2023 IEEE Symposium Series on Computational Intelligence (SSCI)
Mexico City, Mexico. December 5-8, 2023

Energy-efficient Hot-rolling Scheduling of High-quality Steel Products

Ziyan Zhao*, Zikuo Bian*, Chenglong Wang†, Kun Zou*, Shixin Liu*‡
* College of Information Science and Engineering, Northeastern University, Shenyang, 110819, China
† Shandong Iron & Steel Group Rizhao Co., Ltd, Rizhao, 276800, China
‡ The State Key Laboratory of Synthetical Automation for Process Industries, Shenyang, 110819, China
zhaoziyan@mail.neu.edu.cn

***Abstract*—Steel production involves many energy-intensive processes. Under the goal of carbon peak and carbon neutrality, it is essential to study the steel production scheduling problems aiming at energy saving to realize the green manufacturing of steel production processes. Aiming at the hot rolling process of high-quality steel products, a novel energy-saving production scheduling problem is studied in this paper. Unlike existing research, this paper additionally considers temperature constraints and the optimization of temperature-keeping equipment assignment in producing high-quality steel products. To solve it efficiently, this paper presents an improved simulated annealing algorithm where destruction and construction strategies from iterated greedy algorithms are embedded into it. Experimental results show that the presented algorithm has obvious advantages compared with other competitive peers. Its excellent solution performance means its great application potential.**

***Index Terms*—Green manufacturing, production scheduling, temperature-keeping equipment assignment, simulated annealing algorithm, destruction and construction.**

I. Introduction

Steel production is a basic industry with complex production systems [1]. Many processes during production, such as steelmaking and hot rolling, are energy-intensive. Under the pressure of energy and environment on a global scale, achieving energy saving and efficiency improvement in the production process is an important demand of the steel industry [2]. In steel production processes [3], steelmaking is responsible for the smelting of molten steel. The molten steel after smelting is solidified into slabs (at high temperature) by a continuous casting machine and crystallizer in a continuous casting process. The slabs are then processed into steel strips of given specifications by a rolling mill in a hot rolling process [4]. A hot rolling process requires the slabs to reach a certain temperature before entering a rolling mill. Under ideal conditions, the slabs produced by continuous casting are processed directly into a hot rolling mill in turn, which can make full use of heat energy [5]. However, due to the constraints of a hot rolling process, hot slabs obtained from continuous casting often cannot be directly into the billet in order, resulting in a gradual decline in the temperature of some slabs. Those not reaching the required temperature need to go through a heating furnace before entering a rolling mill, consuming a lot of energy. Therefore, hot rolling scheduling aims to optimize the processing sequence of slabs under the premise of meeting the process constraints to achieve the goal of energy saving and efficiency improvement.

For some high-quality steel products, such as stainless ones, the above process is more complex. They require that the slabs produced by a continuous casting machine must be kept above a certain temperature before entering a rolling mill. Otherwise, their microstructure and properties would change, resulting in unqualified products. For such high-quality steel products, in order to maintain the temperature, it is necessary to add temperature-keeping equipment between continuous casting and hot rolling processes. Temperature holding hoods and soaking pits are typically equipment. The former is simply to cover the hot slabs to slow down the speed of temperature drop, which is characterized by that the insulation process does not consume additional energy but the insulation time is limited. The latter is usually heated by burning gas, which is characterized by continuous heat preservation but additional energy consumption. Therefore, for the hot-rolling scheduling of such high-quality steel products, additional consideration is needed to optimize the assignment of temperature-keeping equipment for each slab to further achieve energy saving and efficiency improvement in the production process. A basic hot-rolling scheduling problem can be reduced to a traveling salesman problem, which is a well-known NP-hard problem that cannot be solved precisely in polynomial time. Existing research mostly presents meta-heuristic algorithms to solve it [6]–[9]. The complexity of a hot-rolling scheduling problem of high-quality steel products considering the optimization of temperature-keeping equipment assignment is further increased. Thus, it is also NP-hard.

