Supplementary Material for the Paper "Q-learning-based Hierarchical Cooperative Local Search for Steelmaking-continuous Casting Scheduling Problem"

XX, XXX, XX

This material includes seven supplementary parts of the paper "Q-learning-based Hierarchical Cooperative Local Search for Steelmaking-continuous Casting Scheduling Problem".

Part 1. Literature Review

Yu [1] develop a multi-objective nonlinear planning model of SCCSP by considering the effect of operational time delay, and propose a three-stage rescheduling heuristic, including batch splitting, forward scheduling, and backward. Tang [2] investigated the SCCSP under the circumstance of fresh charge arrival and suggested an enhanced differential evolution (DE) algorithm coupled with an incremental method to address this issue by rescheduling the casts in a dynamic environment. Li [3] developed a novel SCCSP model with numerous refining sub-stages and presented an improved fruit fly optimization algorithm to solve it. Assuming that the cast subsequence is unknown, Pan [4] suggested a cooperative co-evolutionary artificial bee colony algorithm that used two sub-swarms to tackle sub-problems. Taking machine breakdowns into account, Long [5] analyzed the impact of machine breakdown on the continuous caster stage, and developed a dynamic scheduling model to describe SCCSP with continuous caster breakdown. Then, a hybrid approach for this problem that combines genetic algorithms with variable neighborhood searches is suggested. Peng [6] investigated a real-world SCC process with adjustable processing times, where machine breakdown is considered an unexpected disruption, and devised an improved artificial bee colony-based rescheduling algorithm to minimize a weighted sum of the six realistic objectives. Considering the uncertain processing time of the SCC process, Kammammettu [7] established a stochastic scheduling model of the SCC process with unknown processing time and developed a multi-stage adaptive optimization strategy to reduce the maximum completion time. Jiang [8] proposed a prediction-based approach for SCCSP with uncertain processing time, and described the uncertain processing time of the SCC process as a stochastic variable with known expectation and variance.

Part 2. Problem description

2.1 The SCC process

As the most important process in contemporary steel manufacturing systems, steelmaking continuous casting (SCC) is primarily made up of three stages: steelmaking, refining, and continuous casting. During the steelmaking stage, hot molten iron is turned into molten steel by a Linz–Donawitz ('LD' in Fig. 2) converter [9]. When the carbon, sulphur, silicon, and other impurity content is reduced to a desirable level, the hot molten steels must be delivered to the refining stage. In the refining stage, the impurities are further removed using a Ruhrstahl Heraeus ('RH' in Fig. 2) [10], and the required alloy components are added to the molten steel. The charge is the basic unit in the steelmaking and refining stages. Each charge needs to go through the steelmaking stage and multiple refining stages according to the given precedence. The waiting time of the charges should be as short as possible to avoid a large temperature drop, which could result in an additional cost for reheating. The continuous casting stage is in charge of converting liquid steel into solid slabs. At this stage, a series of charges must be constantly cast, without a break, on the same continuous caster (CC) [11]. The cast is the basic unit in this stage, and there is a sequence-independent and detachable setup time before each cast processing. In our discussion of SCCSP, we assume that the processing times, the setup times, and the transportation times are known in advance and remain constant. Besides, identical parallel machines are set up at each stage to speed up the production process and balance the inter-flow of charges.

2.2 Solution representation and decoding strategy

In this subsection, a two-segment coding vector [4] and a bidirectional decoding strategy are introduced for the SCCSP. In the coding phase, a two-segment coding vector $\pi = (u, v)$ (see Fig. 1) that contains a charge subsequence u and a cast subsequence v is used to represent the solution. It is worth noting that the elements π_i^{cast} in the cast subsequence v is composed of α_i elements in charge subsequence u. For the example in Fig. 1, $\pi = \{(1,2,3,7,4,6,5), (1,2,3,4)\}$ is a feasible solution of SCCSP, where π_1^{cast} is consisted of π_1 and π_2 with the same backdrop color.

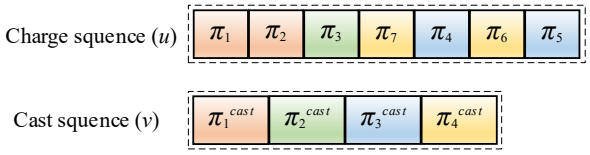


Fig. 1. The two-segment coding strategy.

In the decoding phase, a bidirectional decoding strategy is presented to convert a solution $x=(u, v)$ into a complete schedule. In the forward decoding process, each charge is assigned to the first available LD and RH following the charge subsequence u from the front to the end. In the continuous casting stage, each cast needs to be allocated to the first available CC without violating constraints. In particular, it is necessary to ensure that the charge in the same cast must be processed on the same machine. In the reverse decoding process, we first consider the charge of the last cast. If there is the idle time between the charge and its previous one in the same cast, it is necessary to shift the previous one enabling it to be connected to the latter one, so as to eliminate the casting break. We also address the remaining charges in the continuous casting stage using a similar method. Then,

we fix the operations at the continuous casting stage and move the operations in the steelmaking and refining stage to the right as compact as possible to reduce the waiting time between charges. The bidirectional decoding strategy is an effective method to reduce the average waiting time of schedule π by shifting the operations to the right without changing the completion time.

In the next subsection, we will introduce the permutation-based model based on the solution representation and decoding strategy proposed in this section, and illustrate the effectiveness of the bidirectional decoding strategy through a randomly generated example.

2.3 *The permutation-based model of SCCSP*

This paper considers the situation in which each of the charge follow the same processing route (i.e., steelmaking, refining, and then continuous casting). The permutation-based model of the SCCSP is given below. In the steelmaking and refining stage, charge k (k=1,2, …, N) need to undergo stage i=1 (i.e., the steelmaking stage) to the stage i=S −1 (i.e., the last refining stage) as shown in Fig. 1. There are m_i parallel machines in each stage and charge k can be processed on any one of them at the same stage. Besides, the transport times t_i must be considered when a charge is transported to the next stage. In the continuous casting stage, N charges are combined into Z casts according to the previous production scheme and each cast j (j=1,2, …, Z) only needs to complete one operation. A sequence-independent and separable setup time is required before a new cast is prepared for processing. Each cast or charge can only be processed on one machine at any time, and each machine can only process one cast or charge simultaneously. The waiting time of each cast or charge should be as short as possible to avoid a large temperature drop, which could result in an additional cost for reheating. In addition, each of the charge in the same cast need to be processed continuously according to the priority given in advance. The detailed notation descriptions used in the model are listed in Table 1.

TABLE I
NOTATIONS APPLIED IN OPTIMIZATION MODEL

Symbol	Description
M_s	Set of the machine contained in stages i, m_i is the total number of machines in stages i, M_s = {m_1, m_2, …m_i};
π_k	The *k-th* charge in the charge subsequemce u;
π_j^{cast}	The *j-th* cast in the cast subsequemce v, π_j^{cast} = {l_{j-1}+1, l_{j-1}+2, …, l_j}, where π_{j-1}+b is the *b-th* charge included in the π_j^{cast};
π	The schedule or permutation, π=(u, v).
α_j	The total number of the charge contained in cast j, α_j=‖π_j^{cast}‖;
$\pi_k^{i(a)}$	Charge k on the machine a in stage i;
t_i	The transportation time for a charge to be transported from stage i-1 to stage i;
ST_j	Independent setting time for the cast j;
$T_i(a)$	The total number of charges or casts on machine a in stage i;
$p_{\pi_i^{s(a)}}$	The processing time of π_i at machine a in stage s;
$L_{\pi_k^{s(a)}}$	The starting time of π_k at machine a in stage s;
$C_{\pi_k^{s(a)}}$	The completion time of π_k at machine a in stage s;
$C_{\max}(\pi)$	The maximum completion time of schedule π;
$f_{wait}(\pi)$	The average waiting time of schedule π;
Ψ_1	The weight coefficient of $C_{\max}(\pi)$;
Ψ_2	The weight coefficient of $f_{wait}(\pi)$;

Based on the above descriptions and the notations, the permutation-based model of the SCCSP can be established as follows.

$$C_{\pi_k^{1(a)}}=\sum_{b=1}^{k} p_{\pi_b^{1(a)}}, a=1,2,\ldots m_1, k=1,2,\ldots,T_1(a) \quad (1)$$

$$C_{\pi_k^{i(a)}}=\max(C_{\pi_k^{i-1}}+t_i, C_{\pi_{k-1}^{i(a)}})+p_{\pi_k^{i(a)}} \quad (2)$$

$$\Delta = C_{\pi_{l_{j-1}+b+1}^{S(a)}} - p_{\pi_{l_{j-1}+b+1}^{S(a)}} - C_{\pi_{l_{j-1}+b}^{S(a)}}, \quad b=1,2,\ldots,\alpha_j \quad (3)$$

$$\begin{cases} C_{\pi_{l_{j-1}+b+1}^{S}} = p_{\pi_{l_{j-1}+b+1}^{S}} + C_{\pi_{l_{j-1}+b}^{S}}, & \Delta \le 0; \\ C_{\pi_{l_{j-1}+b}^{S}} = C_{\pi_{l_{j-1}+b+1}^{S}} - p_{\pi_{l_{j-1}+b+1}^{S}}, & \Delta > 0; \end{cases} \quad (4)$$

$$C_{\pi_j^{cast(a)}}=\max(C_{\pi_{l_{j-1}+1}^{S(a)}} - p_{\pi_{l_{j-1}+1}^{S(a)}}, C_{\pi_{j-1}^{cast(a)}}+ST_{\pi_j^{cast(a)}})+p_{\pi_j^{cast(a)}} \quad (5)$$

$$C'_{\pi_k^i}=C_{\pi_k^i}+\min\{L_{\pi_{k+1}^i}-C_{\pi_k^i}, L_{\pi_k^{i+1}}-C_{\pi_k^i}\}, i=1,\ldots,S-1 \quad (6)$$

$$L'_{\pi_k^i}=C'_{\pi_k^i}-p_{\pi_k^i} \quad (7)$$

$$C_{\max}(x)=\max_{a=1,2,\ldots,m_s}\sum_{j=1}^{T_S(a)} C_{\pi_j^{cast(a)}} \quad (8)$$

$$f_{wait}(x)=\sum_{i=2}^{S}\sum_{k=1}^{N}(L_{\pi_k^{i(a)}}-L'_{\pi_k^{i-1(a)}}-p_{\pi_k^{i-1(a)}}) \quad (9)$$

$$\min f(x)=\psi_1 C_{\max}(x)+\psi_2 f_{wait}(x) \quad (10)$$

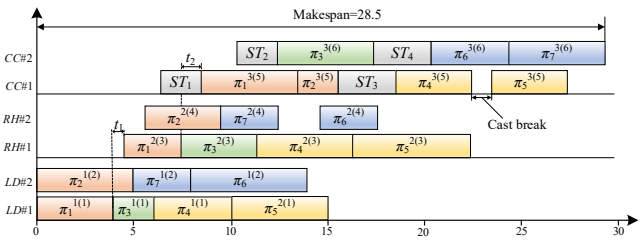
Eqs. (1)-(10) give the permutation-based model of SCCSP. Eq. (1) and Eq. (2) represent the calculation formula of the completion time of $\pi_k^{i(a)}$ in the forward decoding process. Eqs. (3) to (5) represent the elimination of cast break in the continuous casting stage. Eqs. (6) and (7) stand for the reverse decoding process to decrease the total waiting time. Eq. (8) represents the maximum completion time. Eq. (9) represents the average waiting time. Eq. (10) represents the weighted objective function of SCCSP. By assigning different weights to $C_{\max}(\pi)$ and $f_{wait}(\pi)$, the multi-objective problem is transformed into a minimizing single-objective problem. Besides, considering the actual production requires the completion of the production task as soon as possible, we choose Ψ_1=10, Ψ_2=1 in this paper, (the weight indicators are alternative according to actual production needs).

TABLE II
THE SCHEDULE OF PROCESSING TIME, SEQUENCE SETUP TIME AND TRANSPORTATION TIMES.

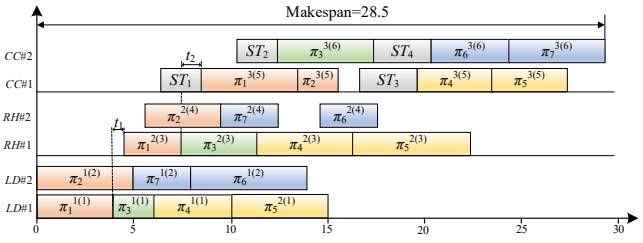
v	π_1^{cast}		π_2^{cast}	π_3^{cast}		π_4^{cast}		TT
u	π_1	π_2	π_3	π_4	π_5	π_6	π_7	
LD	4	5	2	4	5	6	3	—
RH	3	4	4	5	6	3	3	0.5
CC	5	2	4	3	3	4	5	1
ST	2		2	3		3		—

For the purpose of further illustrating the permutation-based model, Fig. 2 depicts the scheduling process as a Gantt chart decoded from the candidate solution π= {(1,2,3,7,4,6,5), (1,2,3,4)} based on a randomly generated instance in Table II. We begin at the stage of steelmaking in Fig. 1. (a). For the charge subsequence u=(1,2,3,7,4,6,5), we assign charge 1 to *LD*#1 and complete it at time $C_{\pi_1^{1(1)}}$ = $p_{\pi_1^{1(1)}}$ =4. The second charge is then given to *LD*#2 and finished at time $C_{\pi_2^{1(2)}}$ = $p_{\pi_2^{1(2)}}$ =5. Next, the remained charges are considered in accordance with Eq. (1) until all of the charges are finished. In the refining stage, all of the charges are assigned to the Gantt chart following Eq. (2). For example, we assigned charge 3 to *RH*#1 and start it at time:

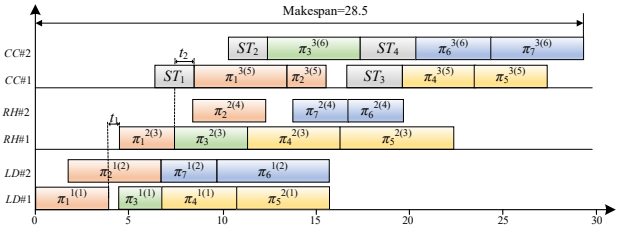
$\max(C_{\pi_3^1}+t_1, C_{\pi_1^{2(3)}})=\max(p_{\pi_1^{1(1)}}+p_{\pi_3^{1(1)}}+t_1, p_{\pi_1^{1(1)}}+t_1+p_{\pi_1^{2(3)}})=7.5$. After the refining stage, we address the cast subproblem according to v=(1,2,3,4), using a similar method. Note that the independent setting time should be considered before the casts enter into the production. According to the Gantt chart generated by Eq. (1) and Eq. (2), there is a cast break between charges 4 and 5 in *CC*#1. Consequently, we must shift the operation of charge 4 to right to eliminate cast break. Specifically, first of all, we need to obtain the length of cast break based on Eq. (3), $\Delta=C_{\pi_5^{3(5)}}-p_{\pi_5^{3(5)}}-C_{\pi_4^{3(5)}}=1>0$, and then, the completion time of charge 4 in *CC*#1 needs to be redefined according to Eq. (4), $C_{\pi_4^{3(5)}}=C_{\pi_5^{3(5)}}-p_{\pi_5^{3(5)}}=23.5$. Finally, a scheduling Gantt chart with a makespan 28.5 and total waiting time 33.5 is obtained without cast break shown in Fig. 3.(b). To reduce overall waiting time even more, the reverse decoding procedure is used in conjunction with Eq. (6) and Eq. (7) to move the operations to the right as compactly as possible. For example, charge 6 is assigned to *RH*#2 and complete it at time $C_{\pi_6^{2(4)}}=17.5$ in the refining stage, while charge 6 is next given to *CC*#2 and start it at time $L_{\pi_6^{3(6)}}=19.5$ in the continuous casting stage. There is a waiting time gap between them with a value of $L_{\pi_6^{3(6)}}-C_{\pi_6^{2(4)}}=2$, therefore we can shorten the wait by relocating the operation of charge 6 on machine *RH*#2 to the right, and resetting $C_{\pi_6^{2(4)}}=L_{\pi_6^{3(6)}}=19.5$. After the right move operation, the total waiting time of the schedule shown in Fig. 2. (b) is decreased to 26. It proves that the reverse decoding can effectively reduce the waiting time of given a scheduling plan. The new Gantt chart is shown in Fig. 2. (c).



(a) Gantt chart for a schedule of the forward decoding process.



(b) Gantt chart for a schedule without cast break.



(c) Gantt chart for a schedule of the reverse decoding process.

Fig. 2. The example of Gantt chart for the randomly generated instances.

Part 3. Analyses of Problem Complexity

This subsection aims to manifest the complexity of the SCCSP. In the SCC process, molten iron must be processed within a certain temperature range, and the waiting time of each job is limited or not acceptable. Hence SCCSP can be defined as a no-wait hybrid flow shop scheduling problem (NHFSP) with transportation times [7] and sequence-independent setup time (*SIST*) ($F_S(P_{m(i)})|$*no-wait*, *TT*, *SIST*$|\ C_{\max}$). In this paper, we confirm the NP-hardness of SCCSP via a reduction from a single-stage NHFSP with $C_{\max}$ criterion ($P_m|$*no-wait*$|C_{\max}$) and a known strongly NP-hard problem, i.e., the single machine scheduling problem with the $C_{\max}$ criterion (1$||C_{\max}$) [12].

Theorem 1: $P_m|$*no-wait*$|C_{\max}$ is NP-hard in the strong sense.

Proof: (1$||C_{\max}$) is defined as follows. There are n jobs need to be scheduled on a single machine. The processing time of each job k on the machine is p_k^{single}. All jobs are available for processing at time 0. At any moment, each job can only be processed on one machine and each machine can only process one job simultaneously. The aim is to minimize the $C_{\max}^{single}(\pi)$. Let $p_{\min}^{single}$ be the smallest one in the set $\{p_1^{single}, p_2^{single}, \ldots, p_n^{single}\}$. Given any instance *Ins*(1) of 1$||C_{\max}$, and a special instance *Ins*(2) of ($P_m|$ *no-wait* $|C_{\max}$) is constructed as follows. There are n jobs to be scheduled on m ($m \geq 3$) machines. The processing times of each job 1 to n-1 on the first machine and on all other machines are set by $p_{k,1}=p_k^{single}$ ($k\in[1, n\text{-}1]$) and $p_{n,1}=\frac{2}{3}p_{\min}^{single}$, $p_{k,l}=p_{\min}^{single}/l^2$ (l=2, 3, …, m) respectively. The aim is to minimize the $C_{\max}(\pi)$.

In fact, for the above instances *Ins*(1) and *Ins*(2), $C_{\max}^{single}(\pi)$ and $C_{\max}(\pi)$ are as follow:

$$C_{\max}^{single}(x)=\sum_{k=1}^{n}p_k^{single}=p_1^{single}+p_2^{single}+\ldots+p_{n-1}^{single}+p_n^{single} \quad (11)$$

$$C_{\max}(x)=\sum_{k=1}^{n-1}p_k^{single}+\frac{2}{3}p_{\min}^{single}+\sum_{l=2}^{m}(p_{\min}^{single}/l^2)\ (l=2, 3, \ldots, m) \quad (12)$$

$$\begin{aligned}\Delta C &= C_{\max}^{single}(x)-C_{\max}(x)=p_n^{single}-\frac{2}{3}p_{\min}^{single}-\sum_{l=2}^{m}(p_{\min}^{single}/l^2)\\ &\geq \frac{2}{3}p_{\min}^{single}-\sum_{l=2}^{m}(p_{\min}^{single}/l^2)=p_{\min}^{single}(\frac{2}{3}-\sum_{l=2}^{m}\frac{1}{l^2})\end{aligned} \quad (13)$$

Define $\varsigma(\alpha)=\sum_{l=1}^{\infty}\frac{1}{l^\alpha}$ as an infinite series with parameter α, here we have $\alpha=2$.

$$\varsigma(2)=\sum_{l=1}^{\infty}\frac{1}{l^2}=1+\frac{1}{2^2}+\frac{1}{3^2}+\ldots\frac{1}{l^2}+\ldots \quad (14)$$

It is well known that $\varsigma(2)$ is converges to $\frac{\pi^2}{6}$. Adjustment Eq. (14), we have

$$\sum_{l=2}^{\infty}\frac{1}{l^2}=\varsigma(2)-1=\frac{\pi^2}{6}-1 \quad (15)$$

For ΔC, we consider an extreme condition $m\to\infty$, then substitute Eq. (15) into Eq. (13), we can obtain

$$\Delta C \geq p_{\min}^{single}(\frac{2}{3}-\sum_{l=2}^{m}\frac{1}{l^2})=p_{\min}^{single}(\frac{2}{3}-\frac{\pi^2}{6}+1)>0 \quad (16)$$

Obviously, it is easily verified that there is a solution x_1 for *Ins*(1) such that $C_{\max}^{single}(\pi_1)\leq\Delta C$ if and only if there exists a solution π_2 for *Ins*(2) such that $C_{\max}(\pi_2)<\Delta C$. Here π_2

can be set as x_1 or other permutations satisfying $C_{\max}(x_2) < \Delta C$. Based on the reduction concept of computational complexity theory [13, 14], a polynomial Turing reduction (denoted by $\rightarrow$) can be established as follows:

$$1|\,|C_{max} \rightarrow P_m|no\text{-}wait\,|C_{\max} \quad (17)$$

Since $1|\,|C_{\max}$ is strongly NP-hard, $P_m|no\text{-}wait|C_{\max}$ is also NP-hard in the strong sense. **Theorem 1** holds. According to the complexity hierarchies of deterministic scheduling problems (see Fig. 3.4 in [14]), a network of reductions from the simplest problem ($1|\,|C_{\max}$) to SCCSP is constructed in Fig. 3. It can be seen that SCCSP is the most complex one which can be transformed into other known NP-hard problems in polynomial time by deleting some of its constraints from Fig. 3, it also implies the strong NP-hardness of SCCSP.

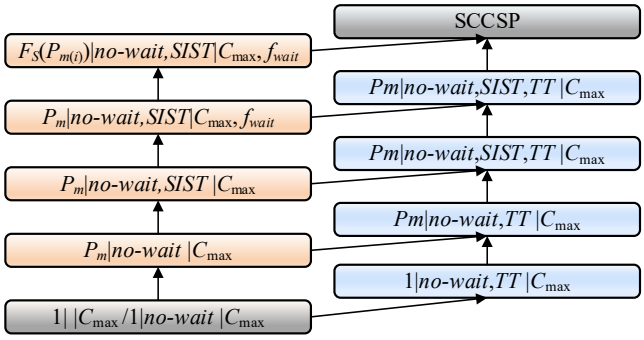


Fig. 3. Basic reductions network of SCCSP.

Part 4. Analyses of the CC of HierC_Q in Section IV

According to Algorithm 6, the computational complexity (CC) of the primary parts of the proposed HierC_Q is as follows. At the initialization phase, the LPT rule and Algorithm 4 are employed to generate an individual, and the permutation-based model is used to calculate its target value. Since the CC of this process is $O(N\log N+NM)$. In Algorithm 1, the CCs of Lines 8-19 and Lines 6-24 are nearly $O(N^2)$ and $O(Ep_{charge}N^2)$ respectively. Note that if without the validity evaluation method in Subsection IV-C-1), the complexity of Lines 8-19 will increase to $O(MN^3)$. Hence, the CC of Algorithm 1 is nearly $O(Ep_{charge}N^2)$. Similarly, the CC of Algorithm 2 is nearly $O(Ep_{cast}Z^2)$, since the speed-up evaluation method in Subsection IV-C-2) can reduce the CC of evaluating solution from $O(MZ^3)$ to $O(Z^2)$. The CC of Algorithm 3 is nearly $O(Ep_{joint}MZ^3)$, because the validity evaluation method and the speed-up evaluation method do not apply here when the two subsequences are improved simultaneous. The CC of Algorithm 5 is $O(NM)$. Let $TC_{\text{HierC\_Q}}$ denote the total CC of HierC_Q, *maxgen* is the maximum iterations of the framework. Then $TC_{\text{HierC\_Q}}$ can be expressed as:

$$TC_{\text{HierC\_Q}}=O(maxgen\times(N\log N+2NM+Ep_{charge}N^2 + Ep_{cast}Z^2+Ep_{joint}MZ^3) \quad (18)$$

From the above analyses, it can be found that the time complexity of HierC_Q is not high since the highest degree of the polynomial of $TC_{\text{HierC\_Q}}$ is two. It indicates that HierC_Q can reach more promising regions and obtain satisfactory solutions within a short time whether the considered problem in this paper or other typical production scheduling problems in real industry scenarios.

