\mu)$, is bounded, which means that there exists an scalar G such that $\|g_F(\mu)\| < G$.

Remark 1. Although the boundedness detection method can avoid the unboundedness of the subgradient, it cannot guarantee the assumption.

Lemma 1. *Let Assumption 1 hold and let $\{\mu_m\}$ be generated by the following method (similar to Step 7)*

$$\mu_{m+1} = P_{R^+}(\mu_m - s_m g_m), \quad m = 0, 1, 2, \ldots, \tag{22}$$

where $s_m \geqslant 0$ and g_m is the subgradient of $F(\mu)$. Therefore, for any given $\mu \in \Psi = \{\mu | \mu \geqslant 0\}$,

$$\|\mu_{m+1} - \mu\|^2 \leqslant \|\mu_m - \mu\|^2 - 2s_m[F(\mu_m) - F(\mu)] + s_m^2\|g_m\|^2.$$

Proof. Lemma 1 can be easily proved using Prop. 6.3.1 in Bertsekas (1999). □

Remark 2. If ISLA adopts Step 7' to update multipliers, Lemma 1 also holds.

The next result is a classical convergence result for the conventional subgradient method of Polyak (1967).

Lemma 2. *Let Assumption 1 hold and assume that the stepsize s_m in* Eq. (22) *is such that*

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

Thus, the sequence $\{\mu_m\}$ generated by the method in Lemma 1 converges to an element of Ψ^*.

The proof method of the following lemma is similar to the method described in Nedic and Bertsekas (2001) and Goffin and Kiwiel (1999).

Lemma 3. *Let Assumption 1 hold. For ISLA, we have $l \to \infty$, and either $\inf_{m\geqslant 0} F(\mu_m) = -\infty$ or $\lim_{l\to\infty} \delta_l = 0$.*

Based on the above analysis and the methods described in Goffin and Kiwiel (1999) and Nedic and Bertsekas (2001), we can obtain the following convergence result.

Proposition 3. *Let Assumption 1 hold. For ISLA, we have $\inf_{m\geqslant 0} F(\mu_m) = F^*$.*

Proof. If $\lim_{l\to\infty} \delta_l > 0$, according to Lemma 3, it can be determined that $\inf_{m\geqslant 0} F(\mu_m) = -\infty$. Thus, the proposition is proved. Therefore, we assume that $\lim_{l\to\infty} \delta_l = 0$. Let L be given by $L = \{l \in \{0,1,2,\ldots\} | \delta_{l+1} = \beta\delta_l\}$. Thus, from Steps 5, 6 and 9, we obtain $\sigma_{m+1} = \sigma_m + \|s_m g_m\| = \sum_{i=M[l]}^{m} \|s_i g_i\|$ such that $M[l+1] = m+1$ and $l+1 \in L$ whenever $\sum_{i=M[l]}^{m+1} s_i\|g_i\| > \sigma_{\max}/(s+1)$ at Step

6. Hence, $\sum_{i=M[l]}^{M[l+1]-1} s_i > \sigma_{\max}/(G(s+1))$. Because the cardinality of L is infinite and s is infinite, we have $\sum_{i=M[\hat{l}]}^{\infty} s_i \geqslant \sum_{l \geqslant \hat{l}, l \in L} \sum_{i=M[l]}^{M[l+1]-1} s_i > \sum_{s=0}^{\infty} \sigma_{\max}/(G(s+1)) \to \infty$. Now, by contradiction, let us assume that $\inf_{m \geqslant 0} F(\mu_m) > F^*$ such that for some $\hat{\mu} \in \Psi = \{\mu | \mu \geqslant 0\}$ and some $\varepsilon > 0, \inf_{m \geqslant 0} F(\mu_m) - \varepsilon \geqslant F(\hat{\mu})$. Since $F(\mu)$ is continuous over Ψ and $\delta_l \to 0$, there is a large enough $\hat{l}$ such that $\delta_l \leqslant \varepsilon$ for all $l \geqslant \hat{l}$, so that for all $m \geqslant M[\hat{l}], F_{\text{lev}}^m = F_{\text{best}}^{M[l]} - \delta_l \geqslant \inf_{m \geqslant 0} F(\mu_m) - \varepsilon \geqslant F(\hat{\mu})$. Using this relation and Lemma 1, for $\mu = \hat{\mu}$ and $s_m = t[F(\mu_m) - F_{\text{lev}}^m]/\|g_m\|^2$, we obtain