Hot rolling is a typical process of equipment performance degradation due to the gradual wear of rollers. When the rolls wear to a certain extent, they need to be changed by new ones. The hot-rolling production between two roller changes is called a rolling unit. Hot-rolling scheduling problems can be divided into the ones in a single/multiple rolling unit(s).

Hot-rolling scheduling in a single rolling unit optimizes the slab production sequence within it, which can be translated into a traveling salesman problem and its variants. Kosiba et al. [10] study such a problem for the first time, transform it into a traveling salesman problem, and adopt the precise algorithm proposed by Miller and Pekny [11] to solve it.

Tang and Wang [12] study a two-stage hot-rolling scheduling problem, which considers joint scheduling of reheating furnace and rolling mill. To solve it, the authors propose a scatter search algorithm for the first stage and a decision tree-based heuristic algorithm for the second stage. Li and Tian [13] study a dual-objective optimization problem for the joint scheduling of reheating furnaces and rolling mills. The two objectives of minimizing the unnecessary heating time in a heating furnace and minimizing the total switch cost in a hot rolling process are considered. To solve it, an improved multi-objective differential evolution algorithm is designed.

Hot rolling scheduling in multiple rolling units simultaneously optimizes the slab batching, slab production sequence within each rolling unit, and the production sequence of rolling units. It can be translated into a multiple traveling salesman problem [14] or a vehicle routing problem [15]. Tang et al. transform such a problem into a multiple traveling salesman problem with the goal of minimizing the total switch cost and solve it by designing an improved genetic algorithm [14]. Zhang et al. study a hot-rolling scheduling problem of compact steel strips [16]. The authors transform it into a variant of the multiple traveling salesman problem, and design a method combining fruit fly optimization algorithm and variable neighborhood search to solve it. To handle the problem of steel grade jump of continuous rolling slabs, Chen et al. study a hot-rolling scheduling problem of compact steel strips with virtual slab considered [17], and designed a two-stage method to solve it. In the first stage, a precise method is used to solve the problem of slab batching. In the second stage, an artificial bee colony algorithm is designed to solve the scheduling problem.

Existing research does not take into account the hot rolling process of high-quality steel products with temperature constraints and multiple temperature-keeping equipment. Aiming at this process, this paper studies an energy-saving hot rolling scheduling problem to minimize the number of rolling units and the energy consumption for temperature keeping. It makes the following contributions:

1) It proposes a novel energy-efficient hot-rolling scheduling problem of high-quality steel products. It considers the scheduling in multiple rolling units. In addition to optimizing three subproblems, i.e., slab batching, slab production sequence within each rolling unit, and the production sequence of rolling units, the optimization of temperature-keeping equipment assignment is also optimized under temperature constraints.
2) It designs an improved simulated annealing algorithm for solving the concerned problem by introducing destruction and construction strategies inspired by an iterated greedy algorithm [18].

The rest of the paper is organized as follows. Problem descriptions are given in Section II. An improved simulated annealing algorithm is designed in Section III. Experimental results are shown and analyzed in Section IV. Conclusions and future research issues are given in Section V.

II. Problem Statement

A. Problem statement

This study investigates a hot-rolling scheduling problem with the following characteristics. I jobs (i.e., slabs) can be divided into at most J batches (i.e., rolling units) for production. The number of used batches and the slabs included in each of them are to be determined. The key to the hot-rolling scheduling problem lies in the sequencing of the slabs to be scheduled. The machine can process only one slab at a time. Slab processing is non-preemptive, which means that once a slab is being processed, its priority cannot be modified by other slabs. The width of each of slab i is denoted as d_i, which may be different from other slabs. A rolling unit can be divided into warm-up and main-body parts. In the former, the slabs are arranged from narrow to wide. In the latter, the slabs are arranged from wide to narrow. Typically, a hot-rolling scheduling problem does not consider the slabs in a warm-up part since there are few slabs in warm-up material [2]. Thus, in this work, we only consider the scheduling of slabs in the main-body part.