Part 5. Extended Version of Section V

Part 5 provides extensive test results and analyses of Section V, in which two new Subsection (i.e., Subsections V-B and V-C) are added. Subsection IV-B calibrates HierC_Q's parameters by using Design of Experiment (DOE) technique. Subsection V-D verifies the contributions of each key component of HierC_Q. In addition, Subsections IV-G, and IV-E provide more detailed test results. These two Subsections correspond to Subsections IV-B and IV-C in the submitted paper.

V Computational Comparisons and Statistical Analyses

In this section, numerical simulations and experiments are conducted by using some randomly produced instances with different scales to test and evaluate the superiority and efficacy of HierC_Q. Firstly, a detailed experimental setup is introduced in Section V-A. Secondly, the effect of HierC_Q's hyper-parameters is discussed in Section V-B. Thirdly, comparative experiment of six variants are designed to demonstrate the contributions of each key component of HierC_Q in Section V-C. Moreover, a comprehensive computational campaign of HierC_Q against eleven local search frameworks are investigated in Section V-D. Finally, computational comparisons and statistical analysis for HierC_Q against several state-of-the-art methods and a real-world heuristic method are conducted and discussed in Section V-E.

A. Experimental Setup

In this subsection, we elaborate the preparations before the computational experiments are performed to evaluate and test the HierC_Q framework, including program running environment, test data set, and evaluation index.

This framework is implemented by Delphi 2010 language and the experiments were executed on a PC server with processor Intel i7 - 8550U, 1.8 GHz, 32 GB DDR4 RAM, and operational system Windows 10 Professional Premium. It is worth emphasizing that the experiments introduced here were carried out in a common notebook, with a single CPU, showing the availability of our framework.

At present, there are no existing standard benchmarks of SCCSP so far, hence, several randomly generated problem instances with various scales are generated by using the international mainstream test data generation method which derives from the real-world scenario in the steelmaking complexes from Yunnan province (randomly generating the number of workpieces, machines and processing time within a certain range), that can objectively evaluate the performance of HierC_Q framework for SCCSP. That is, twenty different combinations of $\{S\times Z\}$in $S\in\{3,4,5,6\}$ and $Z\in\{10,15,20,25,30\}$ are considered. The processing time $p_{j\times k}$ and the setup time q_j are randomly generated from a uniform distribution in the range [36,50] and a uniform distribution in the range [80,100] respectively. Other production data for the instances are generated by a uniform distribution in the following range: $m_i\in[3,5]$, $\alpha_j\in[8,12]$, $t_i\in[10,15]$. All of the testing instances and the supplementary material can be downloaded from the website (i.e., https://github.com/ly726564418/SCCS.git). To facilitate a fair comparison, the maximum elapsed CPU time $T_{total}>Z\times S\times\lambda$ (milliseconds) have been consolidated as the

stopping criterion for each program according to the reference [4], where λ is the runtime factor. Considering the problem scale, each compared algorithm is executed 30 times independently.

To verify the efficiency and effectiveness of the compared algorithm, average relative percentage deviation (*ARPD*) is used as the response variable to evaluate the experimental results. *ARPD* of solution π_i is described as follows:

$$ARPD(\pi) = \frac{1}{R}\sum_{i=1}^{R}\left(\frac{f_i(\pi) - f_{best}(\pi)}{f_{best}(\pi)}\right)\times 100\% \qquad (19)$$

where $f_i(\pi)$ is the response variable of solution π yielded by a specific algorithm in the i-th experiment for each instance. $f_{best}(\pi)$ is the best solution produced by all of the compared algorithms in the same test scenario. The *ARPD* is widely favorable for measuring the gap between the solution obtained by a performance-unknown algorithm and the well-known solution found up to present[15]. Obviously, the smaller the *ARPD* value, the better the algorithm performance.

B. Parameter calibration

It's known that the appropriate parameter calibration has a conspicuous effect on the performance of the algorithm, in terms of computational efficiency and solution quality. In this subsection, the Design of Experiment (DOE) technique is calibrated utilizing to decide desirable parameter configurations of HierC_Q. According to the procedure of HierC_Q described in Subsection IV.E, four critical parameters are introduced., i.e., the episode period of charge independent learning model Ep_{charge}, the episode period of cast independent learning model Ep_{cast}, the episode period of synergy learning model Ep_{joint}, and the iteration number γ. After the preliminary experiments, four potential levels of the four factors or parameters are explicitly listed in Table III, and the orthogonal array L16($4^{\wedge}4$) is selected for those four parameters. A group of medium-sized instances (5×15) are selected for experimental analysis, and each configuration is conducted with 30 independent replications with a predefined elapsed CPU time $Z\times S\times 300$ (milliseconds). The orthogonal array and the obtained results are given in Table IV. According to Table IV, the values of average *ARPD* for all factor levels are figured out to analyze the significance rank, where the *Delta* value and the significance rank are reported in Table V. To investigate the effects of the key factors for different scale cases, the trend plots of each factor are illustrated in Fig. 4.

Table III
COMBINATIONS OF PARAMETER VALUES FOR TSOA.

Parameter	parameter level			
	1	2	3	4
Ep_{charge}	10	15	20	25
Ep_{cast}	5	10	15	20
Ep_{joint}	5	10	15	20
γ	1	2	3	4

According to Table V and Fig. 4, the parameter Ep_{charge} has the largest *Delta* values, indicating that the episode period of *Charge_QLSF* has the most important significant impact on the performance of HierC_Q, followed by Ep_{joint} and Ep_{cast}, and γ of the least significance. Generally, an iterative improvement process with an appropriate episode period size is conducive to making the algorithm more comprehensive in the search of feasible solution space. However, setting the episode period too small may result in insufficient search and difficulty to reach enough regions, while the excessive value of the episode period may greatly consume a lot of computational efforts and deteriorate the search efficiency. Although the iteration number γ has the smallest impact on the performance of the algorithm, a suitable γ is still conducive to the effective switching of the algorithm between D2R and L2I, ensuring the search vitality of the algorithm. According to the above analysis, a reasonable configuration of all parameters is suggested as follows: Ep_{charge}=15, Ep_{cast}=10, Ep_{joint}=15, γ=2.

Table IV
ORTHOGONAL ARRAY AND RV VALUES.

Combination	Parameter level				*ADPR*
	Ep_{charge}	Ep_{cast}	Ep_{joint}	γ	
1	1	1	1	1	0.299
2	1	2	2	2	0.224
3	1	3	3	3	0.263
4	1	4	4	4	0.310
5	2	1	2	3	0.277
6	2	2	1	4	0.246
7	2	3	4	1	0.245
8	2	4	3	2	0.209
9	3	1	3	4	0.254
10	3	2	4	3	0.264
11	3	3	1	2	0.292
12	3	4	2	1	0.288
13	4	1	4	2	0.296
14	4	2	3	1	0.252
15	4	3	2	4	0.306
16	4	4	1	3	0.319

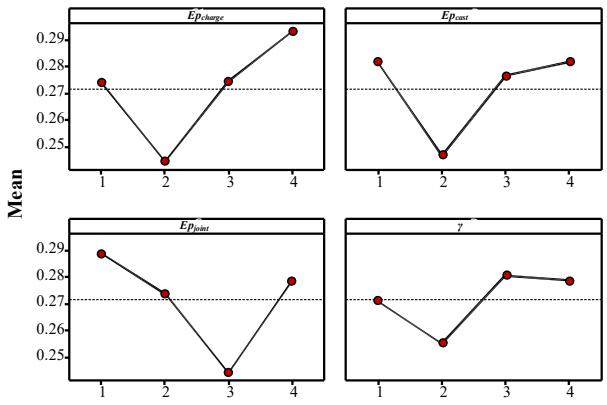


Fig. 4. The factor response trends.

Table V
PERFORMANCE RESPONSE VALUE OF EACH PARAMETER COMBINATION

Lever	Ep_{charge}	Ep_{cast}	Ep_{joint}	γ
1	0.274	0.281	0.289	0.271
2	**0.244**	**0.246**	0.273	**0.255**
3	0.275	0.276	**0.244**	0.281
4	0.293	0.282	0.279	0.279
Delta	0.049	0.035	0.045	0.026
Rank	1	3	2	4

C. Performance analysis of key components in HierC_Q

In this subsection, the influence of key components is investigated to demonstrate their contributions to HierC_Q. There are five main components, i.e., the CM-based reward function in Subsection IV-B-1), the Change_QLSF in Subsection IV-C-1), the Cast_QLSF in Subsection IV-C-2), the SQLSF in Subsection IV-C-3) and the perturbation-and-construction-based solution renewal strategy in Section IV-D. For this purpose, some variants of HierC_Q are implemented to verify the contributions of these components. That is, HierC_Q$_{v1}$ only adopt the fitness based reword function as

shown in the Eq. (20). HierC_Q$_{v2}$ does not adopt the Change_QLSF. HierC_Q$_{v3}$ does not employ the Cast_QLSF. HierC_Q$_{v4}$ does not used the SQLSF. HierC_Q$_{v5}$ without the perturbation-and-construction-based solution renewal strategy. Besides, we also explored a population-oriented learning frame in HierC_Q$_{v6}$, specifically, the starting point of HierC_Q$_{v6}$ is not a single solution, but a population.

$$r(s,a)=\begin{cases}1 & \Delta f(\pi)<0\\ 0 & otherwise\end{cases} \qquad (20)$$

In HierC_Q and its six variants, the individual initialization and updating strategies are absolutely the same, and the parameter configuration of above seven algorithms are also duplicate. All variants are independently run 30 times on each instance with a termination criterion of $Z\times S\times\lambda$ milliseconds of elapsed CPU time. The experimental results under the running time factor λ=200, 300, and 400 are given in Table VI-VIII.

It is clear from Table VI-VIII and Fig. 8 that the *ARPD* and *SD* values obtained by HierC_Q are obviously better than those obtained by its six variants for almost instances. Among them, the results obtained by HierC_Q$_{v5}$ are clearly the worst in terms of *ARPD*, which indicates that the D2R tier with the perturbation-and-construction-based solution renewal strategy has significant contributions to improving the performance of HierC_Q. The D2R tier updates the candidate solution by randomly selecting a disturbance operator to destroy the candidate solution partially or reconstruct a new solution for a new improvement iteration, which can effectively avoid the framework from falling into local optimum. The results obtained by HierC_Q$_{v4}$ are the second-worst, this is because the population-oriented learning frame means the improvement operators need to be performed for each individual in the population, which will cause substantial computational complexity. The results yielded by HierC_Q$_{v1}$ are the third worst, which implies the effectiveness of the reward function proposed in this paper. Furthermore, comparing with the results yielded by HierC_Q$_{v2}$, HierC_Q$_{v3}$, and HierC_Q$_{v4}$, it is clear that HierC_Q$_{v2}$ achieves the worst results, which reveals that the Charge_QSLF largely affects the performance of HierC_Q among the three QLSFs.

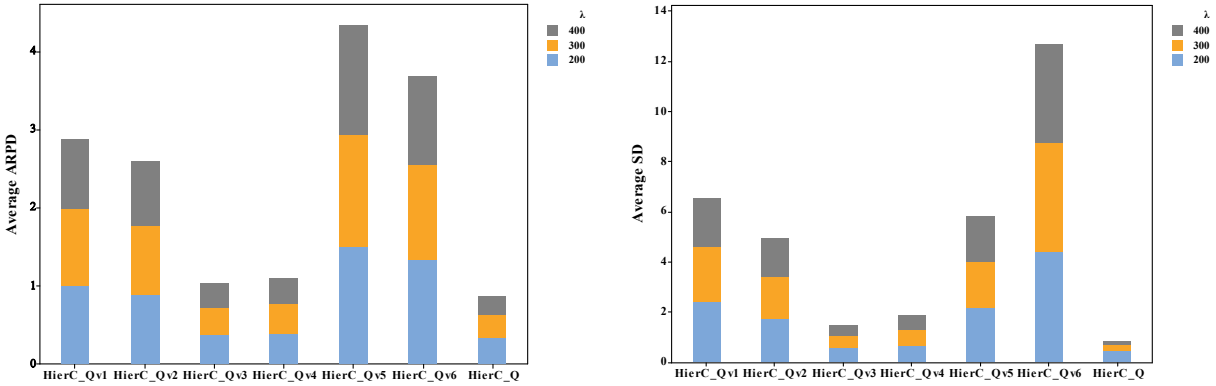


Fig. 5. Comparisons of HierC_Q's performance of the key components.

Table VI
COMPARISON RESULTS OF HIERC_Q WITH ITS SIX VARIANTS UNDER λ=200.

Instance	λ	HierC_Q$_{v1}$		HierC_Q$_{v2}$		HierC_Q$_{v3}$		HierC_Q$_{v4}$		HierC_Q$_{v5}$		HierC_Q$_{v6}$		HierC_Q	
$S\times Z$		*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*
3×10		1.096	0.813	0.830	0.461	**0.612**	*0.407*	*0.645*	**0.229**	0.809	**0.313**	1.169	1.201	**0.522**	0.696
3×15		0.549	1.228	0.473	0.522	*0.281*	0.551	**0.272**	*0.472*	0.359	**0.253**	0.516	1.050	**0.271**	**0.204**
3×20		1.123	4.028	0.912	2.172	**0.429**	**0.184**	*0.452*	*0.505*	0.735	1.960	1.498	5.970	**0.358**	**0.146**
3×25		1.820	6.742	1.177	3.927	**0.477**	*1.016*	*0.483*	**0.881**	4.156	6.293	2.566	8.729	**0.409**	**0.376**
3×30		1.659	1.069	1.577	**0.097**	*0.587*	**0.394**	**0.585**	0.596	1.532	*0.424*	1.668	1.254	**0.496**	0.606
4×10		1.124	1.860	1.500	3.285	**0.523**	**0.294**	*0.609*	*0.325*	1.655	7.384	1.617	4.556	**0.500**	**0.101**
4×15		0.590	**0.370**	0.529	**0.398**	**0.330**	0.601	*0.385*	0.724	0.595	0.708	0.519	*0.460*	**0.367**	0.479
4×20		0.496	5.444	0.260	4.408	**0.133**	**0.749**	*0.159*	0.909	0.320	*1.821*	0.908	7.105	**0.125**	**0.479**
4×25		1.982	5.841	1.338	2.777	**0.550**	**0.201**	*0.555*	**0.432**	4.634	4.807	2.364	5.649	**0.480**	*0.437*
4×30		1.303	0.740	0.867	1.365	*0.663*	0.745	**0.610**	**0.084**	1.134	0.460	1.419	*0.374*	**0.489**	**0.158**
5×10	200	0.708	1.834	0.383	**0.000**	**0.297**	*0.465*	*0.321*	0.519	0.681	1.068	0.830	2.223	**0.246**	**0.339**
5×15		0.805	2.035	1.274	1.225	**0.269**	0.954	*0.283*	*0.801*	0.658	**0.746**	1.214	4.907	**0.206**	**0.568**
5×20		2.247	5.127	1.367	2.875	**0.575**	**0.517**	*0.583*	0.937	5.145	*0.604*	3.646	4.894	**0.423**	**0.236**
5×25		0.482	1.016	0.537	*0.972*	**0.188**	**0.654**	*0.212*	1.539	0.418	1.243	0.764	14.022	**0.154**	**0.350**
5×30		0.442	*0.954*	0.445	**0.000**	**0.307**	2.067	*0.314*	2.663	0.309	**0.861**	0.391	5.842	**0.304**	1.637
6×10		0.564	2.093	1.424	4.885	**0.245**	**0.385**	*0.281*	**0.296**	0.850	9.020	1.242	7.569	**0.220**	*0.421*
6×15		0.809	2.356	1.254	1.464	**0.291**	*0.293*	*0.296*	**0.181**	0.741	0.895	1.062	2.519	**0.230**	**0.170**
6×20		0.636	1.477	0.352	1.752	**0.256**	**0.342**	*0.270*	*0.417*	0.867	**0.208**	0.678	1.560	**0.248**	1.266
6×25		1.425	2.627	0.800	1.788	**0.514**	**0.367**	*0.545*	*0.545*	4.247	3.924	2.160	7.868	**0.481**	**0.164**
6×30		0.363	0.977	0.605	0.419	**0.140**	**0.104**	*0.179*	0.640	0.134	*0.415*	0.645	1.075	**0.127**	**0.182**
Average		1.011	2.431	0.895	1.740	**0.383**	**0.565**	*0.402*	*0.685*	1.499	2.170	1.344	4.441	**0.333**	**0.451**

Table VII
COMPARISON RESULTS OF HIERC_Q WITH ITS SIX VARIANTS UNDER λ=300.

Instance	λ	HierC_NS$_{v1}$		HierC_NS$_{v2}$		HierC_NS$_{v3}$		HierC_NS$_{v4}$		HierC_NS$_{v5}$		HierC_NS$_{v6}$		HierC_NS	
$S\times Z$		*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*
3×10		1.035	0.462	0.759	0.355	**0.573**	**0.338**	*0.576*	0.409	0.751	*0.354*	1.092	1.194	**0.530**	**0.323**
3×15	300	0.545	1.661	0.446	1.145	**0.290**	*0.538*	*0.292*	0.846	0.352	**0.285**	0.428	1.227	**0.280**	**0.317**
3×20		1.101	2.641	0.874	1.937	**0.385**	*0.338*	*0.431*	**0.229**	0.679	1.770	1.324	4.838	**0.367**	**0.076**
3×25		1.842	4.240	1.159	4.694	**0.414**	**0.529**	*0.466*	**0.672**	4.087	4.578	2.116	8.495	**0.314**	*0.730*

3×30	1.419	1.761	1.566	0.323	**0.560**	0.351	*0.567*	*0.180*	1.477	**0.054**	1.643	0.841	**0.406**	**0.089**
4×10	1.068	1.441	1.466	3.356	**0.505**	*0.497*	*0.527*	**0.325**	1.336	4.427	1.578	3.967	**0.463**	**0.137**
4×15	0.588	**0.267**	0.521	**0.329**	**0.325**	0.481	*0.329*	0.694	0.590	0.655	0.501	*0.476*	**0.367**	0.483
4×20	0.488	3.123	0.243	2.007	**0.132**	**0.746**	*0.154*	*0.758*	0.274	1.330	0.909	8.645	**0.104**	**0.069**
4×25	1.874	4.067	1.194	3.051	**0.433**	**0.035**	*0.478*	*0.355*	4.439	3.664	2.135	5.802	**0.388**	**0.144**
4×30	1.302	0.495	0.814	0.728	**0.547**	0.441	*0.588*	*0.236*	1.055	0.691	1.400	**0.126**	**0.428**	**0.107**
5×10	0.767	1.816	0.400	**0.377**	**0.266**	*0.534*	*0.296*	**0.166**	0.664	0.949	0.733	2.911	**0.255**	0.678
5×15	0.800	2.712	1.271	1.438	**0.242**	0.961	*0.272*	*0.732*	0.643	**0.000**	1.167	4.504	**0.152**	**0.367**
5×20	2.233	5.926	1.364	2.476	**0.505**	**0.118**	*0.535*	*0.588*	4.962	1.709	3.104	7.356	**0.383**	**0.151**
5×25	0.431	1.986	0.537	0.775	**0.161**	**0.414**	*0.163*	*0.575*	0.429	0.784	0.739	12.486	**0.127**	**0.453**
5×30	0.431	1.104	0.443	**0.435**	*0.314*	2.842	**0.298**	2.538	0.303	**0.667**	0.371	4.703	**0.287**	*0.732*
6×10	0.614	1.865	1.382	5.023	*0.220*	**0.113**	**0.212**	*0.484*	0.854	8.922	0.944	7.065	**0.161**	**0.051**
6×15	0.750	1.819	1.250	1.272	**0.260**	**0.032**	*0.285*	*0.423*	0.736	0.604	**1.136**	2.830	0.169	**0.023**
6×20	0.600	1.100	0.303	0.948	**0.229**	*0.450*	*0.291*	1.028	0.874	**0.414**	0.617	1.962	**0.178**	**0.000**
6×25	1.347	3.295	0.784	1.629	**0.441**	**0.355**	*0.495*	*0.815*	4.185	4.135	1.564	3.724	**0.354**	**0.017**
6×30	0.317	1.183	0.606	*0.346*	*0.157*	**0.084**	**0.115**	0.353	0.106	0.652	0.652	2.633	**0.099**	**0.250**
Average	0.978	2.148	0.869	1.632	**0.348**	**0.510**	*0.369*	*0.620*	1.440	1.832	1.208	4.289	**0.291**	**0.260**

Table VIII
COMPARISON RESULTS OF HIERC_Q WITH ITS SIX VARIANTS UNDER λ=400.

Instance	λ	HierC_NS$_{v1}$		HierC_NS$_{v2}$		HierC_NS$_{v3}$		HierC_NS$_{v4}$		HierC_NS$_{v5}$		HierC_NS$_{v6}$		HierC_NS	
S×*Z*		*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*
3×10		1.024	0.710	0.755	0.575	**0.488**	*0.536*	*0.577*	0.565	0.699	**0.361**	1.031	0.712	**0.454**	**0.000**
3×15		0.482	1.022	0.428	1.213	**0.284**	*0.287*	*0.285*	0.557	0.337	**0.003**	0.433	0.973	**0.240**	**0.000**
3×20		0.969	2.941	0.847	2.421	**0.340**	**0.064**	*0.415*	*0.291*	0.689	1.903	1.238	4.300	**0.284**	**0.000**
3×25		1.677	4.839	1.133	3.494	*0.388*	**0.446**	**0.381**	*0.550*	3.944	4.064	1.908	6.418	**0.249**	**0.000**
3×30		1.355	1.214	1.531	0.659	*0.502*	**0.249**	**0.465**	*0.431*	1.474	0.822	1.645	0.901	**0.413**	**0.000**
4×10		1.062	1.487	1.342	2.937	**0.459**	**0.000**	*0.531*	**0.100**	1.356	3.850	1.455	3.961	**0.393**	*0.239*
4×15		0.534	0.588	0.455	1.006	**0.297**	**0.363**	**0.242**	0.776	0.602	0.853	0.457	*0.576*	*0.327*	**0.000**
4×20		0.437	2.434	0.212	**0.394**	**0.131**	0.683	*0.146*	0.678	0.252	*0.516*	0.761	7.533	**0.070**	**0.000**
4×25		1.522	3.515	1.146	2.690	**0.370**	**0.031**	*0.438*	*0.380*	4.286	4.080	2.053	5.145	**0.294**	**0.000**
4×30		1.249	0.478	0.786	0.919	*0.513*	0.358	**0.503**	*0.288*	1.028	0.647	1.379	**0.231**	**0.344**	**0.000**
5×10	400	0.621	0.862	0.393	0.800	**0.260**	**0.430**	*0.273*	*0.599*	0.635	0.656	0.695	2.258	**0.286**	**0.351**
5×15		0.713	3.052	1.264	1.535	**0.219**	**0.390**	*0.259*	*0.969*	0.635	1.029	1.081	4.822	**0.154**	**0.241**
5×20		2.064	3.440	1.359	2.804	*0.495*	*0.535*	**0.493**	**0.273**	4.889	1.696	3.106	4.535	**0.283**	**0.000**
5×25		0.403	2.078	0.536	0.542	**0.128**	**0.259**	*0.167*	*0.530*	0.440	0.968	0.718	10.038	**0.117**	**0.000**
5×30		0.429	2.456	0.444	**0.195**	*0.298*	*1.470*	**0.275**	3.459	0.313	**1.349**	0.376	5.216	**0.259**	1.663
6×10		0.527	2.342	1.307	5.292	**0.190**	*0.395*	*0.199*	**0.049**	0.784	8.455	0.828	7.167	**0.136**	**0.000**
6×15		0.668	1.510	1.240	0.993	**0.213**	0.316	*0.229*	**0.084**	0.648	*0.288*	1.037	5.163	**0.152**	**0.000**
6×20		0.553	1.664	0.301	1.604	0.218	0.657	**0.185**	*0.219*	0.831	**0.186**	0.572	1.276	**0.150**	**0.040**
6×25		1.092	1.846	0.765	1.278	**0.358**	*0.236*	*0.407*	**0.159**	4.131	3.794	1.486	5.351	**0.239**	**0.000**
6×30		0.295	1.167	0.594	**0.000**	0.114	0.296	*0.106*	**0.194**	**0.080**	0.455	0.630	2.306	**0.089**	*0.211*
Average		0.884	1.982	0.842	1.567	**0.313**	0.400	*0.329*	0.557	1.403	1.799	1.145	3.944	**0.247**	**0.137**

D. Comparison of HierC_Q with eleven neighborhood search based heuristic frameworks

In this subsection, we provide the comparison results of HierC_Q with several LSFs and their variants to verify the effectiveness of the proposed framework and developed improvement operators.