$$\begin{aligned} \|\mu_{m+1} - \hat{\mu}\|^2 &\leqslant \|\mu_m - \hat{\mu}\|^2 - 2s_m[F(\mu_m) - F(\hat{\mu})] + s_m^2\|g_m\|^2 \\ &\leqslant \|\mu_m - \hat{\mu}\|^2 - 2s_m[F(\mu_m) - F_{\text{lev}}^m] + s_m^2\|g_m\|^2 \\ &\leqslant \|\mu_m - \hat{\mu}\|^2 - t(2-t)[F(\mu_m) - F_{\text{lev}}^m]^2/\|g_m\|^2. \end{aligned}$$

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

$$t(2-t)\sum_{m=M[\hat{l}]}^{\infty} [F(\mu_m) - F_{\text{lev}}^m]^2/\|g_m\|^2 \leqslant \|\mu_{M[\hat{l}]} - \mu\|^2.$$

and consequently $\sum_{m=0}^{\infty} s_m^2 < \infty$. Because $\sum_{m=0}^{\infty} s_m = \infty$ and $s_m > 0$, we can obtain that $\liminf F(\mu_m)_{m \to \infty} = F^*$ (Lemma 2), which contradicts the assumption.

Remark 3. According to the presented proof and Lemma 2, we know that SSLA and ISLA are actually two different types of step-size subgradient methods. Numerical experiments (Section 6.2.2) demonstrate that ISLA performs much better than SSLA in addressing SCC scheduling problems.

6. Numerical experiments

6.1. Algorithm parameters

To analyze the performance of the presented algorithm in solving the SCC scheduling problem, our computational study compares three types of algorithms. For simplicity, we denote the SSLA adopting standard subgradient (Step 7 in Section 5.4) as S, the ISLA adopting standard subgradient as I, the CSA adopting standard subgradient as C, the boundedness detection method (in Section 5.2.1) as B, the time horizon method as H, and the improved projection method (Step 7' in Section 5.4) as P,

CB: the algorithm C adopting method B;
CBP: the algorithm C adopting methods B and P;

The abbreviations IB, IBP, IH and IHP have similar meanings. The parameters of three types of algorithms (CSAs, ISLAs and SSLAs) are set as follows.

For ISLAs (IB, IBP, IH and IHP), the parameters are set as follows: $\varepsilon_1 = 1e-5$, $\varepsilon_2 = 1e-5$, $\varepsilon_3 = 1e-5$, $W = 5$, $\alpha = 0.8$, $\beta = 0.8$, $t = 1.2$, $U = \sum_{j=1}^{S} \sum_{i \in \Omega} P_{ij}$, $\delta_1 = (P(\mu_1) + F(\mu_1))/5$, $\sigma_{\max} = (P(\mu_1) + F(\mu_1))/\|g_F(\mu_1)\|$, $\mu_1 = 10^n g_F(0)/\|g_F(0)\|^2$, $n = \left\lfloor \log_{10}\left(\|g_F(0)\|^2/U\right) \right\rfloor$.

For CSAs (CB, CBP, CH and CHP), we set the maximum iterations equal to 500, and $\|g_m\| < \varepsilon_1$ as the termination criterion. The step-size of CSAs is the same as Step 7 of ISLA except that $-F_{\text{lev}}^m$ is replaced with $+P_{\text{best}}$. This step-size rule (also called target value rule) is commonly adopted in LR approaches Bertsekas (1999).

For SSLAs (SB, SBP, SH and SHP), the parameters are the same as those of ISLAs except that SSLAs set $h(s) \equiv 1$ in Step 6.

All algorithms are evaluated with respect to two performance measures: relative duality gap and computational time. Because LR cannot guarantee optimal solutions, the relative duality gap $g = (u - l)/l \times 100\%$ is used as a criterion to measure the solution optimality of the algorithm, where u is the upper bound derived from the modified feasible solution and l is the lower bound obtained from the solution of the relaxed problem. All of algorithms were implemented in the C# language and run on a PC with 3.4-GHz 8-GB-CPU using the Windows 7 operating system (64-bit).

6.2. Problem instances and computation results

6.2.1. Problem instances

By carefully analyzing the actual production data from Baosteel Complex of China, we generate a total of 2520 test instances. The problem structures are described as follows.