When the width of a freshly processed slab is smaller than the upcoming slab to be processed, or when the number of slabs with decreasing widths in a rolling unit reaches the maximum value limit $\mathcal{K}$, a roller change operation is required. Slabs have different release time from the upstream processes, denoted as r_i, and rolling time p_i. In actual production, a hot-rolling mill cannot handle slabs as their released time due to complex process constraints, resulting in the high-temperature slabs released from the upstream processes needing to wait. During the waiting period, the temperature of them gradually decreases, while the hot rolling process has temperature restrictions. Therefore, a decision needs to be made regarding the temperature-keeping equipment for the high-temperature slabs in waiting. The available temperature-keeping equipment options are temperature holding hoods and soaking pits. Different temperature-keeping equipment assignment patterns are formed based on the waiting time of the slabs before reaching the hot rolling mill, including two types: rolling after temperature keeping in a temperature holding hood and rolling after temperature keeping in a soaking pit as shown in Fig. 1. The characteristics of the former are that it is energy-efficient but has an upper limit on temperature-keeping time, denoted as U. The characteristics of the latter are that it consumes energy but has no upper limit on the temperature-keeping time. Let h_i represent the temperature-keeping time of the i-th slab in a soaking pit, where a larger h_i indicates higher energy consumption. Therefore, h_i is positively correlated with energy consumption. The total temperature-keeping time of the slabs in a soaking pit is used as a measure of the overall energy consumption.

The concerned hot-rolling scheduling problem determines the rolling sequence of each slab, i.e., determining in which rolling unit and at which position within it a slab should be rolled, and assigns temperature-keeping equipment for the slabs. Two objective functions are considered in this study.

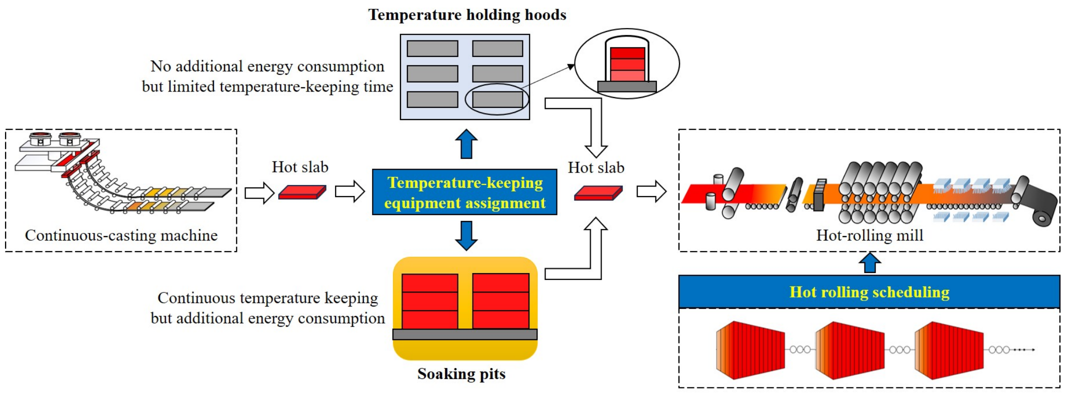
Temperature holding hoods
No additional energy consumption but limited temperature-keeping time
Hot slab
Temperature-keeping equipment assignment
Hot slab
Continuous-casting machine
Hot-rolling mill
Hot rolling scheduling
Continuous temperature keeping but additional energy consumption
Soaking pits

Fig. 1. Illustration of the concerned problem

The first one (f_1) is to minimize the total number of rolling units. The second one (f_2) is to minimize the total energy consumption used for temperature keeping.