We consider five LSFs, i.e., variable neighborhood search (VNS) [16], iterated local search (ILS) [17], great deluge Algorithm [18], adaptive large neighborhood search (ALNS) [19], and hyper-heuristic genetic algorithm (HHGA) [20]. The core idea behindVNS, LNS, and GD is to first perturb the current best solution to jump out of the local optimum, and then perform the iterated neighborhood local search to find more high-quality solutions. The perturbation strength has to be sufficient to pull the search trajectory to a different attraction basin leading to promising regions and to drive the algorithm away from the local optimum. ALNS is an effective local search method proposed for routing and delivery problems [21-23]. By measuring the effect of each operator, ALNS can automatically select a good operator to destroy or repair the current solution to find a better solution. Hyper-heuristic algorithm [24] is a new type of intelligent optimization framework. This framework manipulates or manages low-level heuristics (LLHs) through a high-level strategy (HLS), so as to search different regions of solution space. Different techniques such as case-based reasoning, local search, and genetic programming are used for HHA to select low-level heuristics. In this paper, the genetic algorithm is employed as a high-level strategy.

Noted that all the above frameworks use PDIOs and perturbation operators covered in this paper. Besides, in the scheduling literature, *Swap*, *Insert* and *Exchange* are commonly used neighborhood operators for the permutation-based representation. Therefore, it is necessary to make a further comparison of the performance of the proposed PDNOs in Subsection IV-A. For this purpose, some variants (i.e., VNS*var*, ILS*var*, GD*var*, ALNS*var*, HHGA*var*, and HierC_Q*var*) adopted those three classical neighborhood operators implemented to verify the effectiveness of our proposed operations. All steps of these compared algorithms are strictly implemented according to the literature, and the speed-up evaluation methods presented in this paper are utilized to accelerate the search process. Moreover, the parameters of these compared algorithms are directly taken from the original literature. The pseudo-codes of these frameworks for solving SCCSP and descriptions of specifics are reported in Part 6 which has been uploaded to the above website. All frameworks are independently run 30 times on each instance and Z×S×λ milliseconds elapsed CPU time is adopted as the termination criterion. The statistical results under the running time factor λ=200, 300, and 400 are obtained and these results under λ=200 and 400 are reported in Table IX. From this table, it can be easily seen that the *ARPD* values of HierC_Q under each λ are better than those of the other compared frameworks under the same λ for all

instances, and the *ARPD* values of HierC_Q under λ=200 are better than all the *ARPD* values of the other compared algorithms under λ=400, from which concluded that HierC_Q is the one with the best performance.

To further analyze the differences in performance among the compared algorithms, all results are analyzed using multifactor ANOVA. The ANOVA is a mainstream method to evaluate whether or not there are substantial differences in the values obtained by all of the compared algorithms. It is important to note that three main hypotheses—normality, homoscedasticity, and residual independence—are tested in statistical trials with a 95% confidence level. All hypotheses are easily satisfied based on an analysis of the residuals from the experimental results. Fig. 6 and Fig.7 display two group means plots with 95% Tukey's HSD confidence intervals for the compared algorithms under λ=200, 300, and 400. It can be seen from Fig. 6 that HierC_Q is statistically superior to all the other algorithms under the same λ with an obvious margin whether with the traditional improvement operator (i.e., *Swap*, *Insert*, and *Exchange*) or the operators proposed in this paper. This shows that HierC_Q has better performance than the existing LSFs. Furthermore, it's obvious through each subgraph of Fig.7 that under the same λ, the results obtained by each LSF with PDNOs are statistically outperformed those with the traditional operators by a considerable margin. Hence, we can safely conclude that PDNO has a powerful search engine for tackling the fitness of SCCSP.

Table IX
COMPARISON RESULTS OF HIERC_Q WITH ELEVEN HF_NSS UNDER λ=200 AND 400.

$S\times Z$	λ	VNS_{var}	VNS	ILS_{var}	ILS	GD_{var}	GD	$ALNS_{var}$	ALNS	$HHGA_{var}$	HHGA	$HierC\_Q_{var}$	HierC_Q
3×10		0.755	0.758	0.727	0.799	0.721	0.753	0.818	0.81	1.678	**0.588**	*0.641*	**0.522**
3×15		*0.131*	**0.124**	0.135	0.152	**0.125**	0.154	0.173	0.168	1.327	0.302	0.304	0.271
3×20		0.834	0.820	0.862	0.699	0.677	0.606	0.662	0.773	*0.546*	0.584	**0.493**	**0.358**
3×25		1.166	1.044	1.182	0.993	0.874	0.950	1.288	0.919	**0.457**	0.767	*0.536*	**0.409**
3×30		1.596	1.481	1.614	1.610	1.620	1.632	1.615	1.632	0.187	*0.926*	**0.765**	**0.496**
4×10		1.338	1.427	1.230	1.303	1.005	1.460	1.118	1.424	*0.690*	0.762	**0.668**	**0.500**
4×15		0.447	0.449	0.471	0.536	0.474	0.491	**0.089**	**0.084**	0.546	0.409	*0.343*	0.367
4×20		0.616	0.611	0.325	0.348	0.332	0.391	0.274	0.298	0.679	*0.209*	**0.174**	**0.125**
4×25		0.939	1.100	1.100	1.020	0.849	0.900	1.015	0.977	**0.580**	0.853	*0.686*	**0.480**
4×30		0.715	0.806	0.824	0.627	*0.516*	0.574	0.664	0.655	**0.416**	0.959	0.723	**0.489**
5×10	200	0.374	0.368	*0.326*	0.383	**0.273**	0.372	0.464	0.397	0.688	0.354	0.360	**0.246**
5×15		1.220	1.106	1.256	1.260	1.248	1.235	1.270	1.269	0.739	*0.387*	**0.390**	**0.206**
5×20		1.170	1.178	1.210	1.127	1.027	1.097	1.062	*0.931*	1.187	0.964	**0.754**	**0.423**
5×25		0.531	0.506	0.533	0.532	0.535	0.525	0.537	0.532	0.678	**0.222**	*0.240*	**0.154**
5×30		0.127	**0.120**	**0.121**	0.154	*0.126*	0.155	0.167	0.156	0.519	0.333	0.318	0.304
6×10		1.417	0.956	1.357	1.426	1.421	1.478	1.604	1.555	0.657	*0.365*	**0.309**	**0.220**
6×15		1.168	0.928	1.213	1.208	1.226	1.167	1.242	1.236	0.715	*0.412*	**0.345**	**0.230**
6×20		0.240	*0.222*	**0.217**	0.256	**0.214**	0.247	0.338	0.334	0.840	0.333	0.308	0.248
6×25		0.640	0.702	0.670	*0.612*	**0.581**	0.624	0.671	0.71	0.702	0.642	0.630	**0.481**
6×30		0.544	0.454	0.560	0.552	0.568	0.567	0.599	0.600	0.724	*0.205*	**0.197**	**0.127**
Average		0.798	0.758	0.797	0.780	0.721	0.769	0.783	0.773	0.728	*0.529*	**0.459**	**0.333**
3×10		0.754	0.756	0.713	0.773	0.704	0.760	0.789	0.750	1.626	**0.601**	*0.690*	**0.454**
3×15		0.116	**0.111**	**0.112**	0.137	*0.114*	0.125	0.144	0.142	1.336	0.335	0.286	0.240
3×20		0.789	0.772	0.776	0.675	*0.538*	0.633	0.607	0.637	0.563	0.567	**0.522**	**0.284**
3×25		1.299	1.116	1.176	1.081	0.897	1.073	0.998	0.785	**0.494**	0.687	*0.566*	**0.249**
3×30		1.558	1.586	1.626	1.512	1.548	1.563	1.629	1.632	**0.181**	0.926	*0.761*	**0.413**
4×10		1.255	1.365	1.067	1.260	1.098	1.147	1.232	1.036	0.717	*0.688*	**0.659**	**0.393**
4×15		0.444	0.434	0.451	0.042	0.456	**0.003**	**0.043**	*0.072*	0.548	0.460	0.410	0.327
4×20		0.526	0.443	0.450	0.433	0.338	0.325	0.266	0.311	0.692	**0.198**	*0.203*	**0.070**
4×25		1.222	1.263	1.147	0.974	0.870	0.861	0.824	0.867	0.497	0.705	0.688	0.294
4×30		0.634	0.695	0.689	*0.542*	0.564	0.597	0.636	0.567	**0.418**	0.899	0.703	**0.344**
5×10	400	0.326	0.320	*0.289*	0.356	**0.251**	0.319	0.416	0.362	0.783	0.393	0.366	**0.286**
5×15		1.170	1.036	1.232	1.210	1.226	1.236	1.275	1.262	0.782	*0.406*	**0.341**	**0.154**
5×20		1.363	1.133	1.186	1.032	**0.825**	1.014	0.886	0.972	1.343	0.914	*0.843*	**0.283**
5×25		0.532	0.472	0.535	0.532	0.533	0.535	0.532	0.537	0.711	*0.234*	**0.191**	**0.117**
5×30		**0.102**	**0.109**	*0.112*	0.124	0.115	0.128	0.130	0.148	0.516	0.354	0.350	0.259
6×10		1.389	0.905	1.419	1.403	1.298	1.486	1.549	1.514	0.698	0.395	0.292	**0.136**
6×15		1.125	0.962	1.199	1.225	1.214	1.228	1.246	1.254	0.695	**0.394**	**0.323**	**0.152**
6×20		0.183	*0.168*	0.179	**0.153**	0.171	0.178	0.254	0.233	0.840	0.352	0.296	**0.150**
6×25		0.717	0.552	0.640	**0.443**	0.485	*0.483*	0.580	0.603	0.718	0.690	0.648	**0.239**
6×30		0.601	0.489	0.541	0.554	0.574	0.575	0.600	0.6	0.746	*0.203*	**0.183**	**0.089**
Average		0.805	0.734	0.777	0.723	0.691	0.713	0.732	0.714	0.745	*0.520*	**0.466**	**0.247**

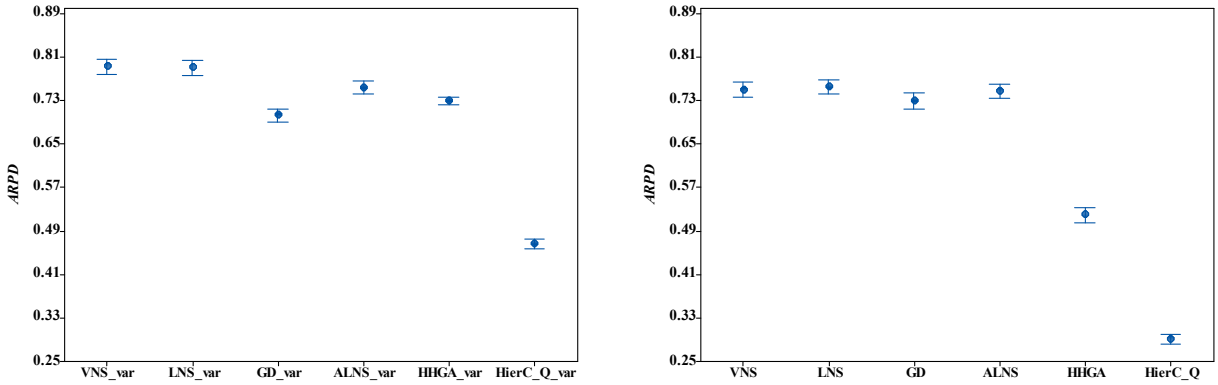


Fig. 6. Comparisons of HierC_Q with five HF_NSs.

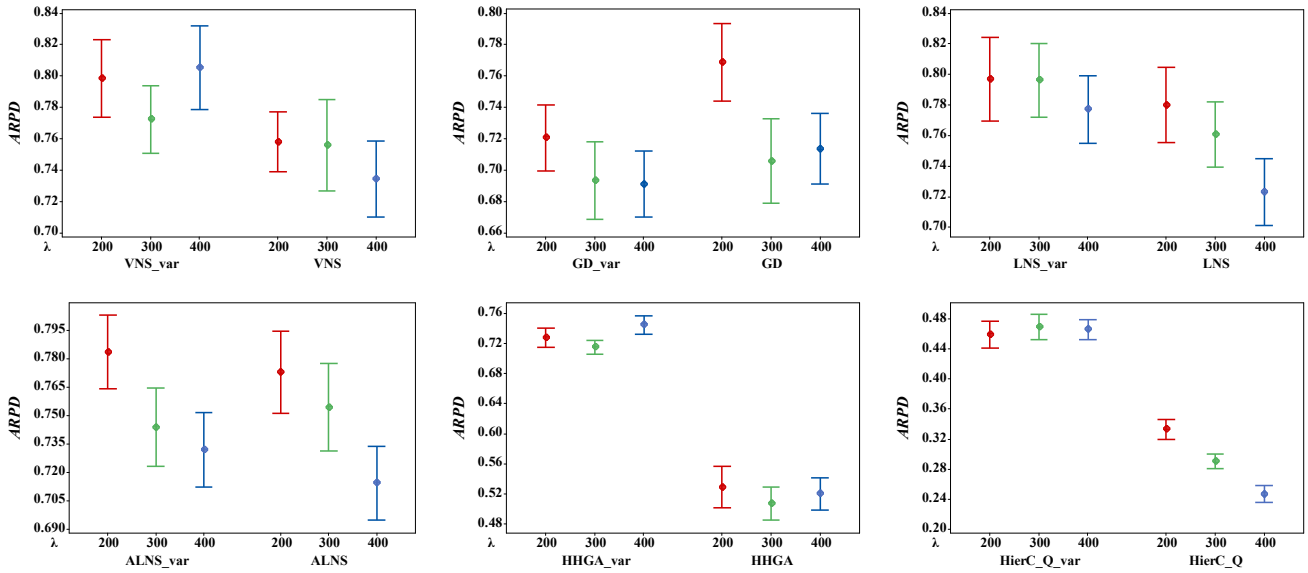


Fig. 7. Comparisons of the performance of the PDIOs.

E. Comparison HierC_Q with the state-of-the-art methods

To investigate the effectiveness of HierC_Q, this section aims to compare the performance of HierC_Q against several state-of-the-art algorithms published in recent literature. These algorithms are classified into three groups: The first group is an industrial heuristic (IDH) which is widely utilized in modern iron and steel companies. IDH handles the cast subproblem and the charge subproblem separately, with the former being solved first. For the cast subproblem, each cast is first arranged in decreasing order of processing time to produce a cast subsequence. Each cast in the cast subsequence must be picked in order from the back to the front and then assigned to the first casting machine available as quickly as practicable. In this manner, the partial scheduling scheme (i.e., the starting processing time and the processing machine of each cast) in the continuous casting stage is determined. Meanwhile, the start processing time of each charge in this stage is collected. For the charge subproblem, all charges are arranged in ascending order of their obtained start times in the continuous casting stage to produce a charge subsequence. Each charge is extracted from the charge subsequence from the end to the beginning and then assigned to an available refining machine or converter from the last refining stage to the steelmaking stage. In each of these stages, each charge is allocated to the first available machine in reverse order and launched as late as possible. So far, the full scheduling scheme has been determined. The second group include four algorithms, i.e., PIDE [25], HFOA [26], IABC [6], CCABC [4], these algorithms are designed to deal with multifarious SCCSPs; The third group include four algorithms, i.e., ISA[27], HGA[28], IPSO[29], and EIGA [30], where these algorithms are developed to solve the hybrid flowshop scheduling problems.

All algorithms are thoroughly re-implemented in line with the original literature and also suitably tailored to the problem under consideration. The pseudo-codes of these algorithms and special descriptions are supplied in Part 7 of the online supplementary material. Meanwhile, the validity evaluation method and speed-up evaluation method developed in Subsection IV-C is also incorporated into these re-implemented algorithms to improve search efficiency. The parameters of all compared algorithms are derived from the recommended settings in the literature. All algorithms are individually executed 30 times for each instance, with the identical termination requirements that the maximum elapsed CPU time is $Z\times S\times\lambda$ milliseconds. The comprehensive comparison results of all compared algorithms are reported in Table X-XII and Fig. 7. It can be seen that the *APRD* and *SD* values obtained by HierC_Q are obviously better than those obtained by the compared algorithms for all instances. Moreover, HierC_Q under λ=200 can obtain better results than the other compared algorithms under λ=400 for almost instances. The numerical results reflect that the

presented HierC_Q can effectively solve the SCCSP. In addition, the presented HierC_Q achieves the lowest overall average values of *ARPD* and *SD* on 20 different scale instances, indicating that the average performance of HierC_Q is better than the other nine compared algorithms.

To further analyze the differences in performance among the compared algorithms, all results are analyzed utilizing the analysis of ANOVA. The means plots and 95% of Tukey's HSD confidence interval graphs of all results obtained by HierC_Q and the 9 compared algorithms under different instance scales are shown in Fig. 6. It is worth noting that the overlap interval between any two algorithms indicates that there is no significant difference in their performance. From Fig. 6, it is evident that there is no overlap between HierC_Q and other compared algorithms in different scenarios, which means that the results obtained by HierC_Q are strongly and statistically significantly different from compared algorithms. Furthermore, to intuitively show the discreteness and symmetry of the obtained results, the box plots of all results based on different instances are given in Fig. 8 (a)-(f). Each of the box plots shows the distribution of the data given by different algorithms under λ=200, 300 and 400. It can be concluded from each of the subgraphs in Fig. 8 that: (1) the overall effect of HierC_Q is better than the other comparison algorithms regardless of the different values of λ, (2) with the increase in running time, the performance and discreteness of each algorithm are improved, especially the subgraphs (c), (e), and (f). Taken all subgraphs together, the results obtained by HierC_Q have the lowest degree of discreteness and higher quality of solutions than other comparison algorithms in all cases. This fully demonstrates better search performance and convergence ability of our proposed framework. In addition, the Gantt charts of all the instances are reported in Part 8.

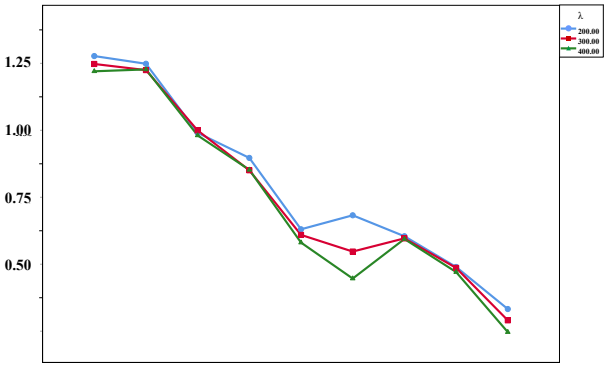


Fig. 6. Interaction plot with 95% Tukey's HSD confidence interval between algorithm.

TABLE X
COMPARISON RESULTS OF HIERC_Q WITH THE STATE-OF-THE-ART METHODS UNDER λ=200.

S×Z	IDH	ISA		PIDE		HGA		HFOA		IPSO		EIGA		IABC		CCABC		HierC_Q	
	ARPD	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*
3×10	2.319	1.479	0.478	1.513	0.463	2.361	*0.346*	0.966	0.419	**0.531**	**0.003**	0.898	0.696	0.694	0.443	*0.673*	**0.336**	**0.522**	0.522
3×15	1.372	0.417	**0.143**	0.417	*0.192*	1.479	**0.162**	0.331	0.311	0.300	0.767	**0.273**	0.347	0.278	0.692	*0.276*	0.766	**0.271**	0.306
3×20	3.427	1.644	5.243	1.482	5.623	0.698	**0.154**	1.100	6.256	**0.366**	0.448	0.651	3.436	**0.358**	*0.339*	*0.561*	1.104	**0.358**	**0.063**
3×25	9.849	2.221	5.804	2.044	6.217	*0.897*	2.876	1.600	4.963	1.799	**1.510**	1.731	3.285	1.701	1.893	**0.674**	*1.638*	**0.409**	**0.376**
3×30	3.352	1.582	*1.079*	1.547	1.729	0.632	1.701	0.994	6.712	0.551	**1.069**	1.129	2.419	0.510	**0.000**	0.812	4.868	**0.496**	3.242
4×10	7.032	1.382	2.680	1.321	1.781	0.921	**0.401**	0.818	2.687	0.517	0.740	*0.509*	*0.630*	**0.462**	1.157	0.643	0.733	**0.500**	**0.280**
4×15	0.874	0.636	**0.676**	0.623	*1.321*	0.674	**0.000**	0.494	1.355	0.380	1.689	*0.378*	1.335	0.380	1.530	**0.370**	2.117	**0.367**	1.778
4×20	2.513	0.504	1.233	0.507	1.646	0.789	*0.769*	0.372	1.712	**0.127**	**0.627**	0.224	1.289	*0.149*	1.523	0.193	0.799	**0.125**	**0.479**
4×25	10.508	2.450	5.022	2.395	3.789	*0.882*	1.543	1.786	3.368	1.896	**0.812**	1.682	3.813	1.630	1.855	**0.668**	*1.121*	**0.480**	**0.437**
4×30	2.218	1.384	1.385	1.410	0.663	**0.499**	*0.534*	0.996	3.424	0.888	**0.207**	1.164	3.180	*0.803*	**0.464**	0.733	2.850	**0.489**	1.620
5×10	4.327	1.323	2.612	1.302	2.136	1.162	1.126	0.678	2.806	**0.272**	1.022	*0.275*	0.923	0.298	0.963	0.342	0.509	**0.246**	**0.000**
5×15	3.017	0.827	1.396	0.873	1.705	1.007	0.432	0.501	1.299	*0.210*	*0.858*	**0.208**	0.979	0.215	**0.836**	0.434	1.562	**0.206**	**0.264**
5×20	7.058	3.386	3.752	3.185	4.170	*1.551*	*2.087*	2.841	3.520	1.925	2.790	1.690	3.655	1.978	2.418	**0.915**	**1.568**	**0.423**	**0.236**
5×25	2.365	0.477	1.435	0.480	**0.845**	0.796	*1.020*	0.402	1.326	**0.157**	1.343	0.275	1.692	*0.174*	1.354	0.247	1.303	**0.154**	**0.350**
5×30	0.729	*0.294*	1.895	**0.280**	2.315	0.617	**1.263**	**0.245**	1.914	0.324	2.413	0.313	2.665	0.307	2.924	0.331	*1.310*	0.304	**0.523**
6×10	3.279	0.786	1.631	0.840	2.227	0.864	0.627	0.366	2.547	*0.247*	1.117	0.297	0.774	**0.230**	**0.133**	0.358	**0.000**	**0.220**	*0.421*
6×15	3.011	0.782	0.989	0.765	1.848	1.045	*0.795*	0.512	0.984	0.283	0.853	**0.252**	**0.479**	*0.261*	1.426	0.413	1.482	**0.230**	**0.170**
6×20	1.477	0.924	1.372	0.923	*1.264*	1.195	**0.537**	0.637	1.596	**0.326**	1.298	0.437	1.509	0.361	1.780	*0.334*	**0.911**	**0.248**	1.266
6×25	8.692	2.507	6.451	2.529	6.430	*0.823*	*1.094*	1.956	3.707	1.378	2.639	1.030	3.074	1.175	1.766	**0.632**	**0.928**	**0.481**	**0.000**
6×30	1.773	0.540	1.986	0.533	3.362	0.921	2.777	0.344	2.586	**0.133**	**0.000**	0.239	2.691	**0.127**	**0.859**	*0.187*	3.080	**0.127**	*1.411*
Average	3.960	1.277	2.363	1.248	2.486	0.991	**1.012**	0.897	2.675	0.631	*1.110*	0.683	1.944	*0.605*	1.218	**0.490**	1.449	**0.333**	**0.687**

Table XI
COMPARISON RESULTS OF HIERC_Q WITH THE STATE-OF-THE-ART METHODS UNDER λ=300.