(1) The number of stages varies among three levels: 3, 4 and 5. The number of machines at each stage varies among three levels: 3, 4 and 5. The number of batches on each machine in the last stage varies among four levels: 2, 3, 4 and 5. The number of jobs in each batch varies among seven levels: 2, 3, 4, 5, 6, 7 and 8.
(2) The objective function coefficients $C_1 = 10$, $C_j = 10 + 20(j-1)(1 < j < S)$, $D_1 = 10$, $D_2 = 110$. The setup time $Su = 80$. The due date is $d_{b(k-1)+m+1} = d_{b(k-1)+m} + \sum_{i=n_{k,1}}^{n_{k,m}} P_{i,S} + T_B$, $d_{i_0} = \sum_{j=1}^{S-1} P_{i_0,j} + \sum_{j=1}^{S-1} T_{j,j+1}$, $i_0 = s(b(k-1)) + 1$, $\bar{k}_m = b(k-1) + m$, $n_{k,1} = s(k_m)$, $n_{k,m} = s(\bar{k}_m) - s(\bar{k}_m - 1)$, $1 \leqslant m < b(k) - b(k-1)$, $1 \leqslant k \leqslant M_S$.
(3) The integer of the transportation time $T_{j,j+1}$ is generated uniformly from $[3, 6]$. The integer of the processing time $P_{i,j}$ is generated uniformly from $[36, 50]$.

We consider nine combinations of the stage and machine levels (Stages vs. Machines): 3 vs. 3, 3 vs. 4, 3 vs. 5, . . ., 5 vs. 5. There are $4 \times 7 = 28$ 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 aforementioned combination configurations, we randomly generate 10 instances, resulting in a total of $3 \times 3 \times 4 \times 7 \times 10 = 2520$ test instances. For simplicity, we denote Stages as S, Machines as M, Batches as B and Jobs as J.

6.2.2. Comparisons between ISLAs and SSLAs

Tables 1 and 2 present the results of applying the ISLAs and SSLAs to some SCC scheduling problems. In order to analyze the effects of the weak oscillation detection (Step 5) and $h(s)$ on the ISLAs, the two tables also provide the results of ISLAs without Step 5, which are denoted as ISLA0s (IB0, IBP0, IH0 and IHP0). Table 1 shows that the running times of SSLAs for small-scale problems are large. The shortest average running time of SSLAs is 1138.2 *s*, while the longest average running time of ISLAs is only 1.3 *s*. Through careful examination, we find that most of the computational time of ISLAs is spent on accumulating tiny paths $\|d_m\|$ (Step 9) to exceed the threshold $\sigma_{\max}$ (Step 6) during late iterations. On the other hand, Tables 1 and 2 demonstrate that ISLAs perform substantially better than SSLAs; IHP in particular performs best. Furthermore, although the only difference between ISLA0s and SSLAs is $h(s)$, ISLA0s performs much better than SSLAs. This implies that $h(s)$ has a significant impact on the efficiency. Similarly, the comparisons between ISLAs and ISLA0s show that Step 5 improves the efficiency of ISLAs significantly. These results suggest that the efficiency can be improved dramatically by introducing the three fine-adjusting strategies in Section 5.4.

6.2.3. Comparisons between ISLAs and CSAs

Before providing the statistical results of all instances, it is worth noting that the gaps of some algorithms for some instances are larger than 100%. We call these algorithms failure algorithms.

Table 1
Average running times of SSLAs, ISLAs and ISLA0s (s).

S vs. M vs. B vs. J	SB	SBP	SH	SHP	IB	IBP	IH	IHP	IB0	IBP0	IH0	IHP0
3 vs. 3 vs. 2 vs. 2	178.1	51.0	125.1	37.2	0.3	0.2	0.2	0.2	11.6	3.5	8.2	2.5
3 vs. 3 vs. 2 vs. 3	1664.7	493.1	1216.6	353.7	0.5	0.4	0.3	0.3	73.8	20.9	53.3	14.9
3 vs. 3 vs. 2 vs. 4	1452.7	367.9	1058.8	269.0	0.6	0.5	0.5	0.4	64.8	16.4	47.3	11.9
3 vs. 3 vs. 2 vs. 5	11169.5	2494.6	8232.9	1824.8	1.4	0.8	1.0	0.6	751.7	169.4	557.1	122.7
3 vs. 3 vs. 2 vs. 6	11294.7	2484.2	8402.3	1843.9	1.6	1.1	1.2	0.8	709.4	162.0	528.3	118.1
3 vs. 3 vs. 2 vs. 7	11728.6	2498.7	8997.5	1900.6	2.1	1.4	1.6	1.0	752.9	157.9	565.7	115.3
3 vs. 3 vs. 2 vs. 8	11522.9	2323.7	8576.0	1738.4	2.4	1.8	1.9	1.3	713.0	147.6	537.2	107.7
Average	7001.6	1530.4	5229.9	1138.2	1.3	0.9	0.9	0.6	439.6	96.8	328.2	70.4

Table 2
Average duality gaps of SSLAs, ISLAs and ISLA0s (%).