Two objective functions considered in this work are treated by weighted summation as the sole one. Before doing it, we need to normalize the objective functions. The overall weighted objective function is defined as:

$$f = \alpha \left[\frac{f_1 - f_{1min}}{f_{1max} - f_{1min}} \right] + \beta \left[\frac{f_2 - f_{2min}}{f_{2max} - f_{2min}} \right] \quad (1)$$

The values of f_{1min}, f_{1max}, f_{2min}, and f_{2max} are determined by considering one objective function separately while temporarily disregarding the other one. This approach allows us to find the minimum value for one objective function while simultaneously finding the maximum value for the other objective function. Let us take f_1 as an example. When the objective function only includes f_1, the obtained objective function value is defined as f_{1min}. Meanwhile, we can calculate f_{2max} by substituting the solution that minimizes f_1 into f_2.

III. Algorithm Design

This work proposes an enhanced simulated annealing algorithm (referred to as SA_DC) for energy-efficient hot rolling scheduling problems. The framework of the presented SA_DC is given in Algorithm 1. It incorporates destruction and construction strategies arising from iterated greedy algorithms [18]–[21] into a simulated annealing algorithm (SA) to further improve its global search ability. It constitutes of encoding and decoding, initialization, neighborhood search, and destruction and construction strategies. They are separately introduced as follows.

A. Encoding and Decoding

During the encoding and decoding process, the encoding part transforms the solution space of the actual problem into a data format suitable for algorithmic solving, while the decoding part converts the algorithm-generated results into a readable form. It is an important step in metaheuristic algorithms. In the context of the simulated annealing algorithm, a permutation-based encoding method is suitable, given that the core of the hot rolling scheduling problem is to solve a sorting problem of the slabs to be scheduled. Hence, we can sort the slabs to be scheduled according to their production sequence and use the sorted result as the encoding part of the algorithm. For example, for a hot rolling scheduling problem involving four slabs, an encoding $\pi = \langle [3,1], [2,4] \rangle$ represents slab 3

Algorithm 1: Outline of SA_DC

```
Input: Initial solution $\pi_0$, Objective function value of
       the initial solution $S_0$, Parameter $T_0$, $T_{end}$, $\gamma$,
       $I_1$, $I_2$
Output: Best solution $\pi_{best}$, Objective function value
        of the best solution $S_{best}$
1  $\pi_{best} \leftarrow \pi_0$;
2  $S_{best} \leftarrow S_0$;
3  for $m = 0$ to $I_1$ do
4      for $z = 0$ to $I_2$ do
5          Generate a neighborhood solution $\pi'$;
6          Decode the neighborhood solution and generate
             a neighborhood solution objective function
             value $S'$;
7          $\Delta = S' - S$;
8          $l \leftarrow$ A random number between 0 and 1;
9          if $S' \leq S$ then
10             $\pi \leftarrow \pi'$;
11             $S \leftarrow S'$;
12         else
13             if $l \leq \exp \frac{-\Delta}{T_0}$ then
14                 $\pi \leftarrow \pi'$;
15                 $S \leftarrow S'$;
16     if $S \leq S_{best}$ then
17         $S_{best} \leftarrow S$;
18     $\pi \leftarrow$ Destruction&Construction($\pi$);
19     $S \leftarrow$ Decoding $\pi$ to obtain its objective function
         value;
20     $T_0 \leftarrow \gamma T_0$;
21 return $\pi_{best}$,$f(S_{best})$
```

being produced in the first position of the first batch, slab 1 being produced in the second position of the first batch, slab 2 being produced in the first position of the second batch, and slab 4 being produced in the second position of the second batch. First, based on the rolling sequence in the encoding, determine the number of rolling units and the positions of each slab in each batch according to their widths. Next, based on the generated rolling units, calculate the start processing time for each slab. Let $\pi[q]$ represent the slab at the q-th position in the encoding. The calculation method for the start time of $\pi[q]$ is as follows:

$$t_{\pi[q]} = \begin{cases} r_{\pi[q]}, & \text{If } q = 1 \\ \max\left\{r_{\pi[q]}, t_{\pi[q']} + p_{\pi[q]} + \mathcal{R}\right\}, & \text{If rollers are changed between slabs } \pi[q] \text{ and } \pi[q'] \\ \max\left\{r_{\pi[q]}, t_{\pi[q']} + p_{\pi[q']}\right\}, & \text{Otherwise} \end{cases} \tag{2}$$