S×Z	IDH	ISA		PIDE		HGA		HFOA		IPSO		EIGA		IABC		CCABC		HierC_Q	
	ARPD	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*
3×10	2.319	1.406	0.822	1.469	0.880	2.340	**0.000**	0.926	0.643	**0.477**	0.685	0.790	*0.137*	*0.641*	0.218	*0.641*	**0.033**	**0.530**	0.187
3×15	1.372	0.417	0.416	0.407	**0.100**	1.481	**0.051**	0.319	*0.304*	**0.284**	1.169	*0.286*	0.824	0.291	0.915	0.317	0.802	**0.280**	0.429
3×20	3.427	1.724	5.082	1.718	5.881	0.715	**0.000**	0.899	4.441	**0.360**	1.467	*0.369*	0.961	0.373	1.186	0.568	*0.849*	**0.367**	**0.219**
3×25	9.849	1.991	4.660	2.008	4.849	*0.959*	2.807	1.546	4.685	1.780	*1.727*	1.477	4.276	1.744	2.142	**0.575**	**1.623**	**0.314**	**0.730**
3×30	3.352	1.549	1.657	1.577	**1.363**	*0.621*	1.500	0.906	5.630	0.615	**0.904**	0.987	3.823	**0.538**	*1.464*	0.870	4.077	**0.406**	1.878
4×10	7.032	1.341	2.087	1.322	2.042	0.867	**0.709**	0.757	2.561	**0.473**	1.716	0.480	1.064	*0.474*	1.492	0.667	*0.891*	**0.463**	**0.321**
4×15	0.874	0.645	**1.082**	0.568	*1.218*	**0.664**	**0.270**	**0.366**	1.412	0.381	2.189	0.415	1.725	*0.371*	1.828	0.374	1.544	**0.367**	1.731
4×20	2.513	0.497	1.440	0.481	1.938	0.821	*0.342*	0.357	1.415	*0.142*	0.635	**0.070**	**0.000**	0.164	0.788	0.222	0.909	**0.104**	**0.069**
4×25	10.508	2.496	4.736	2.069	3.143	*0.831*	*1.152*	1.683	2.700	1.877	1.357	1.356	3.894	1.772	1.941	**0.690**	**0.932**	**0.388**	**0.144**
4×30	2.218	1.378	1.138	1.379	1.392	**0.491**	**0.453**	0.984	3.587	0.910	**0.187**	0.914	3.919	0.760	*0.960*	*0.747*	3.215	**0.428**	1.505
5×10	4.327	1.305	2.004	1.385	2.551	1.216	*0.740*	0.652	2.259	0.312	1.017	**0.270**	0.970	*0.302*	1.296	0.372	**0.074**	**0.255**	**0.254**
5×15	3.017	0.835	1.950	0.858	1.707	1.045	0.601	0.491	1.614	**0.155**	0.742	*0.157*	**0.353**	0.179	*0.533*	0.362	1.597	**0.152**	**0.102**
5×20	7.058	3.267	4.593	3.239	3.422	1.596	*1.315*	2.788	2.762	1.858	2.022	*1.212*	3.729	1.777	2.490	**0.948**	**1.272**	**0.383**	**0.151**
5×25	2.365	0.473	1.480	0.496	*0.812*	0.811	0.946	0.362	2.405	**0.126**	1.602	0.198	**0.695**	*0.164*	1.393	0.213	1.185	**0.127**	**0.000**
5×30	0.729	*0.285*	2.026	0.287	2.078	0.626	**1.298**	**0.212**	1.935	0.288	2.775	0.318	*1.349*	**0.271**	3.462	0.299	1.888	0.287	**0.000**
6×10	3.279	0.783	2.206	0.813	1.750	0.925	*0.809*	0.390	3.424	**0.163**	1.082	*0.169*	**0.562**	0.179	1.275	0.354	3.088	**0.161**	**0.051**
6×15	3.011	0.764	1.384	0.770	1.448	1.038	*0.683*	0.467	1.393	**0.264**	0.952	0.310	**0.243**	*0.276*	1.067	0.371	1.746	**0.169**	**0.023**
6×20	1.477	0.880	1.470	0.922	1.350	1.211	*0.639*	0.594	1.469	**0.273**	0.821	**0.273**	0.820	0.396	1.844	*0.295*	**0.601**	**0.178**	**0.000**
6×25	8.692	2.379	4.525	2.193	4.740	0.807	*1.200*	2.000	4.738	1.350	1.703	*0.707*	1.790	1.156	2.552	**0.668**	**0.781**	**0.354**	**0.164**
6×30	1.773	0.539	*2.083*	0.535	2.485	0.926	**1.387**	0.317	2.979	**0.103**	2.671	0.189	2.116	*0.104*	2.548	0.174	3.790	**0.099**	**1.550**

Average	3.960	1.248	2.342	1.225	2.257	1.000	**0.845**	0.851	2.618	0.610	*1.371*	*0.547*	1.663	0.597	1.570	**0.486**	1.545	**0.291**	**0.475**

Table XII

COMPARISON RESULTS OF HIERC_Q WITH THE STATE-OF-THE-ART METHODS UNDER λ=400.

$S\times Z$	IDH	ISA		PIDE		HGA		HFOA		IPSO		EIGA		IABC		CCABC		HierC_Q	
	ARPD	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*	*ARPD*	*SD*
3×10	2.319	1.509	0.667	1.536	**0.153**	2.285	0.315	0.950	*0.269*	**0.468**	0.305	0.791	0.279	0.623	**0.237**	*0.526*	0.351	**0.454**	0.296
3×15	1.372	0.400	0.285	0.421	*0.200*	1.498	**0.000**	0.304	*0.128*	**0.244**	0.679	*0.249*	0.380	0.251	0.722	0.349	1.292	**0.240**	**0.085**
3×20	3.427	1.771	5.711	1.771	5.657	0.708	**0.065**	1.008	5.389	**0.288**	0.722	0.315	0.406	*0.289*	*0.397*	0.488	1.209	**0.284**	**0.145**
3×25	9.849	2.216	6.236	2.047	7.320	*0.804*	2.214	1.577	5.828	1.825	**1.368**	1.295	4.798	1.631	*1.698*	**0.612**	1.752	**0.249**	**0.000**
3×30	3.352	1.567	1.332	1.584	**0.735**	**0.449**	2.331	0.941	5.340	*0.637*	**0.409**	0.880	4.706	**0.512**	*0.863*	0.878	3.947	**0.413**	2.494
4×10	7.032	1.307	2.350	1.283	1.822	0.998	*0.555*	0.709	2.046	0.107	**0.000**	**0.400**	0.910	*0.402*	1.191	0.721	1.742	**0.393**	**0.440**
4×15	0.874	0.643	*1.439*	0.653	**1.035**	0.643	**0.196**	0.431	1.556	**0.334**	1.814	*0.339*	1.995	0.365	2.088	0.415	1.829	**0.327**	1.862
4×20	2.513	0.496	0.882	0.531	1.535	0.795	**0.000**	0.390	**0.000**	**0.144**	**0.735**	**0.070**	**0.000**	0.193	1.471	*0.166*	*0.744*	**0.070**	**0.000**
4×25	10.508	2.102	3.883	2.028	3.122	*0.893*	0.940	1.702	2.570	1.840	**0.602**	1.234	2.326	1.753	1.809	**0.621**	*0.690*	**0.294**	**0.000**
4×30	2.218	1.190	1.325	1.419	**0.429**	**0.526**	*0.443*	0.982	3.623	0.899	**0.000**	0.782	3.414	0.768	0.960	*0.711*	1.803	**0.344**	1.262
5×10	4.327	0.714	2.244	1.245	1.998	1.170	0.899	0.590	2.540	**0.287**	0.925	*0.300*	0.880	0.301	*0.719*	0.347	**0.351**	**0.286**	**0.010**
5×15	3.017	0.836	1.788	0.800	1.128	1.055	*0.581*	0.506	1.721	**0.155**	**0.346**	*0.158*	0.583	0.227	1.058	0.359	1.809	**0.154**	**0.000**
5×20	7.058	3.454	4.455	3.154	6.390	1.529	1.986	2.680	3.880	1.832	1.757	**0.622**	*1.361*	1.840	2.974	*0.820*	**0.972**	**0.283**	**0.000**
5×25	2.365	0.463	1.622	0.497	1.053	0.814	**0.480**	0.365	1.592	**0.124**	1.692	0.156	*0.634*	*0.154*	0.648	0.218	1.626	**0.117**	**0.453**
5×30	0.729	0.286	2.085	0.290	2.619	0.643	*0.900*	0.265	2.663	**0.263**	2.592	0.057	**0.756**	*0.267*	1.924	0.334	1.721	**0.259**	**0.537**
6×10	3.279	0.872	2.057	0.754	2.287	0.852	0.823	0.325	2.326	0.141	1.123	**0.138**	**0.223**	*0.139*	*0.431*	0.355	1.531	**0.136**	**0.000**
6×15	3.011	0.804	1.261	0.769	1.842	1.009	*0.660*	0.528	1.337	*0.254*	0.880	**0.141**	**0.463**	0.302	1.163	0.356	1.784	**0.152**	**0.000**
6×20	1.477	0.906	1.973	0.939	1.290	1.195	*0.866*	0.618	1.261	*0.301*	1.208	**0.268**	1.066	0.363	1.498	0.330	**0.782**	**0.150**	**0.040**
6×25	8.692	2.333	6.612	2.303	4.839	0.829	**1.184**	1.885	4.567	1.384	2.405	**0.566**	*1.259*	1.416	2.314	*0.640*	1.315	**0.239**	**0.017**
6×30	1.773	0.538	2.595	0.523	2.554	0.924	**1.349**	0.295	2.633	**0.090**	2.806	*0.170*	*2.495*	**0.089**	2.922	0.172	3.400	**0.089**	**1.469**
Average	3.960	1.220	2.540	1.227	2.400	0.981	**0.839**	0.853	2.563	0.581	*1.118*	**0.447**	1.447	0.594	1.354	*0.471*	1.533	**0.247**	**0.456**

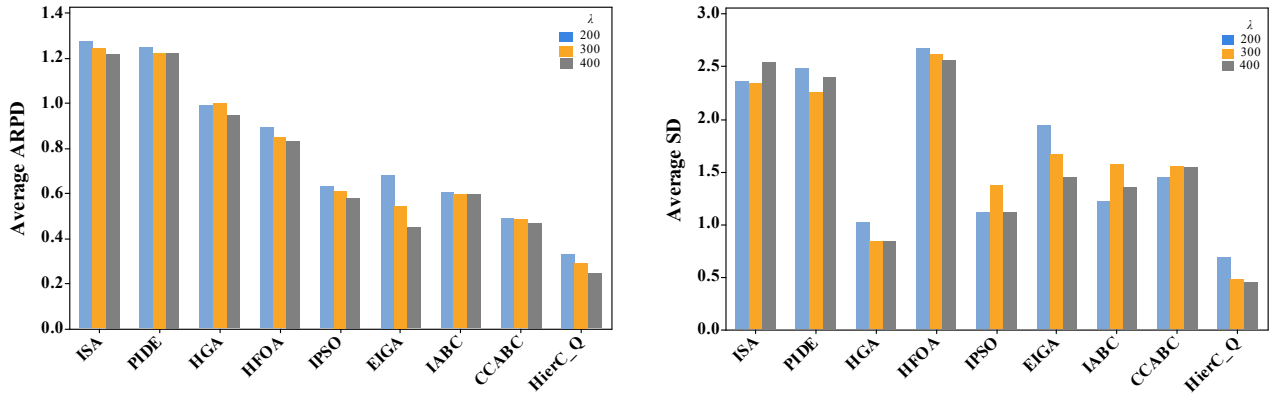
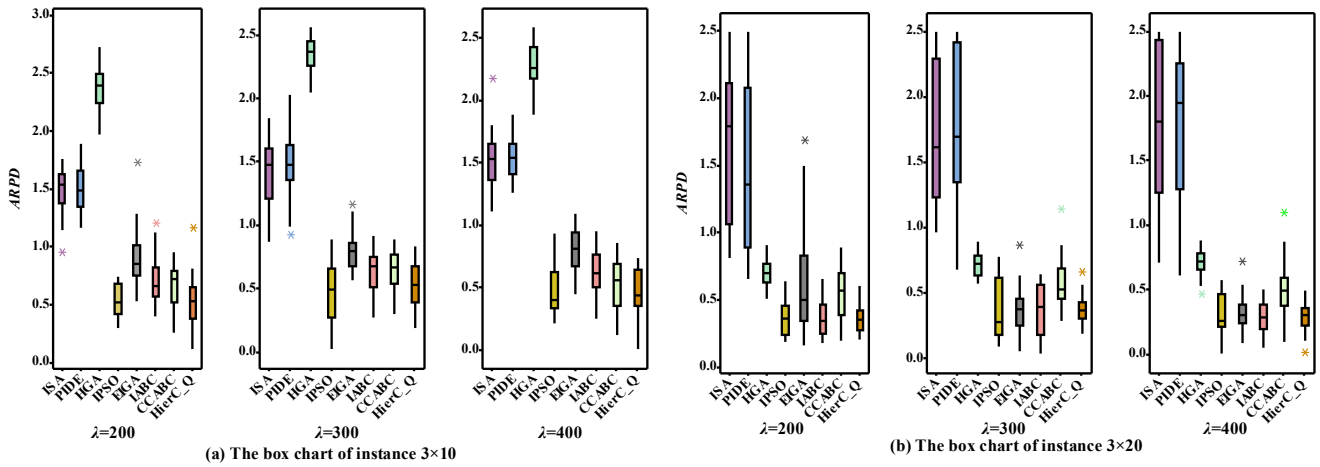


Fig. 7. Comparisons of HierC_Q's performance with the state-of-the-art algorithms.



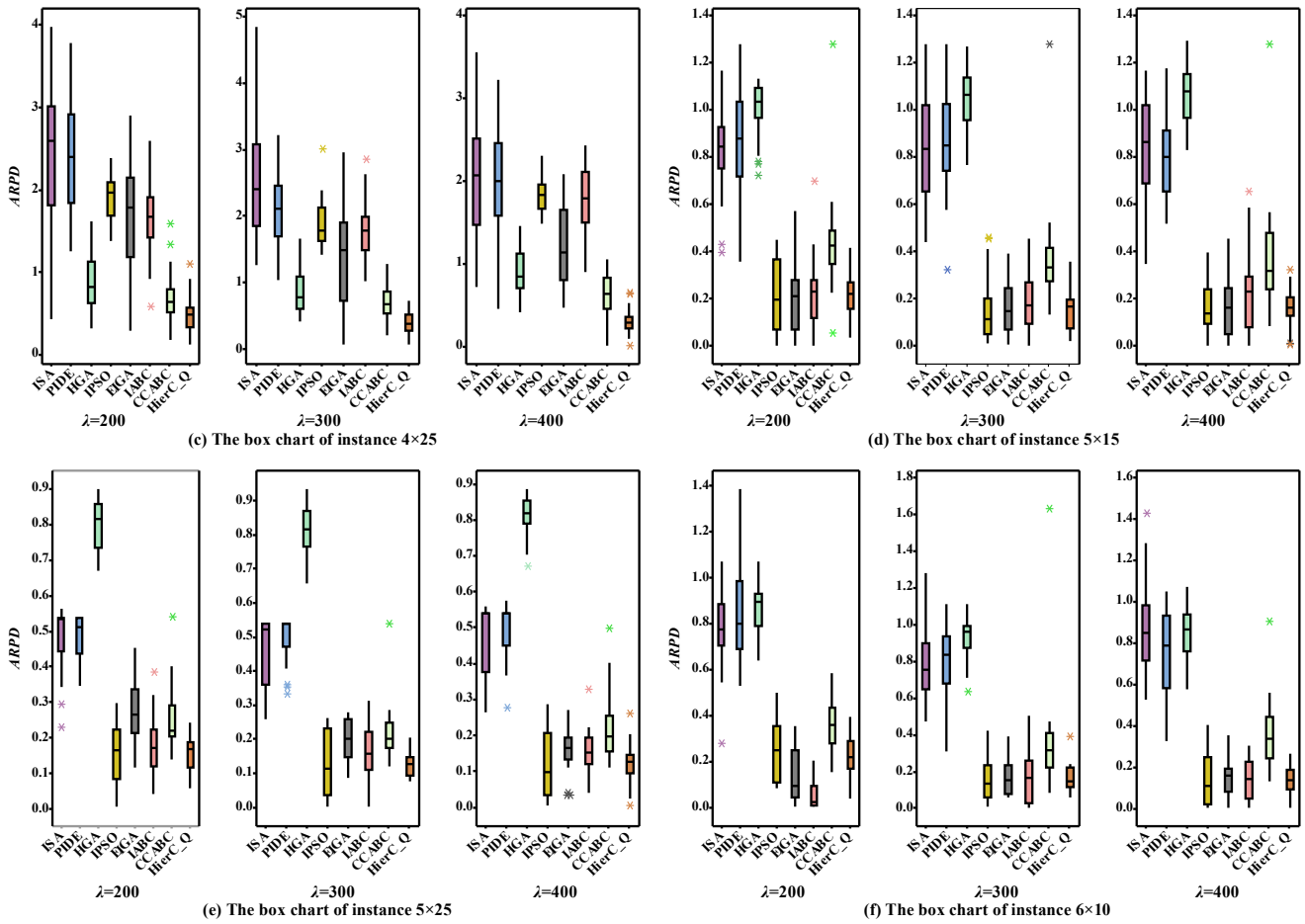


Fig. 1. The box plots for HierC_Q compared with the state-of-the-art algorithms.

Part 6. The pseudo codes of HF_NSs and their variants for solving SCCSP

1. The procedure of VNS.

The procedure of VNS

Input: an initial solution $\pi=(u,v)$;

The perturbation operators $\mathcal{F}_d(d=1, 2)$, d_{max}=2: *Insert_f*(v) and *Inter_r*(v);

The improvement operators $\mathcal{F}_i(i=1, 2, \ldots, 11)$, i_{max}=11:

For charge sequence: *SNS*, *MNS*, *BNS*, *SNI*, *MNI*, *BNI*, *NE*(1), *NE*(3);
For cast sequence: *NSS*, *NSE*, *NSI*;

Output: Best found solution π;

1. **Set** d=1, i=1;
2. **While** (Termination condition is not met) **do**
3. **Repeat**
4. $v'=\mathcal{F}_d(v)$;// *Perturbation procedure based on* $\mathcal{F}_d$
5. $u'\leftarrow u\_$ *constructing procedure*(v')
6. $\pi'=(u', v')$
7. **While** ($i< i_{max}$) **do**//
8. $\pi''=\mathcal{F}_i(\pi')$; //*Improvement procedure based on* $\mathcal{F}_i$
9. **If** $f(\pi'')< f(\pi)$ **then** $\pi=\pi''$, i=1;
10. **Else** $i=i$+1;
11. **End While**
12. $d=d$+1;
13. **Until** $d> d_{max}$
14. **End While**
15. **Return** π

2. The procedure of VNS_{var}.

The procedure of VNS_{v1}

Input: an initial solution $\pi=(u,v)$;

The perturbation operators $\mathcal{F}_d(d=1, 2)$, i_{max}=2: *Insert_f*(v) and *Inter_r*(v);

The improvement operators $\mathcal{F}_i(i=1, 2, \ldots, 6)$, d_{max}=6:

For charge sequence: *Swap*, *Insert, Exchange*;
For cast sequence: *Swap*, *Insert, Exchange*;

Output: Best found solution π;

1. **Set** d=1, i=1;
2. **While** (Termination condition is not met) **do**
3. **Repeat**
4. $v'=\mathcal{F}_d(v)$;// *Perturbation procedure based on* $\mathcal{F}_d$
5. $u'\leftarrow u\_$ *constructing procedure*(v')
6. $\pi'=(u', v')$
7. **While** ($i< i_{max}$) **do**//
8. $\pi''=\mathcal{F}_i(\pi')$; //*Improvement procedure based on* $\mathcal{F}_i$
9. **If** $f(\pi'')< f(\pi)$ **then** $\pi=\pi''$, i=1;
10. **Else** $i=i$+1;
11. **End While**
12. $d=d$+1;
13. **Until** $d> d_{max}$
14. **End While**
15. **Return** π

3. The procedure of ILS

The procedure of ILS

Input: an initial solution $\pi=(u,v)$;

The perturbation operators $\mathcal{F}_d(d=1, 2)$: *Insert_f*(v) and *Inter_r*(v);

The improvement operators $\mathcal{F}_i(i=1, 2, \ldots, 11)$, d_{max}=11:

For charge sequence: *SNS*, *MNS*, *BNS*, *SNI*, *MNI*, *BNI*, *NE*(1), *NE*(3);
For cast sequence: *NSS*, *NSE*, *NSI*;

Output: Best found solution π;

1. Set *count*=0; *max_count*=10;
2. **For** i=1 to i_{max} **do**
3. **Repeat**
4. $\pi'=\mathcal{F}_i(\pi)$;
5. **If** $f(\pi')< f(\pi)$ **then** $\pi=\pi'$, count=0
6. **Else** *count*=*count*+1;
7. **Until** *count*> *max_count*
8. **End For**
9. **While** (Termination condition is not met) **do**
10. Random select a perturbation operators $\mathcal{F}_d$
11. $v''=\mathcal{F}_d(v')$;// *Perturbation procedure based on* $\mathcal{F}_d$
12. $u''\leftarrow u\_$ *constructing procedure*(v')
13. $\pi''=(u'', v'')$, **Set** *count*=0;
14. **For** i=1 to i_{max} **do**
15. **Repeat**
16. $\pi'''=\mathcal{F}_i(\pi'')$;
17. **If** $f(\pi''')< f(\pi)$ **then** $\pi=\pi'''$, count=0
18. **Else** *count*=*count*+1;
19. **Until** *count*> *max_count*
20. **End For**
21. **End While**
22. **Return** π