S vs. M vs. B vs. J	SB	SBP	SH	SHP	IB	IBP	IH	IHP	IB0	IBP0	IH0	IHP0
3 vs. 3 vs. 2 vs. 2	3.79	3.80	3.79	3.80	3.79	3.80	3.79	3.80	3.79	3.80	3.79	3.80
3 vs. 3 vs. 2 vs. 3	6.55	6.55	6.55	6.55	6.55	6.55	6.55	6.55	6.55	6.55	6.55	6.55
3 vs. 3 vs. 2 vs. 4	9.10	9.10	9.10	9.10	9.10	9.10	9.10	9.10	9.10	9.10	9.10	9.10
3 vs. 3 vs. 2 vs. 5	7.09	7.09	7.09	7.09	7.09	7.09	7.09	7.09	7.09	7.09	7.09	7.09
3 vs. 3 vs. 2 vs. 6	9.62	9.62	9.62	9.62	9.62	9.62	9.62	9.62	9.62	9.62	9.62	9.62
3 vs. 3 vs. 2 vs. 7	9.07	9.07	9.07	9.07	9.07	9.07	9.07	9.07	9.07	9.07	9.07	9.07
3 vs. 3 vs. 2 vs. 8	8.33	8.33	8.33	8.33	8.33	8.33	8.33	8.33	8.33	8.33	8.33	8.33
Average	7.65	7.65	7.65	7.65	7.65	7.65	7.65	7.65	7.65	7.65	7.65	7.65

Table 3
The instances and the corresponding failure algorithms.

S vs. M vs. B vs. J	Failure algorithms
4 vs. 5 vs. 5 vs. 8	CB, CBP, CH, IB, IBP
5 vs. 3 vs. 4 vs. 8	CB, CBP, IB, IBP
5 vs. 5 vs. 4 vs. 8	CH
5 vs. 5 vs. 5 vs. 7	CH
5 vs. 5 vs. 5 vs. 8	CH

Table 4
Average duality gaps (%) of ISLAs and CSAs.

S vs. M	CB	CBP	CH	CHP	IB	IBP	IH	IHP
3 vs. 3	5.91	5.99	5.91	5.99	5.90	5.90	5.90	5.90
3 vs. 4	6.16	6.19	6.26	6.19	6.13	6.10	6.19	6.11
3 vs. 5	6.29	6.26	6.79	6.26	6.21	6.18	6.24	6.18
4 vs. 3	6.42	6.48	6.42	6.48	6.42	6.42	6.43	6.42
4 vs. 4	6.75	6.76	6.86	6.76	6.74	6.68	6.86	6.69
4 vs. 5	7.01	7.03	7.88	7.27	7.02	6.96	7.44	7.18
5 vs. 3	6.65	6.69	6.95	6.99	6.62	6.62	6.97	6.92
5 vs. 4	7.18	7.20	7.28	7.20	7.17	7.13	7.33	7.13
5 vs. 5	7.85	7.49	7.45	7.57	7.52	7.39	8.02	7.43
Average	6.69	6.68	6.87	6.75	6.64	6.60	6.82	6.66

The five instances and corresponding failure algorithms are presented in Table 3.

After careful examination, we find that the reasons for failure of different algorithms are different. For the CH, the optimal dual value may be obtained at boundaries and it is every large in iterations. The maximum iteration number 500 is not large enough to obtain a good value. For the algorithms adopting method B, when the dual function value is unbounded, the Lagrangian multipliers will be decreased until the dual function is bounded. Hence, sometimes, the Lagrangian multipliers may be very small and even close to zero. In particular, CB and CBP will fall into a local area in iterations when the Lagrangian multipliers are too small, whereas LB and IBP will be terminated early when the descent range δ_l is less than the given threshold. Indeed, if L adopts method B, it will not guarantee Assumption 1, which is the precondition for convergence. However, it is remarkable that IH and IHP solve all instances successfully.