The value of q' in (2) is taken as $q-1$, which means $q' = q-1$. Given the start time of a slab $\pi[q]$, we can obtain $h_{\pi[q]}$, which represents its temperature-keeping time in a soaking pit.

$$h_{\pi[q]} = \begin{cases} t_{\pi[q]} - r_{\pi[q]}, & \text{If the waiting time exceeds } U \\ 0, & \text{Otherwise} \end{cases} \tag{3}$$

B. Initialization

Initialization is used to generate an initial solution. As the presented algorithm is a single-point-based metaheuristic algorithm, the initialization phase generates only one initial solution. However, determining what kind of initial solution to generate requires careful consideration.

Considering that each slab can only be rolled after being released by upstream processes, we take into account the release time of each slab. during the generation of the initial solution. We sort the slabs in the order of increasing release time. The obtained sequence is used as an initial solution for the algorithm.

C. Neighborhood Search

Neighborhood search is used to find a local optimum within the neighborhood of the current solution. In this context, two types of neighborhoods have been designed: one based on exchange operations, and the other based on insertion operations. When considering the neighborhood based on exchange operations, the neighborhood solutions are obtained by exchanging two elements in the current solution. When considering the neighborhood based on insertion operations, the neighborhood solutions are obtained by taking out one element from the current solution and inserting it into another position.

D. Destruction and construction strategies

A basic simulated annealing algorithm demonstrates high efficiency and effectiveness when solving certain NP-hard problems. Although it has a Metropolitan acceptance rule to improve its global search ability, it is easy to get trapped in local optima. Finding ways to prevent the algorithm from being trapped in local optima is an important consideration. In this study, the destruction and construction strategies from an iterated greedy algorithm are incorporated into a simulated annealing algorithm, effectively assisting in escaping local optima.

1) Destruction: A destruction step is the primary way to help the algorithm escape from local optima by removing a certain number of elements and breaking the current local optimal solution. At this stage, the problem to be considered is how to select the elements to be removed from the current solution (referred to as the destruction strategy). We know that when the width of a just processed slab is smaller than the upcoming slab, or when the number of slabs with decreasing widths in a rolling unit equals the maximum rolling block limit $\mathcal{K}$ for each batch, a roll change operation is required to form a new rolling unit for production. We aim to have the number of slabs in each batch, except the last one, as close as possible to $\mathcal{K}$. However, during the SA algorithm's optimization process, it is prone to get trapped in local optima, which is characterized by a small number of slabs in the majority of hot rolling units. To escape from this situation, we design a destruction strategy. It selects and removes all the slabs in the rolling unit with the fewest number of slabs from the local optimal solution generated by the algorithm (if multiple rolling units have the same minimum number of slabs, randomly choose one of them).

2) Construction: A construction phase involves reintegrating the removed slabs from a destruction phase to form a new solution, and it plays a role in escaping from local optima when combined with a destruction phase. During a construction phase, two issues need to be considered: a) the order in which the removed slabs are selected and inserted, and b) where to insert the selected slabs. In the presented SA_DC, construction is performed in the order of removal, which means following the sequence of slabs from the rolling unit with the fewest number of removed slabs in the destruction phase. A greedy insertion strategy is employed, which selects the best position among all feasible positions for inserting the selected slabs. The best position refers to the one that results in the optimal objective function value after insertion.

The pseudo code for the designed destruction-construction process is presented in Algorithm 2. In the destruction process, the rolling unit with the fewest number of slabs is selected from the current solution π. The removed elements are stored in π^R according to the removal order, while the remaining elements that were not removed are stored in π^D in their original order. During a construction phase, the elements in π^R are reintegrated one by one into π^D at the best possible positions, resulting in a new solution π'. In determining the best position for inserting an element, the following criteria are used: the best position minimizes the objective function value of π^D after insertion. If multiple positions lead to the same minimum objective function value in π^D after insertion, the first position among them is selected. Starting from the reconstructed solution, another neighborhood search is performed to obtain a new local optimal solution.