4. The procedure of ILS_{var}

The procedure of ILS_{v1}

Input: an initial solution $\pi=(u,v)$;

The perturbation operators $\mathcal{F}_d(d=1, 2)$: *Insert_f*(v) and *Inter_r*(v);

The improvement operators $\mathcal{F}_i(i=1, 2, \ldots, 11)$, i_{max}=11:

For charge sequence: *Swap*, *Insert, Exchange*;
For cast sequence: *Swap*, *Insert, Exchange*;

Output: Best found solution π_{best};

1. Set *count*=0; *max_count*=10;
2. **For** i=1 to i_{max} **do**
3. **Repeat**
4. $\pi'=\mathcal{F}_i(\pi)$;
5. **If** $f(\pi')< f(\pi)$ **then** $\pi=\pi'$, count=0
6. **Else** *count*=*count*+1;
7. **Until** *count*> *max_count*
8. **End For**
9. **While** (Termination condition is not met) **do**
10. Random select a perturbation operators $\mathcal{F}_d$
11. $v''=\mathcal{F}_d(v')$;// *Perturbation procedure based on* $\mathcal{F}_d$
12. $u''\leftarrow u\_$ *constructing procedure*(v')
13. $\pi''=(u'', v'')$, **Set** *count*=0;
14. **For** i=1 to i_{max} **do**
15. **Repeat**
16. $\pi'''=\mathcal{F}_i(\pi'')$;
17. **If** $f(\pi''')< f(\pi)$ **then** $\pi=\pi'''$, count=0
18. **Else** *count*=*count*+1;
19. **Until** *count*> *max_count*
20. **End For**
21. **End While**
22. **Return** π

5. The procedure of GD

The procedure of GD

Input: an initial solution $\pi=(u,v)$;
The improvement operators $\mathcal{F}_i$ (i=1, 2, …, 11), i_{max}=11:
For charge sequence: *SNS*, *MNS*, *BNS*, *SNI*, *MNI*, *BNI*, *NE*(1), *NE*(3);
For cast sequence: *NSS*, *NSE*, *NSI*;
Output: Best found solution π;
1. **Set** *count*=0; *max_count*=10, *lever*=$f(\pi)$, *gen*=1, $\pi_{best}=\pi$;
2. **While** (Termination condition is not met) **do**
3. **For** i=1 to i_{max} **do**
4. **Repeat**
5. $\pi'=\mathcal{F}_i(\pi)$;
6. **If** $f(\pi')<f(\pi)$ or $f(\pi')<lever$ **then** $\pi=\pi'$, count=0;
7. **Else** *count*=*count*+1;
8. **If** $f(\pi')<f(\pi)$ **then** $\pi_{best}=\pi'$
9. **Until** *count*> *max_count*
10. **End For**
11. *gen*=*gen*+1; *lever*=*lever*-($f(\pi')$- $f(\pi_{best})$)/*gen*
12. **End While**
13. **Return** π_{best}

6. The procedure of GD$_{var}$

The procedure of GD

Input: an initial solution $\pi=(u,v)$;
The improvement operators $\mathcal{F}_i$ (i=1, 2, …, 6), i_{max}=6:
For charge sequence: *Swap*, *Insert, Exchange*;
For cast sequence: *Swap*, *Insert, Exchange*;
Output: Best found solution π;
1. Set *count*=0; *max_count*=10, *lever*=$f(\pi)$, *gen*=1, $\pi_{best}=\pi$;
2. **While** (Termination condition is not met) **do**
3. **For** i=1 to i_{max} **do**
4. **Repeat**
5. $\pi'=\mathcal{F}_i(\pi)$;
6. **If** $f(\pi')<f(\pi)$ or $f(\pi')<lever$ **then** $\pi=\pi'$, count=0;
7. **Else** *count*=*count*+1;
8. **If** $f(\pi')<f(\pi)$ **then** $\pi_{best}=\pi'$
9. **Until** *count*> *max_count*
10. **End For**
11. *gen*=*gen*+1; *lever*=*lever*− ($f(\pi')-f(\pi)$)/*gen*
12. **End While**
13. **Return** π_{best}

7. The procedure of ALNS

The procedure of ALNS

Input: an initial solution $\pi=(u,v)$;
The destroy operators $\mathcal{F}_d$ (d=1, 2, 3), d_{max}=3:
For cast sequence: *NSS*, *NSE*, *NSI*;
The repair operators $\mathcal{F}_r$ (r=1, 2, …, 8), r_{max}=8:
For charge sequence: *SNS*, *MNS*, *BNS*, *SNI*, *MNI*, *BNI*, *NE*(1), *NE*(3);
Output: Best found solution π_{best};
1. Initial the element of the weight set Φ_d and Φ_r as 1;
2. **While** (Termination condition is not met) **do**
3. Select a destroy operator $\mathcal{F}_i$ and a repair operator $\mathcal{F}_j$ according to Φ_d and Φ_r by using a tournament selection method
4. $v'\leftarrow\mathcal{F}_i(v)$//Apply destroy operator to destroy the cast sequence
5. $u'\leftarrow\mathcal{F}_j(v)$//Apply repair operator to repair the charge sequence
6. $\pi'=(u',v')$
7. **If** $f(\pi')<f(\pi_{best})$ **then**
8. $\pi_{best}=\pi'$, $\Phi_{d(i)}=0.7\times\Phi_{d(i)}+(1-0.7)\times 10$, $\Phi_{r(j)}=0.7\times\Phi_{r(j)}+(1-0.7)\times 10$
9. **If** $f(\pi')<f(\pi)$ **then**
10. $\pi=\pi'$, $\Phi_{d(i)}=0.7\times\Phi_{d(i)}+(1-0.7)\times 5$, $\Phi_{r(j)}=0.7\times\Phi_{r(j)}+(1-0.7)\times 5$
11. **Else**
12. $\Phi_{d(i)}=0.7\times\Phi_{d(i)}+(1-0.7)\times 2$, $\Phi_{r(j)}=0.7\times\Phi_{r(j)}+(1-0.7)\times 2$
13. **End While**
14. **Return** π_{best}

8. The procedure of ALNS$_{var}$

The procedure of ALNS

Input: an initial solution $\pi=(u,v)$;
The destroy operators $\mathcal{F}_d$ (d=1, 2, 3), d_{max}=3:
For cast sequence: *Swap*, *Insert, Exchange*;
The repair operators $\mathcal{F}_r$ (r=1, 2, 3), r_{max}=3:
For charge sequence: *Swap*, *Insert, Exchange*;
Output: Best found solution π_{best};
1. Initial the element of the weight set Φ_d and Φ_r as 1;
2. **While** (Termination condition is not met) **do**
3. Select a destroy operator $\mathcal{F}_i$ and a repair operator $\mathcal{F}_j$ according to Φ_d and Φ_r by using a tournament selection method
4. $v'\leftarrow\mathcal{F}_i(v)$//Apply destroy operator to destroy the cast sequence
5. $u'\leftarrow\mathcal{F}_j(v)$//Apply repair operator to repair the charge sequence
6. $\pi'=(u',v')$
7. **If** $f(\pi')<f(\pi_{best})$ **then**
8. $\pi_{best}=\pi'$, $\Phi_{d(i)}=0.7\times\Phi_{d(i)}+(1-0.7)\times 10$, $\Phi_{r(j)}=0.7\times\Phi_{r(j)}+(1-0.7)\times 10$
9. **If** $f(\pi')<f(\pi)$ **then**
10. $\pi=\pi'$, $\Phi_{d(i)}=0.7\times\Phi_{d(i)}+(1-0.7)\times 5$, $\Phi_{r(j)}=0.7\times\Phi_{r(j)}+(1-0.7)\times 5$
11. **Else**
12. $\Phi_{d(i)}=0.7\times\Phi_{d(i)}+(1-0.7)\times 2$, $\Phi_{r(j)}=0.7\times\Phi_{r(j)}+(1-0.7)\times 2$
13. **End While**
14. **Return** π

9. The procedure of HHGA

The procedure of HHGA

Input: the improvement operators $\mathcal{F}_i$ (i=1, 2, …, 11), i_{max}=11:
For charge sequence: 1: *SNS*; 2: *MNS*; 3: *BNS*; 4: *SNI*; 5: *MNI*; 6: *BNI*; 7: *NE*(1); 8: *NE*(3);
For cast sequence: 9: *NSS*; 10: *NSE*; 11: *NSI*;
Parameter setting: *popsize*=20, *pm*=0.05, *pc*=0.85
Population Initialize:
A low-level individual domain population $\Omega_{low}=\{\pi_1, \pi_2,\ldots, \pi_{popsize}\}$// where $\pi_i=(u_i,v_i)$
A high-level strategy domain population $\Omega_{high}=\{y_1, y_2,\ldots, y_{popsize}\}$ //where $y_i=(a_1, a_2,\ldots a_{11})$ stands for a full arrangement of the above improvement operators
Output: Best found solution π;

1. Combine Ω_{low} and Ω_{high} into an population $\Omega=\{\mu_1, \mu_2,\ldots, \mu_{popsize}\}$,$\mu_i=\{\pi_i, y_i\}$
2. **While** (Termination condition is not met) **do**
3. **For** i=1 to *popsize* **do**
4. **For** j=1 to i_{max} **do**
5. $\pi'=\mathcal{F}_j(\pi_i)$;
6. **If** $f(\pi')<f(\pi)$ **then** $\pi=\pi'$
7. **End For**
8. **End For**
10. $Parent_1 \leftarrow TournamentSlection\{y_1, y_2,\ldots, y_{popsize}\}$
11. $Parent_2 \leftarrow TournamentSlection\{y_1, y_2,\ldots, y_{popsize}\}$
12. $(Child_1, Child_2) \leftarrow Crossover(Parent_1, Parent_2, pc)$
13. $Child_1 \leftarrow Mutation(Child_1, pm)$
14. $Child_2 \leftarrow Mutation(Child_2, pm)$
15. $\mu_{worst} \leftarrow \max(f(\pi_i))$, ($i$=1, 2,…,*popsize*)
16. **If** $f(Child_1)<f(\mu_{worst})$ **then**
17. $\{\mu_1, \mu_2,\ldots, \mu_{popsize}\} \leftarrow \{\{\mu_1, \mu_2,\ldots, \mu_{popsize}\}-\mu_{worst}\}+\{Child_1\}$
17. **If** $f(Child_2)<f(\mu_{worst})$ **then**
18. $\{\mu_1, \mu_2,\ldots, \mu_{popsize}\} \leftarrow \{\{\mu_1, \mu_2,\ldots, \mu_{popsize}\}-\mu_{worst}\}+\{Child_2\}$
19. **End While**
20. **Return** π

10. The procedure of HHGA$_{var}$

The procedure of HHGA $_{v1}$

Input: the improvement operators $\mathcal{F}_i$ (i=1, 2, …, 6), i_{max}=6:
For charge sequence: 1: *Swap*; 2: *Insert*; 3: *Exchange*;
For cast sequence: 4: *Swap*; 5: *Insert*; 6: *Exchange*;
Parameter setting: *popsize*=20, *pm*=0.05, *pc*=0.85
Population Initialize:
A low-level individual domain population $\Omega_{low}=\{\pi_1, \pi_2,\ldots, \pi_{popsize}\}$// where $\pi_i=(u_i,v_i)$
A high-level strategy domain population $\Omega_{high}=\{y_1, y_2,\ldots, y_{popsize}\}$ //where $y_i=(a_1, a_2,\ldots a_{11})$ stands for a full arrangement of the above improvement operators
Output: Best found solution π;

1. Combine Ω_{low} and Ω_{high} into an population $\Omega=\{\mu_1, \mu_2,\ldots, \mu_{popsize}\}$,$\mu_i=\{\pi_i, y_i\}$
2. **While** (Termination condition is not met) **do**
3. **For** i=1 to *popsize* **do**
4. **For** j=1 to i_{max} **do**
5. $\pi'=\mathcal{F}_j(\pi_i)$;
6. **If** $f(\pi')<f(\pi)$ **then** $\pi=\pi'$
7. **End For**
8. **End For**
10. $Parent_1 \leftarrow TournamentSlection\{y_1, y_2,\ldots, y_{popsize}\}$
11. $Parent_2 \leftarrow TournamentSlection\{y_1, y_2,\ldots, y_{popsize}\}$
12. $(Child_1, Child_2) \leftarrow Crossover(Parent_1, Parent_2, pc)$
13. $Child_1 \leftarrow Mutation(Child_1, pm)$
14. $Child_2 \leftarrow Mutation(Child_2, pm)$
15. $\mu_{worst} \leftarrow \max(f(\pi_i))$, ($i$=1, 2,…,*popsize*)
16. **If** $f(Child_1)<f(\mu_{worst})$ **then**
17. $\{\mu_1, \mu_2,\ldots, \mu_{popsize}\} \leftarrow \{\{\mu_1, \mu_2,\ldots, \mu_{popsize}\}-\mu_{worst}\}+\{Child_1\}$
18. **If** $f(Child_2)<f(\mu_{worst})$ **then**
19. $\{\mu_1, \mu_2,\ldots, \mu_{popsize}\} \leftarrow \{\{\mu_1, \mu_2,\ldots, \mu_{popsize}\}-\mu_{worst}\}+\{Child_2\}$
20. **End While**
21. **Return** π

11. The procedure of HierC_NS$_{var}$

HierC_Q$_{var}$ is a variant of HierC_Q, the difference between them is the neighborhood operation. We only need to replace the neighborhood operations of algorithm 3 and algorithm 4 with the following ones: For charge sequence: *Swap*; *Insert*; *Exchange*; For cast sequence: *Swap*; *Insert*; *Exchange*.

Part 7. Pseudo codes of the state-of-the-art methods

1. The procedure of IDH

IDH handles the cast subproblem and the charge subproblem separately, with the former being solved first. For the cast subproblem, each cast is first arranged in decreasing order of processing time to produce a cast subsequence. Each cast in the cast subsequence must be picked in order from the back to the front and then assigned to the first casting machine available as quickly as practicable. In this manner, the partial scheduling scheme (i.e., the starting processing time and the processing machine of each cast) in the continuous casting stage is determined. Meanwhile, the start processing time of each charge in this stage is collected. For the charge subproblem, all charges are arranged in ascending order of their obtained start times in the continuous casting stage to produce a charge subsequence. Each charge is extracted from the charge subsequence from the end to the beginning and then assigned to an available refining machine or converter from the last refining stage to the steelmaking stage. In each of these stages, each charge is allocated to the first available machine in reverse order and launched as late as possible. So far, the full scheduling scheme has been determined.

The procedure of IDH

Input: The processing time of each charge in each stage;
Output: A flexible solution π;
1. Calculate the processing time of each cast according the charges in the continuous casting stage
2. Sort all of the casts in decreasing order of their processing time and generate the cast sequence v
3. Schedule the casts to the first available machine according to v in the continuous casting stage
4. Compute the starting processing time for each charge of each cast
5. Sort all of the charges in an increasing order of the start processing time and obtain a charge sequence u
6. Compose v and u into a feasible solution π and calculate its fitness $f(\pi)$
7. **Return** π

2. The procedure of ISA

The improved simulated annealing algorithm is proposed to minimize both total completion time and total tardiness of the hybrid flowshops with sequence-dependent setup and transportation times. All steps of this algorithm are strictly implemented according to the literature, including the neighborhood search structure, a diversification mechanism, and local search method. To make ISA applicable to the issues discussed in this paper, we have replaced its evaluation functions with the weighted sum of the maximum completion time and the average waiting time in Section II. We also incorporate the proposed encoding, decoding, and initialization method into ISA. It is worth noting that SCCSP is a dual vector encoding, so the neighborhood search needs to operate on two subsequences. Besides, since the validity evaluation method and the speed-up evaluation method presented in Subsection IV-C are a general, ISA utilizes them to improve search efficiency. The parameters of these compared algorithms are directly taken from the original literature and properly calibrated. Besides, we employ the Taguchi method to select the optimum parameters. The initial temperature T_0 is set as 20, the final temperature T_f is fixed at 170, the number of neighborhood searches n_T at each temperature is 100, and the cooling schedule type is hyperbolic. The procedure of ISA for the SCCSP is as follows.

The procedure of ISA

Output: Best found solution π_{best};
1. **Parameter setting**: T_0=20, T_f=170, n_T =100;
2. Initialize π
3. **While** (Termination condition is not met) **do**
4. $T \leftarrow CoolingSchedule(T_0, T_f)$
5. $Count \leftarrow 0$
6. **Repeat**
7. $\pi' \leftarrow Shift(\pi)$
8. **If** $f(\pi') < f(\pi)$ **then** $\pi \leftarrow \pi'$
9. **Else If** rand(0,1) $< \exp\{(f(\pi)-f(\pi'))/T\}$ then $\pi \leftarrow \pi'$
10. $Count \leftarrow Count+1$
11. **Until** $Count = n_T$
12. $\pi' \leftarrow DM2(\pi)$
13. $\pi'' \leftarrow LocalSearch\ (\pi')$
14. **If** $f(\pi'') < f(\pi)$ **then** $\pi \leftarrow \pi'$
15. **End While**
16. **Return** π_{best}

3. The procedure of PIDE

The PIDE is proposed for the dynamic steelmaking-continuous casting scheduling problem (DSCCSP). Different from the problem studied in our paper, this problem considers the unforeseen changes that occur in the production system. The main idea of PIDE is to improve the generated population via three operations: mutation, crossover, and selection, and generate to a new schedule when an unexpected event happens via a novel incremental mechanism. Consequently, we must tailor PIDE to the problem under consideration. Specifically, the population updating method under the incremental mechanism is removed. The remaining steps of PIDE are strictly implemented according to the literature, including the real-coded matrix-based encoding and decoding method, the hybrid heuristic population initialization method, and the novel evolutionary operations of the proposed PIDE. It is important to note that the validity evaluation method and the speed-up evaluation methods are not applicable to PIDE. The parameters of these compared algorithms are directly taken from the original literature and properly calibrated. The population size NP is set as 50, the crossover probability CR is fixed at 0.5, and the mutation factor F is set as 0.95. The procedure of PIDE for the SCCSP is as follows.

The procedure of PIDE

Output: Best found solution π_{best};
1. **Parameter setting**: NP =50, CR=0.5, F=0.95
2. Initialize population $\mathbf{P_0}\{\pi_1, \pi_2,..., \pi_{NP}\}$, and evaluate each solution in the population.
3. Set the set of archived superior solutions **B** is empty, and initialize the memory population $\mathbf{P^M}$ as $\mathbf{P_0}$.
3. **While** (Termination condition is not met) **do**
4. Save the current population $\mathbf{P_g}$ to $\mathbf{P^C}$
5. **For** i=1 to NP
6. Randomly choose one from the p best vectors in $\mathbf{P^M}$ as $\pi^{\mathbf{M}}_{\boldsymbol{best,g}}$;
7. Randomly choose $\pi_{r1,g}$, $\pi_{r2,g}$, $\pi_{r3,g}$ from $\mathbf{P_g}$;
8. Randomly choose $\pi_{r4,g}$ from $\mathbf{P_g} \cup \mathbf{B}$;
9. Randomly choose $\pi_{r5,g}$, $\pi_{r6,g}$ from $\mathbf{P^M}\backslash\{ p \}$.
10. $v_{i,g}= \pi_{i,g}+F(\pi^{\mathbf{M}}_{\boldsymbol{best,g}} - \pi_{i,g})+F(\pi_{r1,g} - \pi_{r2,g})+ F(\pi_{r3,g} - \pi_{r4,g})+ F(\pi_{r5,g} - \pi_{r6,g})$
11. Generate j_{rand} = a uniformly distributed random integer $\in \{1, 2, \ldots, D\}$
12. **For** j =1 to D
13. **If** $rand_j(0,1)=<CR$ or $j=j_{rand}$ then
14. $u_{j,i,g} =v_{j,i,g}$
15. **Else** $u_{j,i,g} = \pi_{j,i,g}$
16. **End For**
10. **If** $f(u_{i,g})<f(\pi_{i,g})$ **then**
11. $\pi_{i,g}= u_{i,g}$
12. **Else** $\pi_{i+1,g}=\pi_{i,g}$, $\pi_{i,g}\rightarrow\mathbf{B}$
13. **End For**
14. Randomly remove solutions from B so that 15. $|\mathbf{B}|=<NP$. Set $\mathbf{P^M}$ to be $\mathbf{P^C}$
16. Set the generation counter $g = g + 1$.
17. **End While**
18. **Return** π_{best}

4. The procedure of HGA

The hybrid genetic algorithm (HGA) is proposed for the hybrid flowshops with sequence-dependent setup times and machine eligibility. The HGA incorporates four new crossover operators (i.e., SJOX, SBOX, SJ2OX, and SB2OX) and restart schemes to improve search efficiency. We adopt the same objective evaluation, encoding, decoding, and initialization method (one individual is generated by using the method in our paper, and the other is randomly generated). For the selection, we simply use one of the most well-known selection schemes, which is the tournament selection method. The four crossover operators are employed for the charge subsequence, the cast subsequence, or both of the selected two subsequences. To further avoid premature convergence in the population HGA has implemented two methods, named restart and generational schemes. All steps of HGA are strictly implemented according to the literature. Besides, it is necessary to note that the validity evaluation method and the speed-up evaluation methods are also not applicable to HGA. The parameters of these compared algorithms are directly taken from the original literature and properly calibrated. More precisely, the population size, crossover operators, crossover probability, and mutation probability all take the same value: Ps=50, SB2OX, Pc =0.5, Pm =0.01 respectively.

The procedure of HGA

Output: Best found solution π_{best};
1. **Parameter setting**: Ps=50, Pc=0.5, Pm=0.1
2. Initialize population $\{\pi_1, \pi_2,..., \pi_{Ps}\}$
3. **While** (Termination condition is not met) **do**
4. $Parent_1 \leftarrow TournamentSlection\{\pi_1, \pi_2,..., \pi_{Ps}\}$
5. $Parent_2 \leftarrow TournamentSlection\{\pi_1, \pi_2,..., \pi_{Ps}\}$
6. $(Child_1, Child_2) \leftarrow Crossover(Parent_1, Parent_2, pc)$
7. $Child_1 \leftarrow Mutation(Child_1, pm)$
8. $Child_2 \leftarrow Mutation(Child_2, pm)$
9. $\mu_{worst} \leftarrow \max(f(\pi_i))$, $(i=1, 2,\ldots,popsize)$
10. **If** $f(Child_1)<f(\mu_{worst})$ **then**
11. $\{\mu_1, \mu_2,\ldots, \mu_{Ps}\}\leftarrow\{\{\mu_1, \mu_2,\ldots, \mu_{Ps}\}-\mu_{worst}\}+\{Child_1\}$
12. **If** $f(Child_2)<f(\mu_{worst})$ **then**
13 $\{\mu_1, \mu_2,\ldots, \mu_{Ps}\}\leftarrow\{\{\mu_1, \mu_2,\ldots, \mu_{Ps}\}-\mu_{worst}\}+\{Child_2\}$
14. Update π_{best}
15. **End While**
16. **Return** π_{best}

5. The procedure of HFOA

Similar to the PIDE, the hybrid fruit fly optimization algorithm (HFOA) is also proposed for solving the dynamic steelmaking continuous casting scheduling problems (DSCCSP) by considering machine breakdown and processing variation disruptions. Therefore, we must replace some components of HFOA to adapt it to our considered problem. First, we have substituted the coding and decoding method, the evaluation functions, as well as the population initialization strategy presented in our paper. Besides, considering the applicability of the neighborhood search operator, the problem-dependent neighborhood operators are employed to perform the local search, and the validity evaluation method and the speed-up evaluation methods are used to improve its performance. The remaining steps of HFOA are strictly implemented according to the literature. The parameters of HFOA are directly taken from the original literature and properly calibrated. The population size PS=20. The number of iterations L_n=0.6. The size of the neighborhood in the smell-based search process $SN=n/5$. The learning probability p_l=0.7. the maximum learning strength l_{max}=5. The destruction length LEN=4.