To compare the successful results, all of the following statistical results exclude the results obtained for all algorithms for the five instances in Table 3. Because the full details of the test results are quite substantial, we summarize these data in more compact tables and statistic results. Table 4 presents the average duality gaps, and Table 5 presents the average lower bounds. Table 4 shows that the average duality gaps generated by all of the algorithms are less than 6.87%. The duality gaps of the four ISLAs are smaller than those of the four CSAs. Moreover, the gaps of the four ISLAs are less than 6.82%, whereas those of the four CSAs are greater than 6.68%. The gaps of CB (CBP) are slightly better than those of CH (CHP), whereas those of IB (IBP) are nearly the same as those of IH (IHP). A more careful examination of the experimental data shows that IHP yields better results than the four CSAs in 241 out of 247 instances. Similar results can also be found in Table 5.

Table 5
Average lower bounds of ISLAs and CSAs.

S vs. M	CB	CBP	CH	CHP	IB	IBP	IH	IHP
3 vs. 3	823,343	822,842	823,343	822,842	823,386	823,380	823,387	823,380
3 vs. 4	1,096,841	1,096,958	1,094,242	1,096,958	1,096,733	1,097,406	1,095,384	1,097,372
3 vs. 5	1,371,944	1,372,454	1,361,841	1,372,477	1,372,343	1,373,035	1,371,955	1,373,034
4 vs. 3	1,002,419	1,001,872	1,002,357	1,001,859	1,002,280	1,002,445	1,002,203	1,002,445
4 vs. 4	1,335,849	1,335,926	1,333,505	1,335,949	1,335,341	1,336,417	1,333,242	1,336,417
4 vs. 5	1,469,213	1,469,099	1,455,795	1,666,335	1,468,816	1,469,712	1,663,127	1,667,551
5 vs. 3	1,134,748	1,134,273	1,229,463	1,229,133	1,134,818	1,134,850	1,228,459	1,229,770
5 vs. 4	1,640,195	1,639,936	1,638,273	1,639,962	1,639,842	1,640,479	1,637,269	1,640,480
5 vs. 5	2,031,565	2,050,206	1,598,395	2,046,343	2,045,280	2,050,911	2,030,074	2,050,814
Average	1,322,902	1,324,841	1,281,913	1,356,873	1,324,316	1,325,404	1,353,900	1,357,918

The lower bounds of the four ISLAs are larger than those of the four CSAs; in particular, the lower bounds of IBP and IHP are the largest.

Table 6 presents the average running time of the eight algorithms in different combinations of the stage and machine. Fig. 2 shows the average running time of the eight algorithms for various problem sizes (problem size = S × M × B × J). Pronounced difference among the eight algorithms can be observed. IHP and IBP are far superior to the four CSAs. Table 6 and Fig. 2 demonstrate that the average running times of IHP is the least. In addition, Fig. 2 shows that the improved projection of multipliers method (Step 7') plays an important role in reducing the running time of ISLA. The running time of IH fluctuates greatly as problem size increases, whereas IHP increases smoothly and slowly with problem scale.

In summary, the algorithm IHP performs much better than the other seven algorithms in terms of robustness, duality gap and running time.

Table 6
Average running time (*s*) of ISLAs and CSAs.

S vs. M	CB	CBP	CH	CHP	IB	IBP	IH	IHP
3 vs. 3	16.7	16.0	12.4	11.7	5.1	2.7	3.8	2.0
3 vs. 4	28.2	26.6	21.1	19.5	16.6	4.9	19.8	3.8
3 vs. 5	45.0	41.8	34.0	30.7	32.0	7.7	32.0	5.8
4 vs. 3	29.1	28.0	21.7	20.6	9.9	5.0	11.6	3.7
4 vs. 4	53.7	51.0	40.3	37.6	31.0	9.6	46.9	7.9
4 vs. 5	78.6	74.3	65.8	60.5	51.1	15.8	84.8	12.5
5 vs. 3	40.9	39.5	32.6	31.1	15.6	7.6	21.4	5.9
5 vs. 4	84.3	80.8	63.2	59.5	44.2	15.2	73.8	12.6
5 vs. 5	118.6	111.9	88.9	82.2	112.7	22.5	175.7	20.7
Average	55.0	52.2	42.2	39.3	35.3	10.1	52.2	8.3

6.2.4. Comparisons between IHP and CPLEX

In this experiment, we considered a representative combination of the stage and machine level (*i.e.*, S vs. M = 3 vs. 3). We solved the instances using the standard MILP software (CPLEX 12.4) with the default settings and the following termination criterion: relative MIP gap tolerance: 0.01; optimizer time limit: 300 *s*.