IV. Experimental Results and Analysis

A. Experimental design

To evaluate the performance of the proposed algorithm, extensive experiments are conducted based on production data from a steel plant. The data consist of 5 instances. The scales of them are $I \in \{10, 20, 30, 40, 50\}$. Five testing algorithms are compared in terms of their performance, including SA algorithm with neighborhood search using exchange (referred to as SA_E), SA algorithm with neighborhood search using insertion (referred to as SA_I), taboo search algorithm with neighborhood search using exchange (referred to as TS_E), taboo search algorithm with neighborhood search using insertion (referred to as TS_I), and the presented SA_DC. The number of iterations for each of these algorithms is set to 200,000. To mitigate the effects of randomness in the algorithms, each testing algorithm is executed five times for each instance. The algorithms are implemented in Python and the experiments are conducted on a laptop computer with 8GB RAM, Intel Core i5-10210U processor (1.60GHz).

B. Results and comparison

Table I provides a comparison of the average value, variance, and running time of four algorithms without destruction and construction strategies, i.e., SA_E, SA_I, TS_E, and TS_I. A smaller average value indicates better algorithm performance, implying a higher overall level of solution quality.

Algorithm 2: Destruction and construction strategies

```
Input: Initial solution $\pi_0$, objective function value of
       the initial solution $S_0$
Output: Improved solution $\pi_I$
1  $\pi^R, \pi^D \leftarrow$ Randomly remove elements from $\pi_0$ that
     belong to the batch with the minimum number of
     slabs. Store the removed elements in $\pi^R$ in the order
     of removal, while the remaining sequence is stored in
     $\pi^D$;
2  $g \leftarrow 0$;
3  for $m = 1$ to $size(\pi^R)$ do
4      $S^* \leftarrow +\infty$;
5      for $z = 1$ to $size(\pi^D) + g$ do
6          $\pi^C = \pi^D$;
7          $\pi^C \leftarrow$insert the $m$th element from $\pi^R$ into the
             $z$th position of $\pi^C$;
8          if $f(\pi^C) \leq S^*$ then
9              $p^* = z$;
10             $S^* = f(\pi^C)$;
11     $\pi_I \leftarrow$insert the $m$th element from $\pi^R$ into the $p^*$h
         position of $\pi^D$;
12 return $\pi_I$;
```

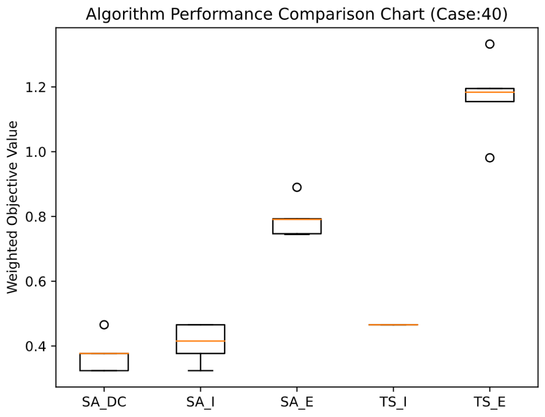

Fig. 2. Box plot of the tested algorithms

A smaller variance indicates better stability in algorithmic solutions. In Table I, we can observe that:
1) A simulated annealing algorithm always shows better performance than a taboo search one with the same neighborhood search operation;
2) An algorithm with an insertion operation always shows better performance than the same algorithm with an exchange operation; and
3) SA_I outperforms other three algorithms.
Although 3), upon examining its solution results, we find that it tends to get trapped in local optima. Therefore, we integrate destruction and construction strategies into it.