The procedure of is HFOA

Output: Best found solution π_{best};
1. **Parameter setting**: PS, L_n, SN, p_l, l_{max}, LEN;
2. Initialize population $\{\pi_1, \pi_2,..., \pi_{Ps}\}$ and evaluate each solution in the population. Set num=1.
3. **While** (Termination condition is not met) **do**
4. //Smell-based search
5. **Repeat**
6. **For** i=1 to NP
7. Generate a neighboring solution $\pi_{i,1}$ using the neighborhood structures
8. Record $\pi_{i,1}$ into the set T_i
9. **End**
9. **Until** $num>SN$
8. //Vision-based search
9. Evaluate each neighboring solution in T_i
10. Update each fruit fly and the best solution with the best neighboring solutions
11. Sort the entire population according to their fitness values
12. Generate a neighboring solution π_j around the best fruit fly by performing the perturbation-and-construction-based solution renewal strategy
13. Replace the worst solution with π_j
14. //Exploration procedure
15. Set the update iteration number (UIN) for each solution
15. Find the solution π_u with UIN bigger than L_n
16. Generate π_g by performing the IG-based local search for the best fruit fly
18. Replace π_u with π_g
17. **End While**
18. **Return** π_{best}

6. The procedure of IPSO

The improved particle swarm optimization algorithm (IPSO) is used for solving s the multi-stage hybrid flowshop scheduling problem with identical parallel machines. It develops a heuristic rules-based population initialization method and a variable neighborhood search method (VNS), to improve the solution on the foundation of the basic PSO algorithm. It should be pointed that the two heuristic rules (i.e., SPT dispatching rule and NEH heuristics) for population initialization in IPSO and the decoding method were not suitable for SCCSP, therefore, we employ the same encoding, decoding, and initialization method in our paper. Besides, we extend the VNS based on two simple neighborhood operations (i.e., adjacent pairwise interchange and swap) in IPSO to the problem depend neighborhood operators designed in this paper, and the validity evaluation method and the speed-up evaluation methods are used to improve its performance. The parameters of IPSO are directly taken from the original literature and properly calibrated. The procedure of IPSO for the SCCSP is as follow.

The procedure of is IPSO

Output: Best found solution π_{best};
1. **Parameter setting**: The population size: N; Number of iterations: I; Self-learning factor: C_1; Global learning factor: C_2;
2. Initialize population $\{\pi_1, \pi_2,..., \pi_{Ps}\}$, the velocity V_i^t, and position P_i^t of each particles
3. **While** (Termination condition is not met) **do**
4. Find out the global best particle π_{global}, i=1;
5. **Repeat**
6. *// Updating the velocity*
7. $V_{id}^{t+1} \leftarrow V_{id}^t + 0.5\times C_1[P_{id}^t - X_{id}^t] + 0.5\times C_2[G_{global}^t - X_{id}^t]$
8. *// Updating the position*
9. $X_{id}^{t+1} \leftarrow X_{id}^t + V_{id}^t$
10. Determine the new permutations by using the SPV rule
11. $i \leftarrow i+1$
12. **Until** $i > N$
13. Obtain the near-optimal solution x'
14. $\pi'' \leftarrow VNS(\pi')$
15. **If** $f(\pi'') < f(\pi')$ **then** $\pi'' \leftarrow \pi'$
16. **If** $f(\pi'') < f(\pi_{global})$ **then** $\pi_{global} \leftarrow \pi'$
17. **End While**
18. **Return** π_{best}

7. The procedure of EIGA

The enhanced iterated greedy algorithms (EIGA) is proposed for solving hybrid flexible flowshop scheduling problem with sequence-dependent setup times. All steps of EIGA are strictly implemented according to the literature, except that EIGA's specific destruction and construction mechanisms for the HFSP are replaced with the perturbation-and-construction-based solution renewal strategy presented in Section IV-D, and the search operators need to be executed separately for both subsequences. We also replace its evaluation functions with the weighted sum of the maximum completion time and the average waiting time criterion presented in Section II. Since the validity evaluation method and the speed-up evaluation methods presented in Subsection IV-C are general, EIGA utilizes them to improve search efficiency. Besides, all parameters were carefully implemented according to the original literature. The decrement coefficient θ in destruction is set as 0.05, the initial destruction rate per^1 is 0.2. The initial temperature for the acceptance criterion is T_1 is fixed at 1000. The size of local search *locSize* is 10. The maximum number of iterations for termination *maxIter* is set as 1000. The learning coefficient of HH Δ_{hh} is fixed to unity in all tests. The cooling parameter η for temperature is fixed as 0.95. The procedure of EIGA for the SCCSP is as follows.

The procedure of EIGA

Output: Best found solution π_{best};
1. **Parameter setting**: θ, per^1, T_1, *locSize*, *maxIter*, Δ_{hh}, η;
2. Initialize π_0
3. $\pi \leftarrow localSearch(\pi_0)$, $\pi_{best} \leftarrow \pi_0$
3. **While** (Termination condition is not met) **do**
4. $\pi' \leftarrow$Solution renewal strategy (π)
5. $\pi'' \leftarrow localSearch(\pi')$
6. **If** $acceptanceCriterion(\pi, \pi'', T)$ **then**
7. $\pi \leftarrow \pi''$
8. **If** $f(\pi) < f(\pi_{best})$ **then** $\pi_{best} \leftarrow \pi$
9. **End If**
10. **End If**
11. *update* α
12. **End While**
13. **Return** π_{best}

8. The procedure of IABC

The improved artificial bee colony algorithm (IABC) is proposed for the steelmaking continuous casting scheduling problem with fixed cast subsequence. The benefit of the IABC algorithm is that it can realize a more thorough search of the solution space by designing two different types of VNS (VNS_1, VNS_2) operations for the employed bee and the onlooker bee stages, respectively. However, IABC searches only for the charge subsequence, while keeping the cast subsequence unchanged. Therefore, we must tailor IABC to the considered problem. In particular, we have maintained the two types of VNS operations, and also performed them for the original fixed cast subsequence, as well as changed the original encoding method and objective function to make it applicable to the SCCSP. Remaining steps of IABC are strictly implemented according to the literature. The procedure of IABC for the SCCSP is as follow.

The procedure of IABC

Output: Best found solution π_{best};
1. **Parameter setting**: PS=30, L=100, MI=10, r=3
2. Initialize population $\{\pi_1, \pi_2, ..., \pi_{PS}\}$, and the context vector $\pi_c=(u_{cb}, v_{cb})$, $\pi_{best} \leftarrow \pi_c$, $count$=0.
3. **Repeat**
4. //Employed bee phase
5. **For** i = 1, 2,..., PS **do**
6. $u'_i \leftarrow VNS_1(u_i)$
7. **If** $f(u'_i, v_c) < 1.05 \times f(u_i, v_c)$ then $u_i \leftarrow u'_i$
8. **If** $f(u'_i, v_c) < 1.05 \times f(u_c, v_c)$ then $u_c \leftarrow u'_i$
9. **End**
10. **For** i = 1, 2,..., PS **do**
11. $v'_i \leftarrow VNS_1(v_i)$
12. **If** $f(u_c, v'_i) < 1.05 \times f(u_c, v_i)$ then $v_i \leftarrow v'_i$
13. **If** $f(u_c, v'_i) < 1.05 \times f(u_c, v_c)$ then $v_c \leftarrow v'_i$
14. **End**
15. //Onlooker bee phase
16. **Repeat**
17. Select MI solutions using tournament selection
18. **For** i = 1, 2,..., MI **do**
19. $u'_i \leftarrow VNS_2(u_i)$
20. **If** $f(u'_r, v_c) < 1.05 \times f(u_w, v_c)$ then $u_w \leftarrow u'_r$
21. **If** $f(u'_r, v_c) < 1.05 \times f(u_c, v_c)$ then $u_c \leftarrow u'_r$
22. **End**
23. **For** i = 1, 2,..., MI **do**
24. $u'_i \leftarrow VNS_2(u_i)$
25. **If** $f(u_c, v'_r) < 1.05 \times f(u_c, v_w)$ then $v_w \leftarrow v'_r$
26. **If** $f(u_c, v'_r) < 1.05 \times f(u_c, v_w)$ then $v_c \leftarrow v'_r$
27. **End**
28. $count=count+1$;
29. Until $count>r$
30. Perform the exploration scheme if π_c has not been improved in L iterations
31. $\pi_{best} \leftarrow \pi_c$ if $f(\pi_c) < f(\pi_{best})$
32. Until a termination criterion is met
33. **Return** π_{best}

9. The procedure of CCABC

The cooperative co-evolutionary artificial bee colony (CCABC) algorithm is currently recognized as the most efficient algorithm for solving the considered problem. All steps of this algorithm are strictly implemented according to the literature, and the problem dependent neighborhood operators, the validity evaluation method and the speed-up evaluation method are used to improve search efficiency. The parameters of CCABC, including the population size PS and iterative control parameter θ were carefully implemented according to the original literature and properly calibrated. The procedure of CCABC for the SCCSP is as follow.

The procedure of CCABC

Output: Best found solution π_{best};
1. **Parameter setting**: PS, θ
2. Initialize population $\{\pi_1, \pi_2, ..., \pi_{PS}\}$, and the context vector $\pi_c=(u_c,v_c)$.
3. **Repeat**
3. //Employed bee phase
4. **For** i = 1, 2,..., PS **do**
5. Generate u'_i for u_i using the self-adaptive neighbourhood operator
6. **If** $f(u'_i, v_c) < f(u_i, v_c)$ then $u_i \leftarrow u'_i$
7. **If** $f(u'_i, v_c) < f(u_c, v_c)$ then $u_c \leftarrow u'_i$
8. **End**
9. **For** i = 1, 2,..., PS **do**
10. Generate v'_i for v_i using the self-adaptive neighbourhood operator
11. **If** $f(u_c, v'_i) < f(u_c, v_i)$ then $v_i \leftarrow v'_i$
12. **If** $f(u_c, v'_i) < f(u_c, v_c)$ then $v_c \leftarrow v'_i$
13. **End**
14. //Onlooker bee phase
15. **For** i = 1, 2,..., PS **do**
16. Select a charge sequence u_r using tournament selection
17. Generate u'_r for u_r using the self-adaptive neighbourhood operator
18. **If** $f(u'_r, v_c) < f(u_w, v_c)$ then $u_w \leftarrow u'_r$
19. **If** $f(u'_r, v_c) < f(u_c, v_c)$ then $u_c \leftarrow u'_r$
20. **End**
21. **For** i = 1, 2,..., PS **do**
22. Select a cast sequence v_r using tournament selection
23. Generate v'_r for v_r using the self-adaptive neighbourhood operator
24. **If** $f(u_c, v'_r) < f(u_c, v_w)$ then $v_w \leftarrow v'_r$
25. **If** $f(u_c, v'_r) < f(u_c, v_w)$ then $v_c \leftarrow v'_r$
26. **End**
27. Perform the exploration scheme if π_c has not been improved in θ iterations
28. $\pi_{best} \leftarrow \pi_c$ if $f(\pi_c) < f(\pi_{best})$
29. Until a termination criterion is met
30. **Return** π_{best}

Part 8 Gantt chart of all scheduling schemes

The best results and Gantt chart found by HierC_Q for solving the SCCSP for 20 instances of various scales are reported as follows.

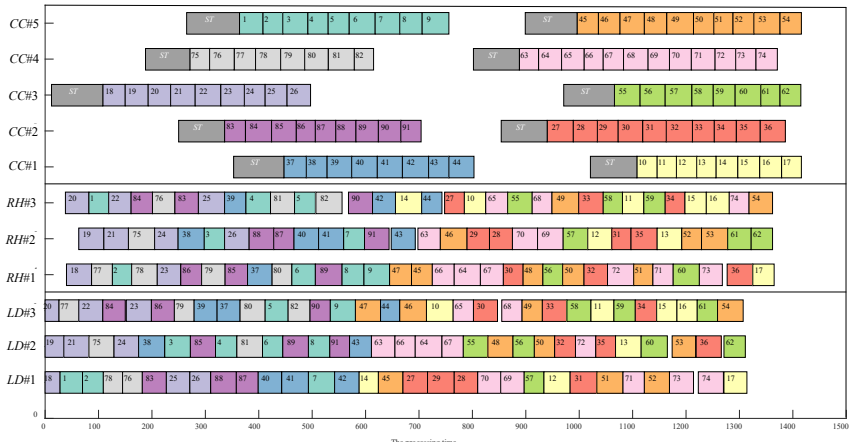


Fig. 9. Scheduling scheme of test date 3×10 (the completion time is 1416, the waiting time is 172)

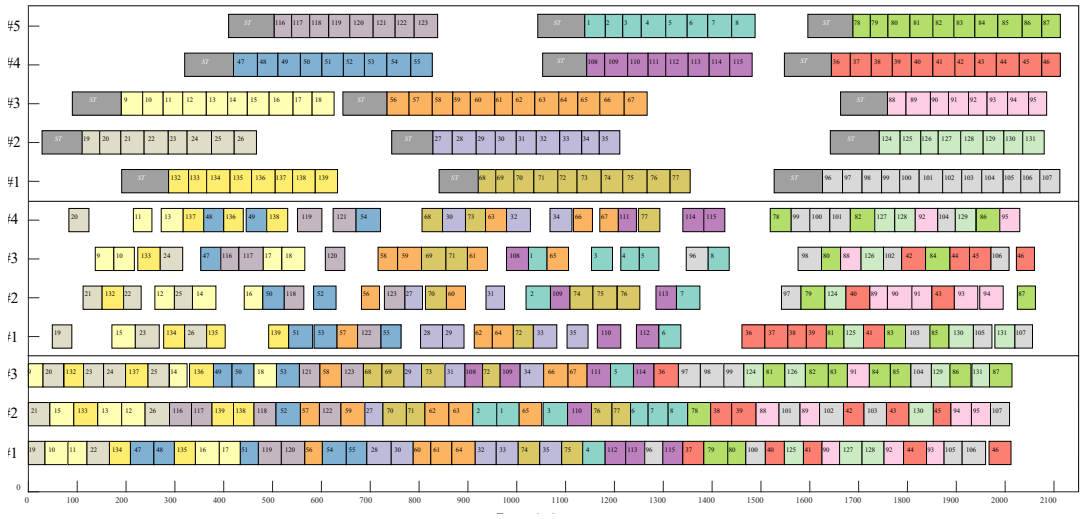


Fig. 10. Scheduling scheme of test date 3×15 (the completion time is 2113, the waiting time is 123).

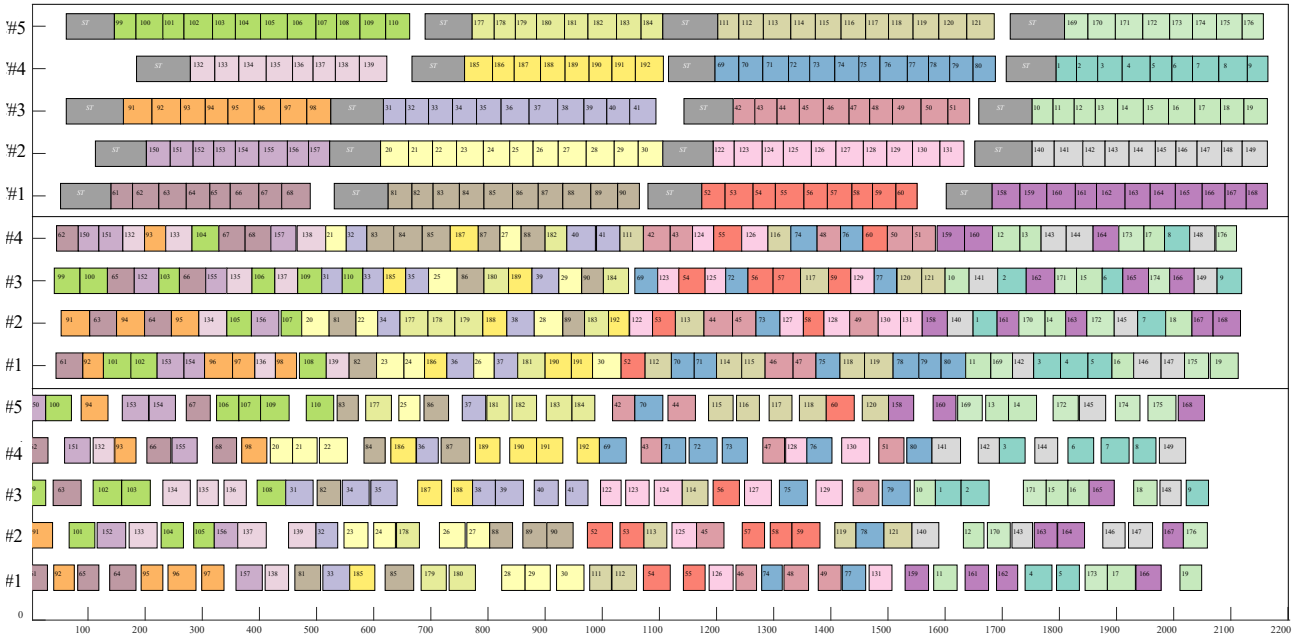


Fig. 11. Scheduling scheme of test date 3×20 (the completion time is 2165, the waiting time is 115)

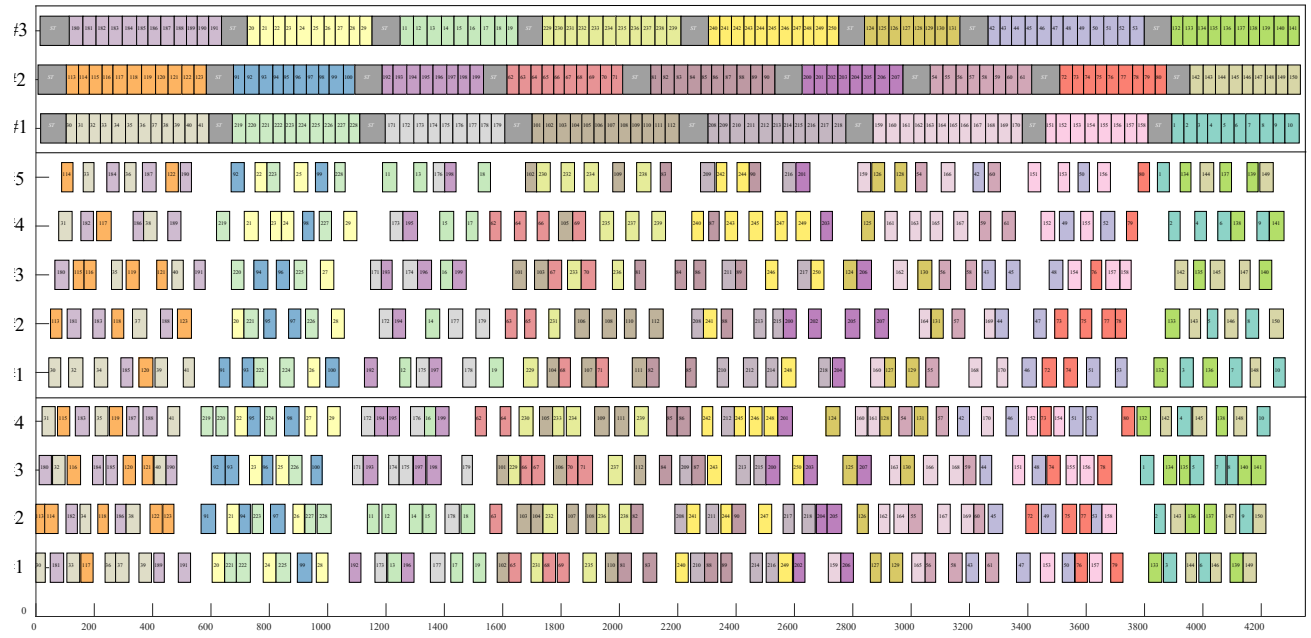


Fig. 12. Scheduling scheme of test date 3×25 (the completion time is 4334, the waiting time is 72)

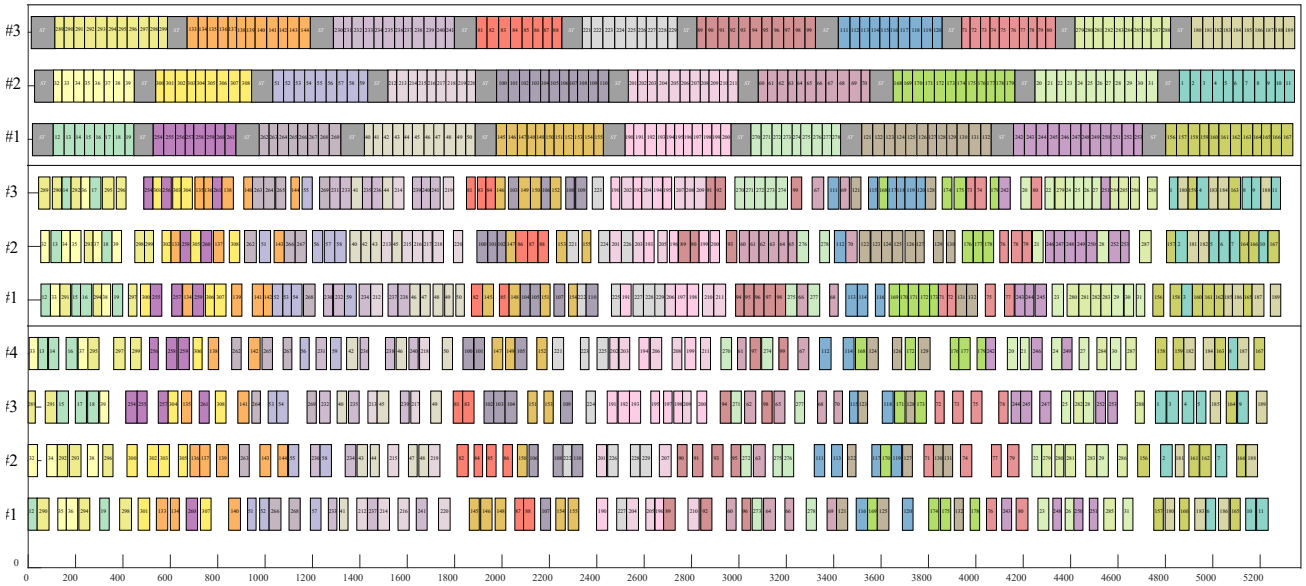


Fig. 13. Scheduling scheme of test date 3×30 (the completion time is 5346, the waiting time is 76)