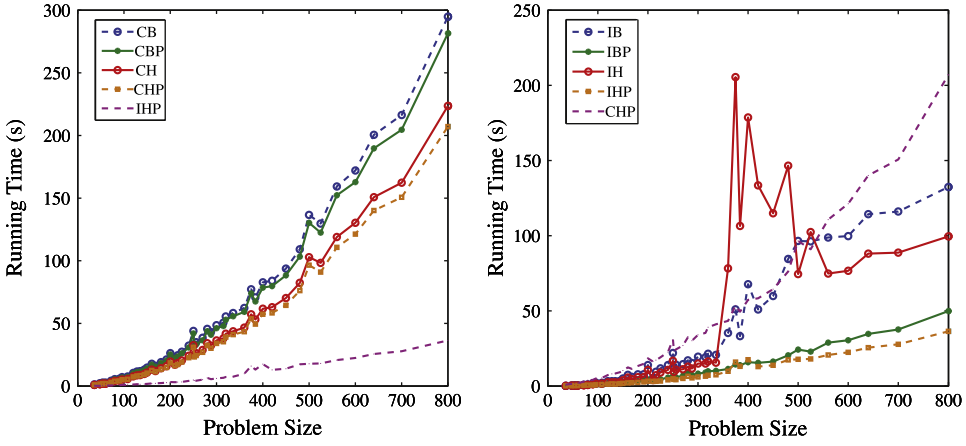

Fig. 2. Comparisons of the running time between ISLAs and CSAs.

Table 7
Results of IHP and CPLEX (S vs. M = 3 vs. 3) (The minimal gaps are in bold).

B vs. J	CPLEX				IHP			
	LB	*UB*	*g* (%)	*t* (s)	*LB*	*UB*	*g* (%)	*t* (s)
2 vs.3	122,940	124,180	**1.01**	8.9	117,808	127,350	8.10	0.3
2 vs. 4	188,587	190,490	**1.01**	61.9	179,298	191,720	6.93	0.5
2 vs. 5	243,850	271,890	**11.50**	301.6	243,846	275,580	13.01	0.7
2 vs. 6	330,180	348,060	**5.42**	303.2	330,178	367,500	11.30	0.9
2 vs. 7	430,110	465,430	8.21	305.5	430,106	451,920	**5.07**	0.9
2 vs. 8	528,920	626,650	18.48	310.0	528,916	568,870	**7.55**	1.2
3 vs. 3	239,560	247,150	**3.17**	301.0	239,555	252,470	5.39	0.4
3 vs. 4	356,070	365,770	**2.72**	303.2	356,064	385,080	8.15	0.7
3 vs. 5	523,970	539,850	**3.03**	307.3	523,965	553,470	5.63	1.0
3 vs. 6	675,950	745,710	10.32	315.9	675,945	726,570	**7.49**	1.4
3 vs. 7	870,750	965,790	10.91	331.6	870,745	941,090	**8.08**	1.9
3 vs. 8	1,103,340	1,186,670	7.55	358.2	1,103,326	1,156,350	**4.81**	2.2
4 vs. 3	402,640	416,770	**3.51**	303.3	396,283	421,050	6.25	0.8
4 vs. 4	595,290	600,060	0.80	20.5	595,278	599,560	**0.72**	0.7
4 vs. 5	848,760	898,900	5.91	323.7	848,750	882,210	**3.94**	1.5
4 vs. 6	1,141,220	1,264,170	10.77	357.8	1,141,207	1,231,320	**7.90**	2.6
4 vs. 7	1,496,230	1,608,130	7.48	402.7	1,496,212	1,572,500	**5.10**	2.9
4 vs. 8	1,854,650	2,205,810	18.93	535.5	1,854,631	1,993,770	**7.50**	4.0
5 vs. 3	586,430	597,870	**1.95**	307.3	586,418	602,450	2.73	0.9
5 vs. 4	906,690	975,300	7.57	323.6	906,682	969,380	**6.92**	1.8
5 vs. 5	1,268,590	1,361,260	7.30	366.4	1,268,572	1,313,360	**3.53**	2.3
5 vs. 6	1,727,900	1,980,200	14.60	475.0	1,727,879	1,821,560	**5.42**	3.4
5 vs. 7	2,234,330	6,831,120	205.73	595.2	2,234,298	2,320,670	**3.87**	4.3
5 vs. 8	2,826,650	9,158,750	224.01	771.7	2,826,620	2,946,770	**4.25**	5.4
Average	795,889	1,242,219	24.66	307.3	794,518	839,283	**6.77**	1.6