Table II compares the solution performance SA_I and SA_DC to evaluate the effectiveness of destruction and construction strategies. The inclusion of the Gap value serves as an evaluation metric to compare the effectiveness of the improved algorithm. The Gap value is defined as follows:

$$Gap = \frac{Obj_{SA\_I} - Obj_{SA\_DC}}{Obj_{SA\_I}} \times 100\% \tag{4}$$

From Table II, it can be observed that:
1) SA_DC yields smaller weighted average values of the objective function than SA_I for each instance scale;
2) The variance is relatively small for SA_DC, except for the instance with a scale of 30. For the other cases, SA_DC exhibits more minor variances compared to SA_I, indicating better solution stability of SA_DC; and
3) One limitation of SA_DC is that it consumes more running time than SA_I for performing destruction and construction operations. However, all instances are solved within a time frame of 51.618 seconds or less, which is sufficiently fast for industrial applications of interest. Therefore, the proposed algorithm exhibits significant potential for practical problem-solving despite the increased computational time.

To visualize the effectiveness of the algorithms, we select an instance with case 40 and compared their results in a boxplot as shown in Fig. 2. From Fig. 2, it is evident that SA_DC demonstrates significantly better performance than other algorithms.

TABLE I. Comparison between SA and TS based on two different neighborhood search methods

Case	SA_E			SA_I			TS_E			TS_I		
	Obj	Deviation	Time/s	Obj	Deviation	Time/s	Obj	Deviation	Time/s	Obj	Deviation	Time/s
10	**0.986**	0	7.43	**0.986**	0	7.43	1.0	0	7.50	1.00	0	7.35
20	0.573	0.000828	12.9	**0.306**	0.00189	12.9	1.09	0.0661	12.9	0.500	0	12.5
30	0.344	0.000326	18.9	**0.320**	0.00138	18.9	0.854	0.0997	18.3	0.435	0.0191	18.1
40	0.793	0.00278	24.1	**0.409**	0.00294	24.3	1.17	0.0126	23.7	0.466	0	23.8
50	0.954	0.00147	30.8	**0.309**	0.00336	32.0	1.04	0.0139	31.0	0.357	0	30.0

TABLE II. Comparison between SA_I and SA_DC

Case	SA_I				SA_DC		
	Obj	Gap to SA_DC	Deviation	Time/s	Obj	Deviation	Time/s
10	**0.986**	0%	0	7.43	**0.986**	0	8.69
20	0.306	24.5%	0.00189	12.9	**0.231**	0.000294	19.6
30	0.320	10.6%	0.00138	18.9	**0.286**	0.00189	36.0
40	0.409	8.80%	0.00294	24.3	**0.373**	0.00269	42.3
50	0.309	7.44%	0.00336	32.0	**0.286**	0.00334	51.6

V. Conclusions and Future Work

This study investigates a new energy-efficient bi-objective hot rolling scheduling problem, considering release time and the total energy consumption of slabs in soaking pits. The objective of the problem is to find a schedule for rolling units of slabs with different energy requirements, while satisfying the production constraints of hot rolling. The two optimization objectives are to minimize the number of rolling units and the total heating time in soaking pits. An improved simulated annealing algorithm based on the destruction and construction strategies is introduced to solve the concerned problem. The experimental results, comparing the performance of five algorithms, demonstrate the high performance of the presented algorithm with substantial data evidence.

In future research, we plan to explore alternative neighborhood search methods, further investigate the mechanism of destruction-construction, and fine-tune algorithm parameters to improve the effectiveness of the algorithm.

Acknowledgment

This work was supported in part by National Key R&D Program of China under Grant No. 2021YFB3301200, National Natural Science Foundation of China under Grant No. 62073069 and 62203093, Guangdong Basic and Applied Basic Research Foundation under Grant No. 2021A1515110827, LiaoNing Revitalization Talents Program under Grant No. XLYC2002041, and Fundamental Research Funds for the Central Universities under Grant No. N2204016.

References

[1] Z. Zhao, X. Yong, S. Liu, and M. Zhou, "Data-driven surplus material prediction in steel coil production," in *2020 29th Wireless and Optical Communications Conference (WOCC)*, pp. 1–6, IEEE, 2020.