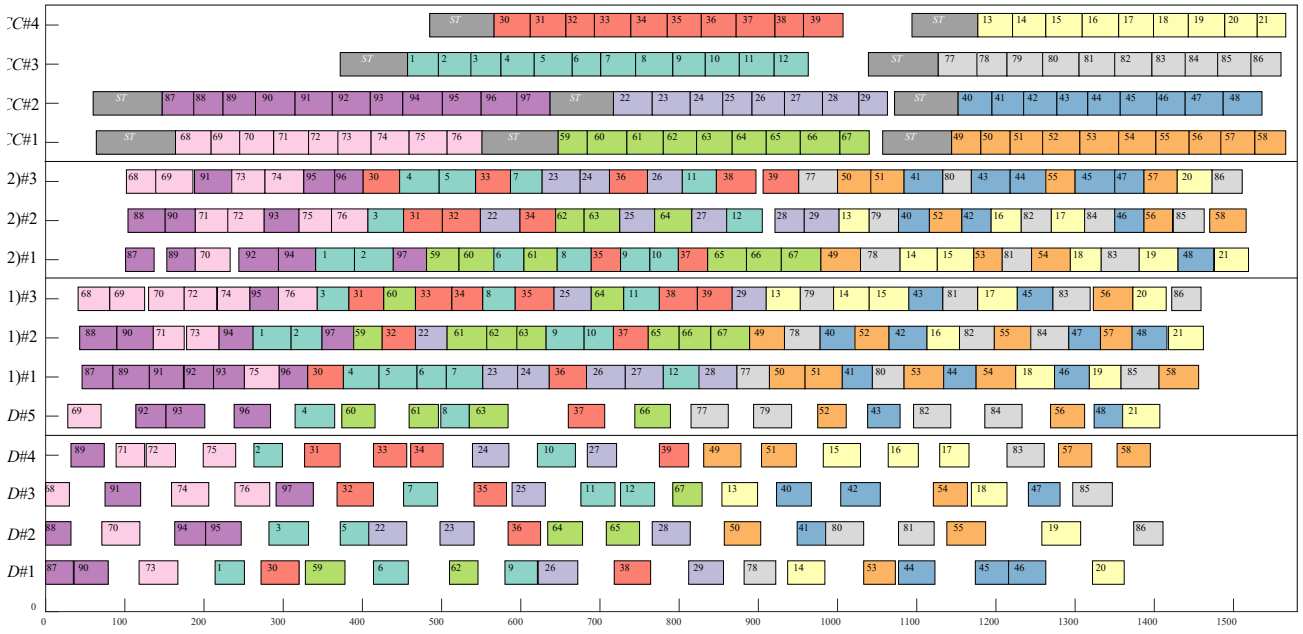


Fig. 14. Scheduling scheme of test date 4×10 (the completion time is 1566, the waiting time is 198)

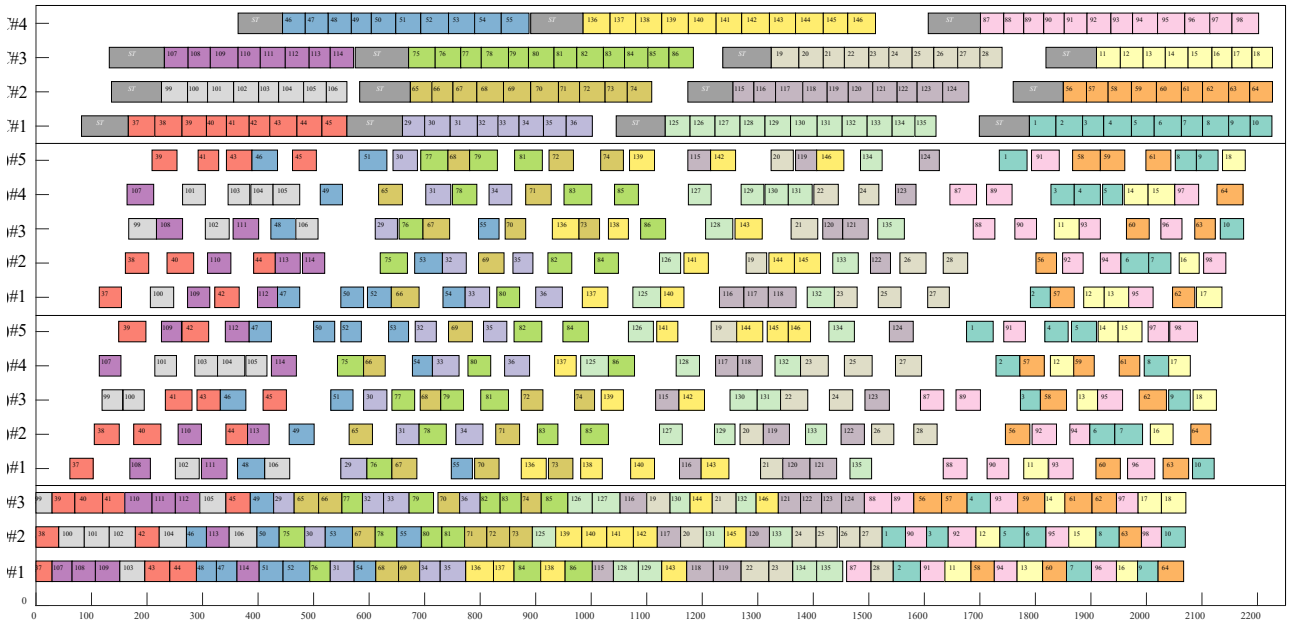


Fig. 15. Scheduling scheme of test date 4×15 (the completion time is 2227, the waiting time is 163)

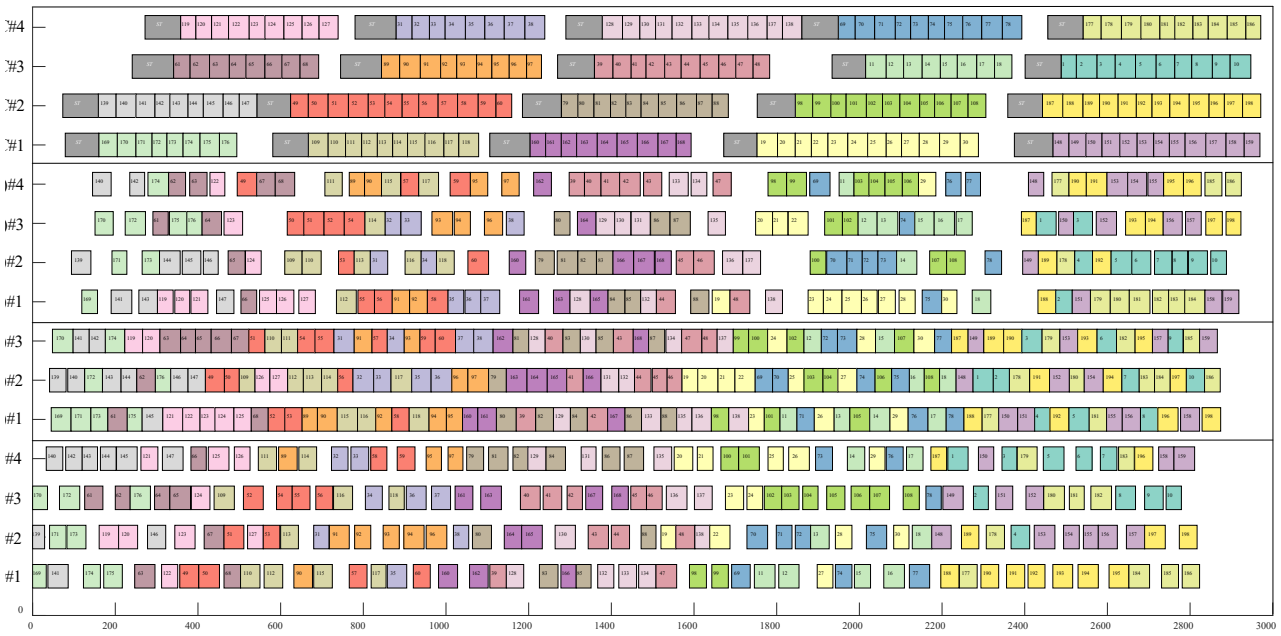


Fig. 16. Scheduling scheme of test date 4×20 (the completion time is 2971, the waiting time is 169)

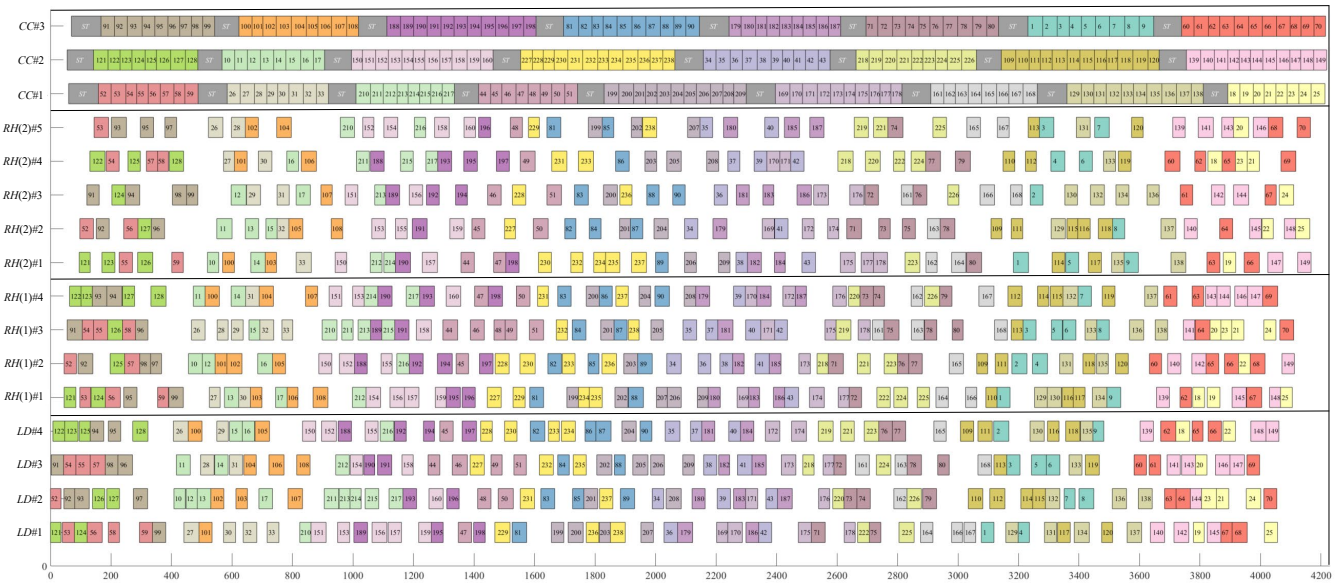


Fig. 17. Scheduling scheme of test date 4×25 (the completion time is 4219, the waiting time is 130)

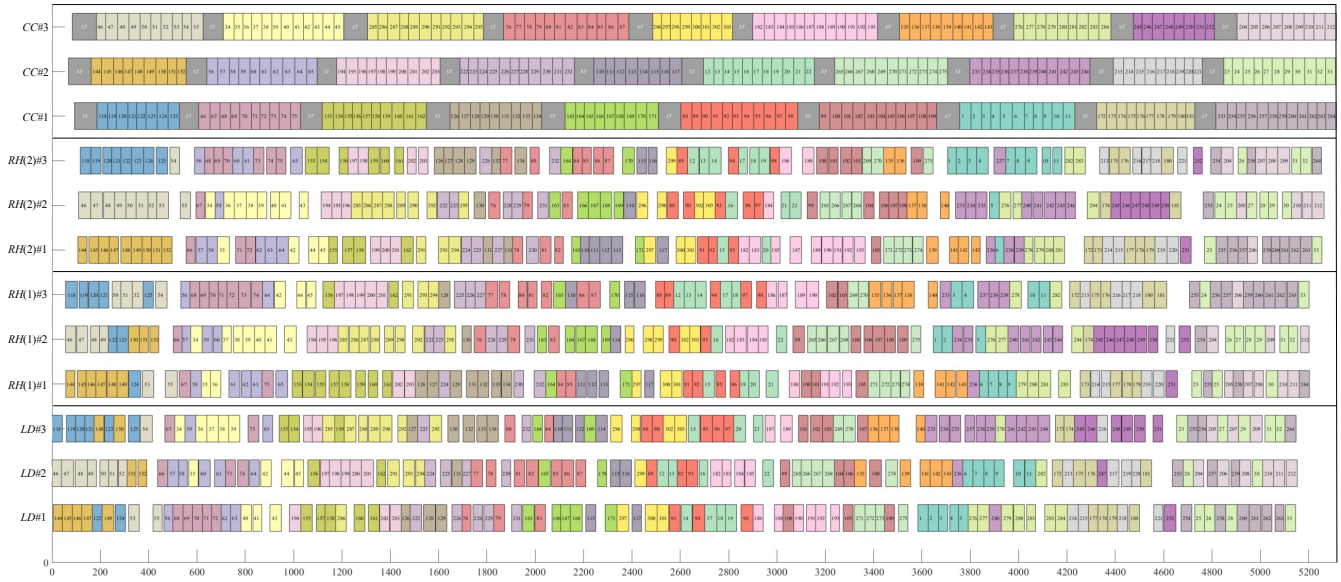


Fig. 18. Scheduling scheme of test date 4×30 (the completion time is 5315, the waiting time is 141)

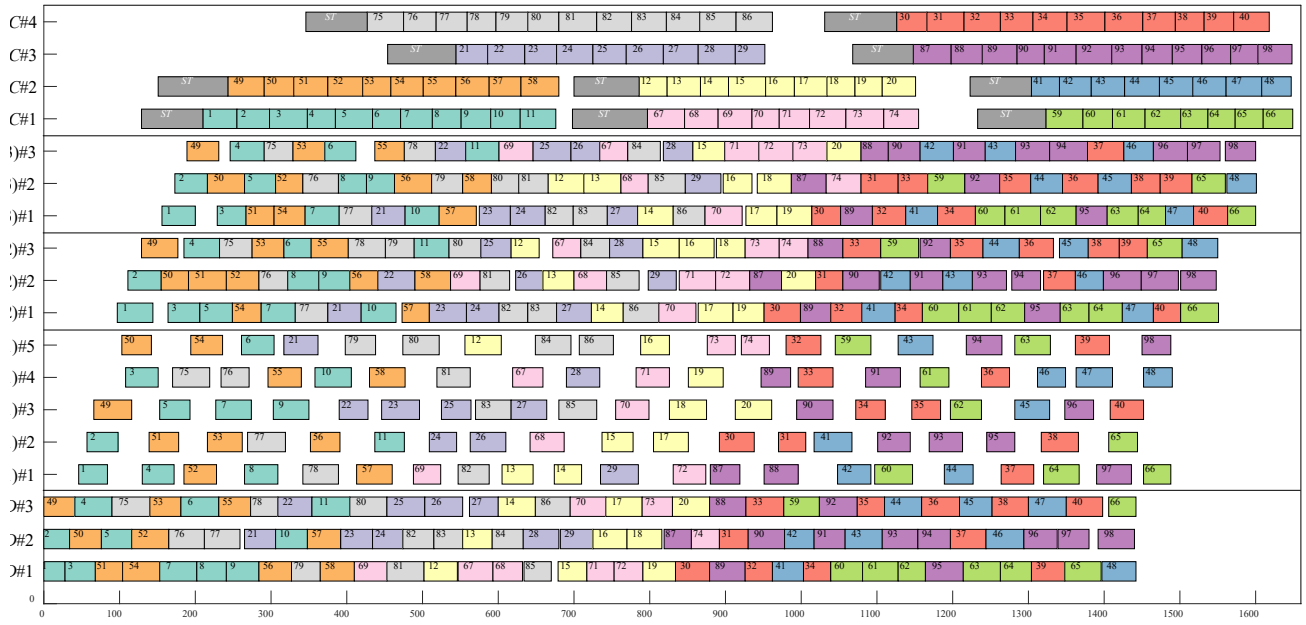


Fig. 19. Scheduling scheme of test date 5×10 (the completion time is 1649, the waiting time is 233)

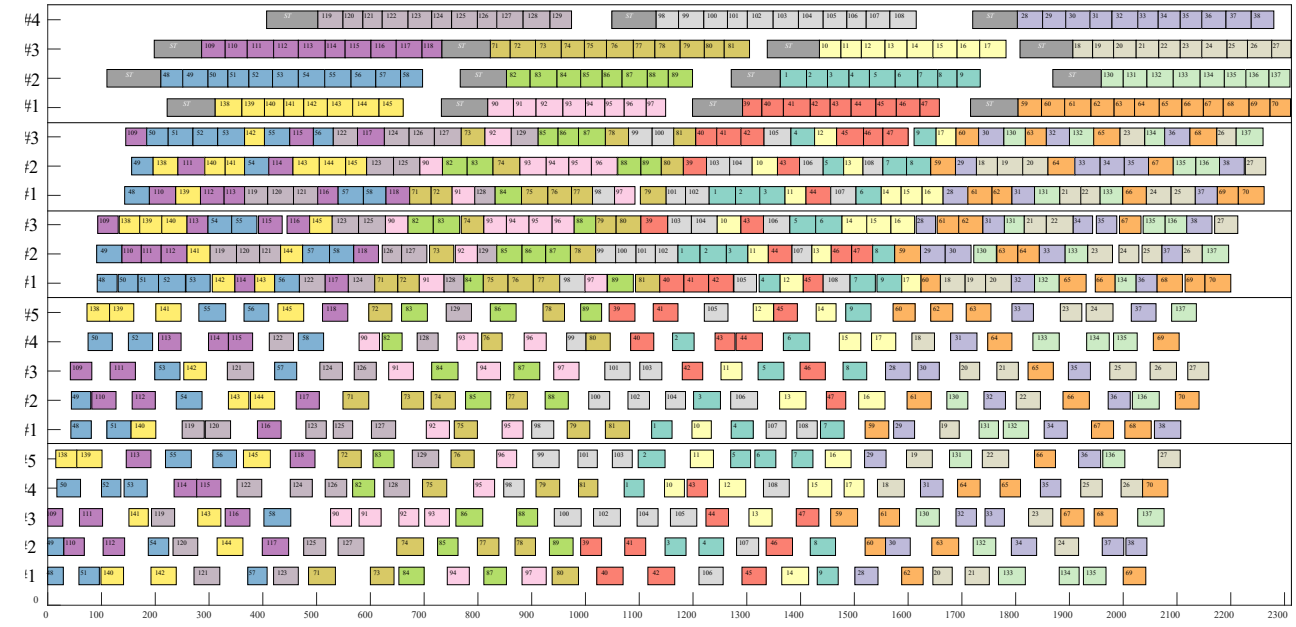


Fig. 20. Scheduling scheme of test date 5×15 (the completion time is 2312, the waiting time is 236)

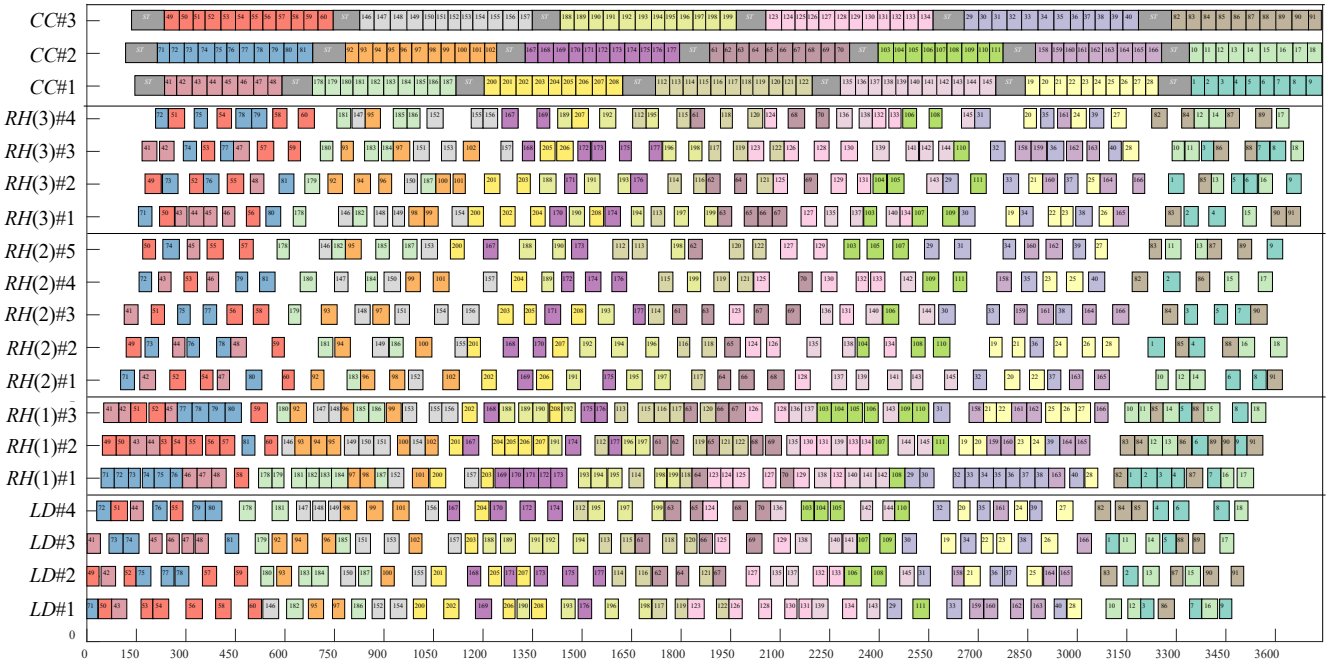


Fig. 21. Scheduling scheme of test date 5×20 (the completion time is 3741, the waiting time is 193)

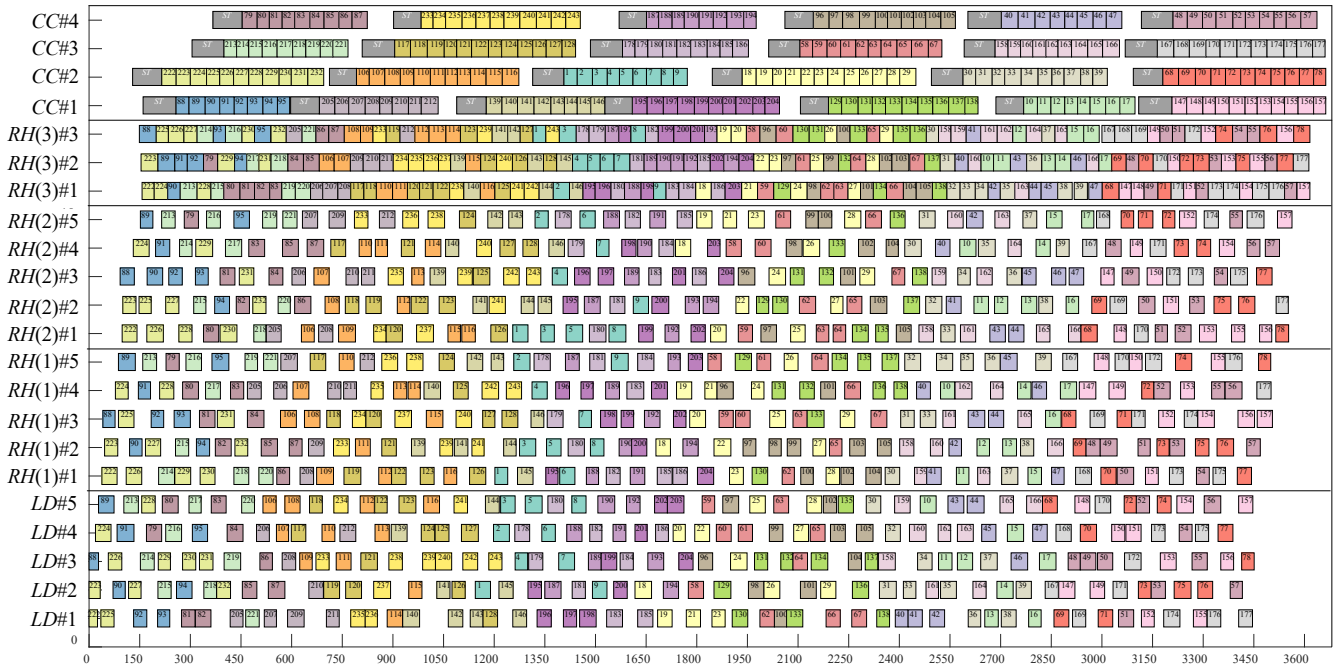


Fig. 22. Scheduling scheme of test date 5×25 (the completion time is 3663, the waiting time is 233)