As in the LR approach, we can obtain the lower bound (LB) and upper bound (UB) of the CPLEX solver. Table 7 presents the computational results of IHP and CPLEX. The table shows that the gaps of CPLEX MIP solver are less than 2% when the problem size is less than 24 ($M \times B \times J \leqslant 24$), wheras the gaps are larger than 10% when the problem size is greater than 60 ($M \times B \times J \geqslant 60$). Table 7 also shows that the differences in the lower bounds between two methods are small, wheras the best feasible solutions of IHP are clearly superior to those of CPLEX when the problem size is greater than 75. These results indicate that the relative gaps of IHP are clearly smaller than those of CPLEX when the problem size is more than 75. In addition, as shown in Table 7, the average running time of IHP is 1.6 *s*, while the average running time of CPLEX is 307.3 *s*.

7. Conclusions

This paper proposed a novel LR approach to a HFS scheduling problem arising from the SCC process. The LD problem was solved by an improved subgradient level algorithm, which guarantees convergence without requiring the optimal dual value in advance. Furthermore, three fine-adjusting strategies were introduced to improve the efficiency of the algorithm. Other approaches based on subgradient algorithms, including CSAs and SSLAs, were also proposed to solve the dual problem. We also addressed the unboundedness problem of Lagrangian relaxation. Comparisons among all algorithms demonstrated that the algorithm IHP is the most promising one in terms of robustness, duality gap and running time. In particular, the running time of IHP is far shorter than that of other algorithms for large-scale problems. Moreover, the comparisons between IHP and the CPLEX (12.4) MIP solver demonstrated that IHP is notably effective and efficient. Further research will be focused on the development and application of the proposed approaches and other similar approaches, such as cutting plane methods.

Acknowledgements

We thanks Prof. Lixin Tang and Prof. Ruben Ruiz for providing valuable suggestion in our revision. This research is partially supported by National Science Foundation of China (61174187, 61333006, 61104179 and 61104174), Program for New Century Excellent Talents in University (NCET-13-0106), Specialized Research Fund for the Doctoral Program of Higher Education (20130042110035), Science Foundation of Liaoning Province in China (2013020016), Basic scientific research foundation of Northeast University under Grant N110208001, Starting foundation of Northeast University under Grant 29321006, IAPI Fundamental Research Funds (2013ZCX04-04), the Fundamental Research Funds for the Central Universities (N120708001), and Shandong Province Key Laboratory of Intelligent Information Processing and Network Security (Liaocheng University).

References

Atighehchian, A., Bijari, M., & Tarkesh, H. (2009). A novel hybrid algorithm for scheduling steelmaking continuous casting production. *Computers & Operations Research, 36*, 2450–2461.

Bellabdaoui, A., & Teghem, J. (2006). A mixed-integer linear programming model for the continuous casting planning. *International Journal of Production Economics, 104*, 260–270.

Bertsekas, D. P. (1999). *Nonlinear programming*. Massachusetts: Athena Scientific.

Brannlund, U. (1993). On relaxation methods for nonsmooth convex optimization. Ph.D. thesis, Royal Institue of Technology, Stockholm.

Brannlund, U., Kiwiel, K. C., & Lindberg, P. (1995). A descent proximal level bundle method for convex nondifferentiable optimization. *Operations Research Letters, 17*, 121–126.

Chen, H., Chu, C., & Proth, J. M. (1998). An improvement of the lagrangian relaxation approach for job shop scheduling: A dynamic programming method. *IEEE Transactions on Robotics and Automation, 13*, 786–795.

Chen, H., & Luh, P. B. (2003). An alternative framework to lagrangian relaxation approach for job shop scheduling. *European Journal of Operational Research, 149*, 499–512.

Gicquel, C., Hege, L., Minoux, M., & van Canneyt, W. (2012). A discrete time exact solution approach for a complex hybrid flow-shop scheduling problem with limited-waited constraints. *Computers & Operations Research, 39*, 629–636.

Goffin, J. L., & Kiwiel, K. C. (1999). Convergence of a simple subgradient level method. *Mathematical Programming, 85*, 207–211.

Hoitomt, D. J., Luh, P. B., & Pattipati, K. R. (1993). A practical approach to job shop scheduling problems. *IEEE Transactions on Robotics and Automation, 9*, 1–13.

Hoogeveen, J. A., & de Velde, S. L. V. (1991). Scheduling around a small common due date. *European Journal of Operations Research, 55*, 237–242.

Kumar, V., Kumar, S., Chan, F. T. S., & Tiwari, M. K. (2006). Auction-based approach to resolve the scheduling problem in the steel making process. *International Journal of Production Research, 44*, 1503–1522.

Lenstra, J., Rinnoy, K. A. H. G., & Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics, 1*, 343–362.

Luh, P. B., Hoitomt, D. J., Max, E., & Pattipati, K. R. (1990). Scheduling generation and reconfiguration for parallel machines. *IEEE Transactions on Robotics and Automation, 6*, 687–696.

Markela, M. M. (2002). Survey of bundle methods for nonsmooth optimization. *Optimization Methods and Software, 17*, 1–29.

Missbauer, H., Hauberb, W., & Stadler, W. (2009). A scheduling system for the steelmaking-continuous casting process: A case study from the steel-making industry. *International Journal of Production Research, 47*, 4147–4172.

Nedic, A., & Bertsekas, D. P. (2001). Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal of Optimization, 12*, 109–138.

Nishi, T., Hiranaka, Y., & Inuiguchi, M. (2010). Lagrangian relaxation with cut generation for hybrid flow shop scheduling problems to minimize the total weighted tardiness. *Computers & Operations Research, 37*, 189–198.

Oguz, C., Ercan, M. F., Cheng, T. C. E., & Fung, Y. F. (2003). Heuristic algorithms for multiprocessor task scheduling in a two-stage hybrid flow-shop. *European Journal of Operations Research, 149*, 390–403.

Pacciarelli, D., & Pranzo, M. (2004). Prodution scheduling in a steelmaking-continuous casting plant. *Computers & Chemical Engineering, 28*, 2823–2835.

Pan, Q.-K., Wang, L., Mao, K., Zhao, J.-H., & Zhang, M. (2013). An effective artificial bee colony algorithm for a real-world hybrid flowshop problem in steelmaking process. *IEEE Transactions on Automation Science and Engineering, 10*, 307–322.

Polyak, B. (1967). A general method of solving extremum problems. *Soviet Mathematics Doklady, 8*, 593–597.

Ribas, I., Leisten, R., & Framinan, J. (2010). Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Computers & Operations Research, 37*, 1439–1454.

Ruiz, R., & Maroto, C. (2006). A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility. *European Journal of Operations Research, 169*, 781–800.

Ruiz, R., Serifoglu, F., & Urlings, T. (2008). Modeling realistic hybrid flowshop scheduling problems. *Computers & Operations Research, 35*, 1151–1175.

Ruiz, R., & Vazquez-Rodriguez, J. (2010). The hybrid flow shop scheduling problem. *European Journal of Operations Research, 205*, 1–18.

Tang, L., Liu, J., Rong, A., & Yang, Z. (2001). A review of planning and scheduling systems and methods for integrated steel production. *European Journal of Operations Research, 133*, 1–20.

Tang, L., Luh, P. B., Liu, J., & Fang, L. (2002). Steel-making process scheduling using lagrangian relaxation. *International Journal of Production Research, 40*, 55–70.

Tang, L., Xuan, H., & Liu, J. (2006). A new lagrangian relaxation algorithm for hybrid flow shop scheduling to minimize total weighted completion time. *Computers & Operations Research, 33*, 3344–3359.

Tang, L., Xuan, H., & Liu, J. (2007). Hybrid backward and forward dynamic programming based lagrangian relaxation for single machine scheduling. *Computers & Operations Research, 34*, 2625–2636.

Wang, S. H. (2003). An improved stepsize of the subgradient algorithm for solving the Lagrangian relaxation problem. *Computers & Electrical Engineering, 29*, 245–249.

Xuan, H., & Tang, L. (2007). Scheduling a hybrid flow shop with batch production at the last stage. *Computers & Operations Research, 34*, 2178–2733.

Zhang, X., & van de Velde, S. (2012). Approximation algorithm for the parallel flow shop problem. *European Journal of Operations Research, 216*, 544–552.