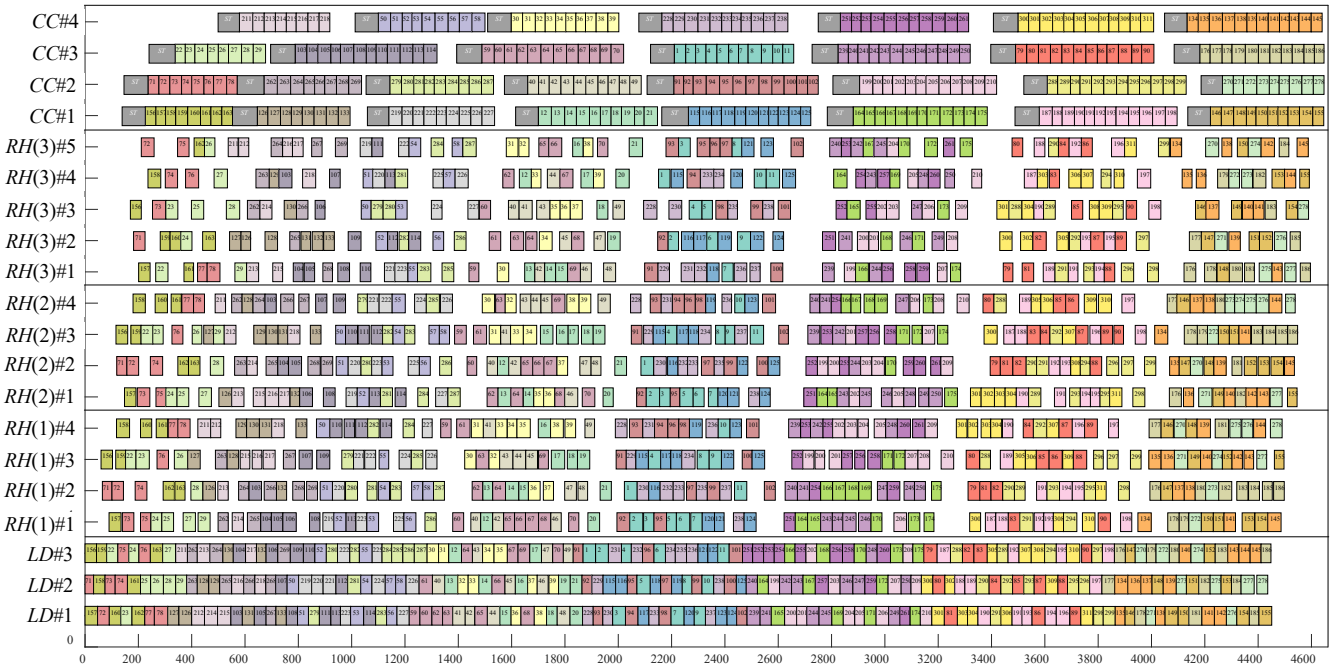


Fig. 23. Scheduling scheme of test date 5×30 (the completion time is 4647, the waiting time is 241)

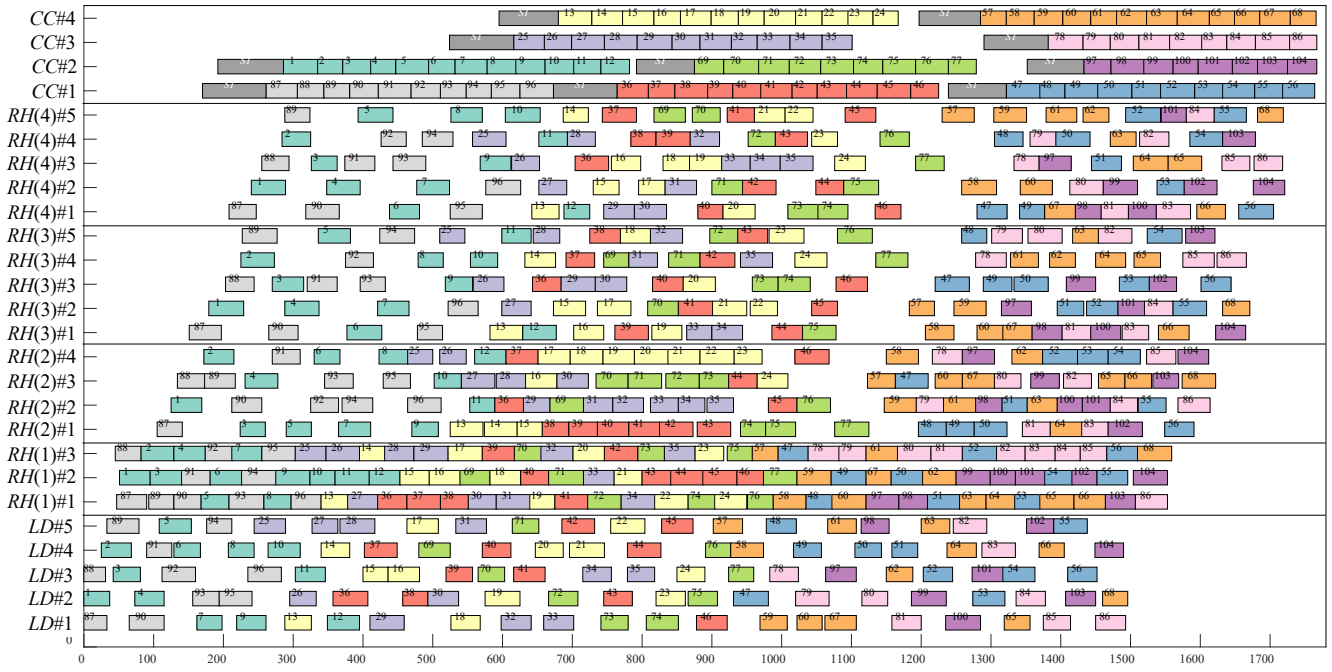


Fig. 24. Scheduling scheme of test date 6×10 (the completion time is 1767, the waiting time is 302)

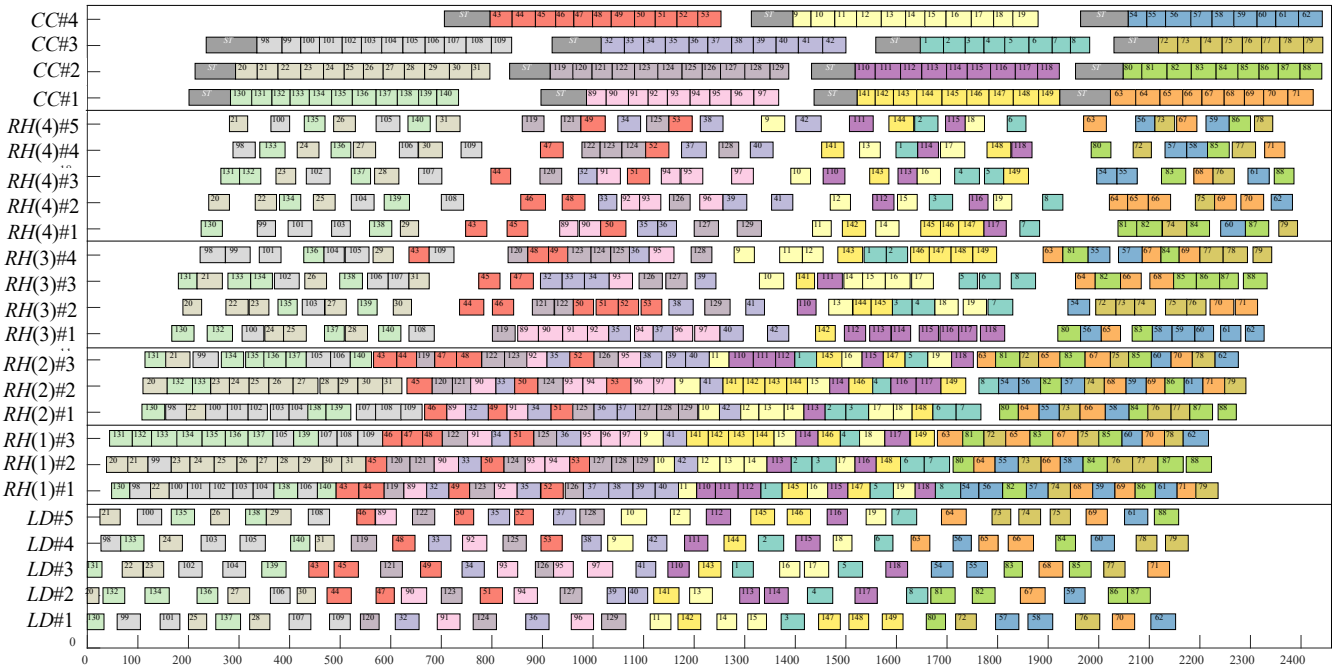


Fig. 25. Scheduling scheme of test date 6×15 (the completion time is 2443, the waiting time is 298)

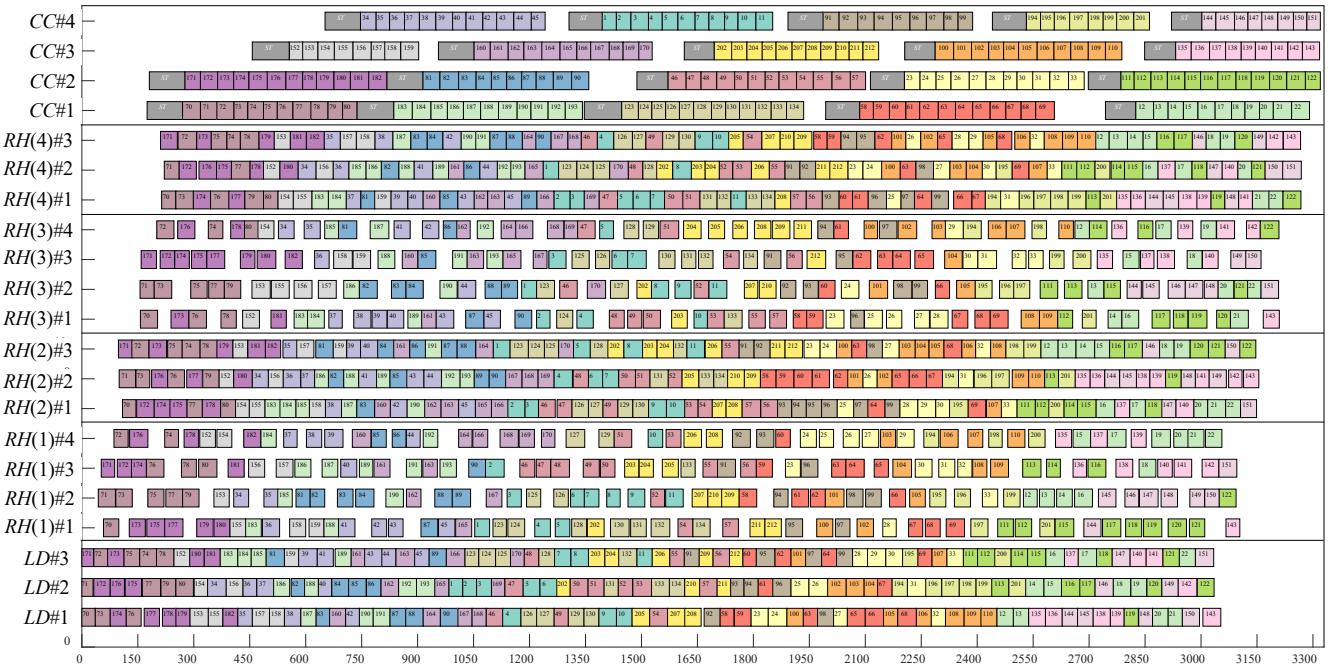


Fig. 26. Scheduling scheme of test date 6×20 (the completion time is 3320, the waiting time is 324)

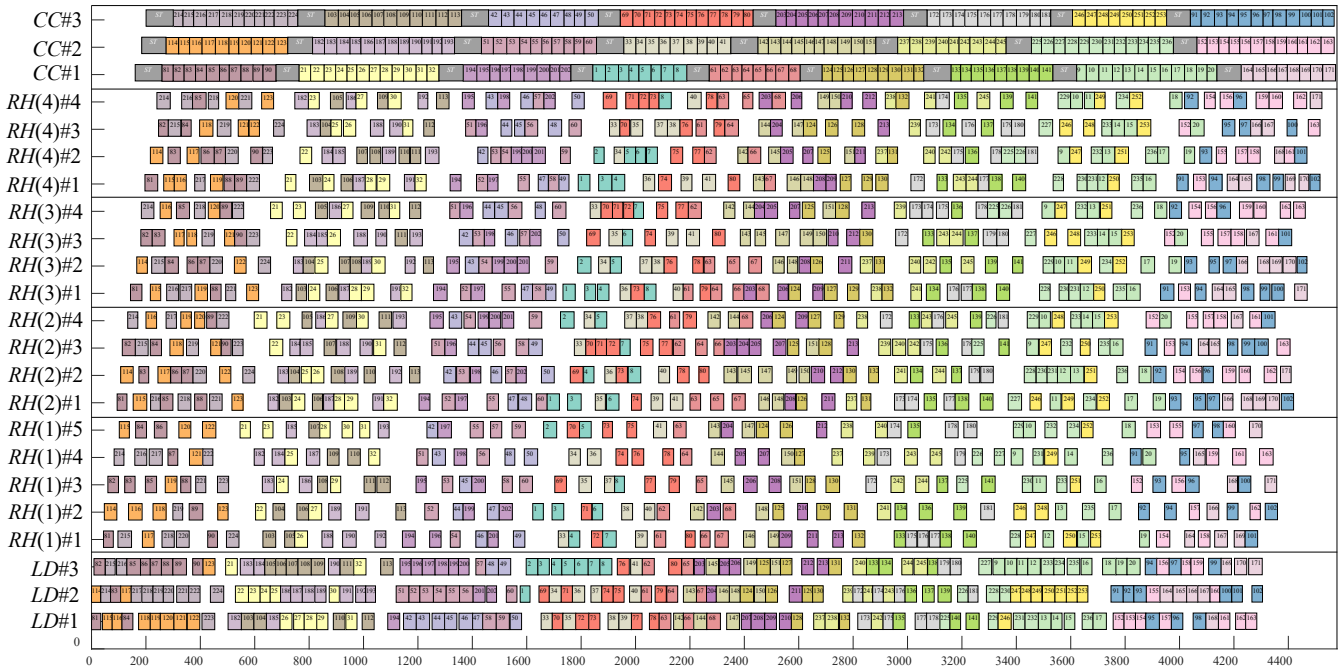


Fig. 27. Scheduling scheme of test date 6×25 (the completion time is 4573, the waiting time is 249)

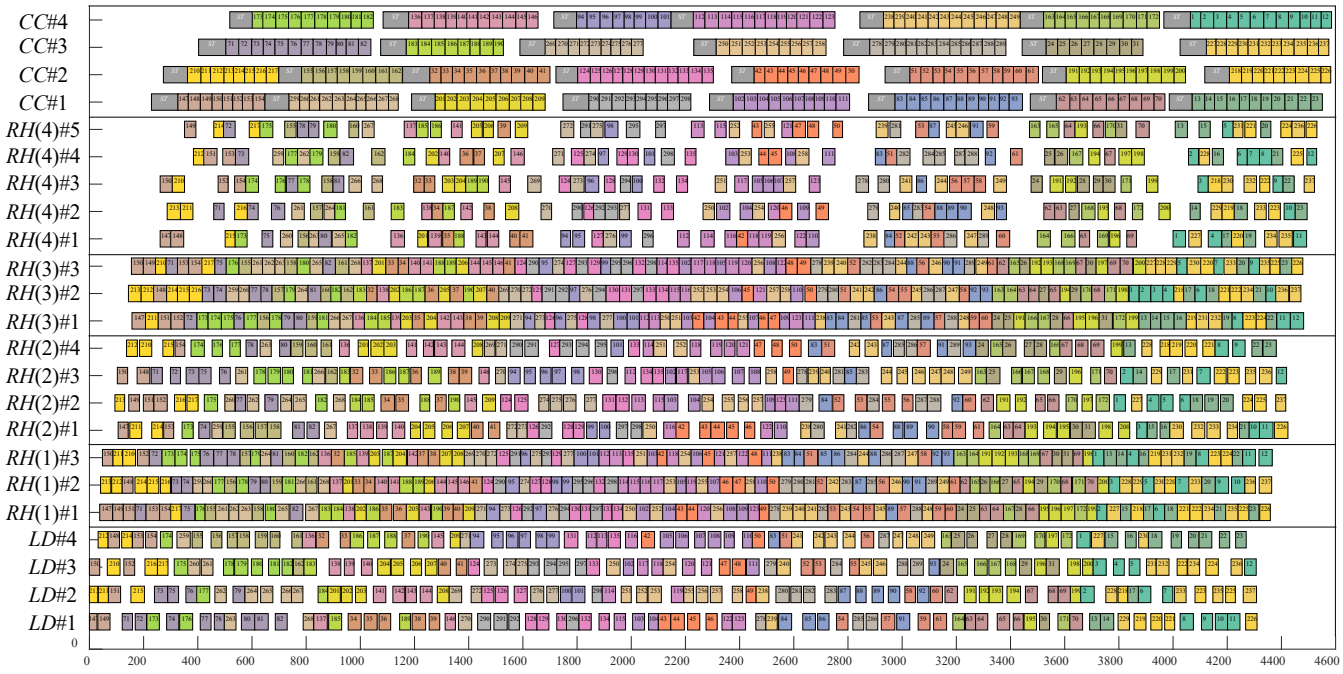


Fig. 28. Scheduling scheme of test date 6×30 (the completion time is 4585, the waiting time is 320)

REFERENCES

[1] S.-p. Yu, and Q.-k. Pan, "A rescheduling method for operation time delay disturbance in steelmaking and continuous casting production process," *Journal of Iron and Steel Research International,* vol. 19, no. 12, pp. 33-41, 2012.

[2] L. Tang, Y. Zhao, and J. Liu, "An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production," *IEEE Transactions on Evolutionary Computation,* vol. 18, no. 2, pp. 209-225, 2013.

[3] J.-q. Li, Q.-k. Pan, K. Mao *et al.*, "Solving the steelmaking casting problem using an effective fruit fly optimisation algorithm," *Knowledge-Based Systems,* vol. 72, pp. 28-36, 2014.

[4] Q.-K. Pan, “An effective co-evolutionary artificial bee colony algorithm for steelmaking-continuous casting scheduling,” *European Journal of Operational Research,* vol. 250, no. 3, pp. 702-714, 2016.

[5] J. Long, Z. Zheng, and X. Gao, “Dynamic scheduling in steelmaking-continuous casting production for continuous caster breakdown,” *International Journal of Production Research,* vol. 55, no. 11, pp. 3197-3216, 2017.

[6] K. Peng, Q.-K. Pan, L. Gao *et al.*, “An improved artificial bee colony algorithm for real-world hybrid flowshop rescheduling in steelmaking-refining-continuous casting process,” *Computers & Industrial Engineering,* vol. 122, pp. 235-250, 2018.

[7] S. Kammammettu, and Z. Li, “Multistage adaptive optimization for steelmaking and continuous casting scheduling under processing time uncertainty,” *IFAC-PapersOnLine,* vol. 51, no. 21, pp. 262-267, 2018.

[8] S.-L. Jiang, Z. Zheng, and M. Liu, “A preference-inspired multi-objective soft scheduling algorithm for the practical steelmaking-continuous casting production,” *Computers & Industrial Engineering,* vol. 115, pp. 582-594, 2018.

[9] W. Bouazza, Y. Sallez, and B. Beldjilali, “A distributed approach solving partially flexible job-shop scheduling problem with a Q-learning effect,” *IFAC-PapersOnLine,* vol. 50, no. 1, pp. 15890-15895, 2017.

[10] İ. Karaoğlan, and S. E. Kesen, “The coordinated production and transportation scheduling problem with a time-sensitive product: a branch-and-cut algorithm,” *International Journal of Production Research,* vol. 55, no. 2, pp. 536-557, 2017.

[11] D. Pacciarelli, and M. Pranzo, “Production scheduling in a steelmaking-continuous casting plant,” *Computers & chemical engineering,* vol. 28, no. 12, pp. 2823-2835, 2004.

[12] L. Wan, and J. Yuan, “Single-machine scheduling to minimize the total earliness and tardiness is strongly NP-hard,” *Operations Research Letters,* vol. 41, no. 4, pp. 363-365, 2013.

[13] P. Brucker, “Scheduling algorithms,” *Journal-Operational Research Society,* vol. 50, pp. 774-774, 1999.

[14] M. Pinedo, “Scheduling: Theory, algorithms, and systems Fifth,” *Cham: Springer International Publishing,* vol. 39, pp. 41, 2016.

[15] S.-Y. Wang, and L. Wang, “An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flow-shop scheduling problem,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 46, no. 1, pp. 139-149, 2016.

[16] N. Mladenović, and P. Hansen, “Variable neighborhood search,” *Computers & operations research,* vol. 24, no. 11, pp. 1097-1100, 1997.

[17] H. R. Lourenço, O. C. Martin, and T. Stützle, "Iterated local search: Framework and applications," *Handbook of metaheuristics*, pp. 129-168: Springer, 2019.

[18] M. Mohmad Kahar, and G. Kendall, “A great deluge algorithm for a real-world examination timetabling problem,” *Journal of the Operational Research Society,* vol. 66, no. 1, pp. 116-133, 2015.

[19] D. Pisinger, and S. Ropke, "Large neighborhood search," *Handbook of metaheuristics*, pp. 99-127: Springer, 2019.

[20] H.-B. Song, and J. Lin, “A genetic programming hyper-heuristic for the distributed assembly permutation flow-shop scheduling problem with sequence dependent setup times,” *Swarm and Evolutionary Computation,* vol. 60, pp. 100807, 2021.

[21] D. Sacramento, D. Pisinger, and S. Ropke, “An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones,” *Transportation Research Part C: Emerging Technologies,* vol. 102, pp. 289-315, 2019.

[22] N. Azi, M. Gendreau, and J.-Y. Potvin, “An adaptive large neighborhood search for a vehicle routing problem with multiple routes,” *Computers & Operations Research,* vol. 41, pp. 167-173, 2014.

[23] Y. Adulyasak, J.-F. Cordeau, and R. Jans, “Optimization-based adaptive large neighborhood search for the production routing problem,” *Transportation Science,* vol. 48, no. 1, pp. 20-45, 2014.

[24] B. Rolf, T. Reggelin, A. Nahhas *et al.*, “Assigning dispatching rules using a genetic algorithm to solve a hybrid flow shop scheduling problem,” *Procedia Manufacturing,* vol. 42, pp. 442-449, 2020.

[25] L. Tang, Y. Zhao, and J. Liu, “An improved differential evolution algorithm for practical dynamic scheduling in steelmaking-continuous casting production,” *IEEE Transactions on Evolutionary Computation,* vol. 18, no. 2, pp. 209-225, 2014, 2014.

[26] Li, Q. Pan, and K. Mao, “A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems,” *IEEE Transactions*

on Automation Science and Engineering, vol. 13, no. 2, pp. 932-949, 2016.

[27] B. Naderi, M. Zandieh, A. K. G. Balagh *et al.*, "An improved simulated annealing for hybrid flowshops with sequence-dependent setup and transportation times to minimize total completion time and total tardiness," *Expert systems with Applications,* vol. 36, no. 6, pp. 9625-9633, 2009.

[28] R. Ruiz, and C. Maroto, "A genetic algorithm for hybrid flowshops with sequence dependent setup times and machine eligibility," *European journal of operational research,* vol. 169, no. 3, pp. 781-800, 2006.

[29] M. Marichelvam, M. Geetha, and Ö. Tosun, "An improved particle swarm optimization algorithm to solve hybrid flowshop scheduling problems with the effect of human factors–A case study," *Computers & Operations Research,* vol. 114, pp. 104812, 2020.

[30] H.-X. Qin, Y.-Y. Han, B. Zhang *et al.*, "An improved iterated greedy algorithm for the energy-efficient blocking hybrid flow shop scheduling problem," *Swarm and Evolutionary Computation*, pp. 100992, 2021.