[2] A. Özgür, Y. Uygun, and M.-T. Hütt, "A review of planning and scheduling methods for hot rolling mills in steel production," *Computers & Industrial Engineering*, vol. 151, p. 106606, 2021.

[3] Z. Zhao, S. Liu, M. Zhou, X. Guo, and L. Qi, "Decomposition method for new single-machine scheduling problems from steel production systems," *IEEE Transactions on Automation Science and Engineering*, vol. 17, no. 3, pp. 1376–1387, 2020.

[4] Y. Ji, S. Liu, M. Zhou, Z. Zhao, X. Guo, and L. Qi, "A machine learning and genetic algorithm-based method for predicting width deviation of hot-rolled strip in steel production systems," *Information Sciences*, vol. 589, pp. 360–375, 2022.

[5] L. Tang, J. Liu, A. Rong, and Z. Yang, "A review of planning and scheduling systems and methods for integrated steel production," *European Journal of Operational Research*, vol. 133, no. 1, pp. 1–20, 2001.

[6] Z. Zhao, S. Liu, M. Zhou, D. You, and X. Guo, "Heuristic scheduling of batch production processes based on petri nets and iterated greedy algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 1, pp. 251–261, 2022.

[7] Z. Zhao, S. Liu, M. Zhou, and A. Abusorrah, "Dual-objective mixed integer linear program and memetic algorithm for an industrial group scheduling problem," *IEEE/CAA Journal of Automatica Sinica*, vol. 8, no. 6, pp. 1199–1209, 2020.

[8] Z. Y. Zhao, S. X. Liu, and M. C. Zhou, "A new bi-objective batch scheduling problem: NSGA-II-and-local-search-based memetic algorithms," in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 2119–2124, IEEE, 2020.

[9] Z. Zhao, S. Liu, M. Zhou, and X. Guo, "Intelligent scheduling for a rolling process in steel production systems," in *2020 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, 2020.

[10] E. D. Kosiba, J. R. Wright, and A. E. Cobbs, "Discrete event sequencing as a traveling salesman problem," *Computers in Industry*, vol. 19, no. 3, pp. 317–327, 1992.

[11] D. L. Miller and J. F. Pekny, "Exact solution of large asymmetric traveling salesman problems," *Science*, vol. 251, no. 4995, pp. 754–761, 1991.

[12] L. Tang and X. Wang, "A two-phase heuristic for the production scheduling of heavy plates in steel industry," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 1, pp. 104–117, 2009.

[13] K. Li and H. Tian, "Integrated scheduling of reheating furnace and hot rolling based on improved multiobjective differential evolution," *Complexity*, vol. 2018, 2018.

[14] L. Tang, J. Liu, A. Rong, and Z. Yang, "A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron & steel complex," *European Journal of Operational Research*, vol. 124, no. 2, pp. 267–282, 2000.

[15] M. Tan, H.-l. Yang, B. Duan, Y.-x. Su, and F. He, "Optimizing production scheduling of steel plate hot rolling for economic load dispatch under time-of-use electricity pricing," *Mathematical Problems in Engineering*, vol. 2017, 2017.

[16] B. Zhang, Q.-k. Pan, L. Gao, X.-l. Zhang, *et al.*, "A hybrid variable neighborhood search algorithm for the hot rolling batch scheduling problem in compact strip production," *Computers & Industrial Engineering*, vol. 116, pp. 22–36, 2018.

[17] Q. Chen, Q. Pan, B. Zhang, J. Ding, and J. Li, "Effective hot rolling batch scheduling algorithms in compact strip production," *IEEE Transactions on Automation Science and Engineering*, vol. 16, no. 4, pp. 1933–1951, 2019.

[18] Z. Zhao, M. Zhou, and S. Liu, "Iterated greedy algorithms for flow-shop scheduling problems: A tutorial," *IEEE Transactions on Automation Science and Engineering*, vol. 19, no. 3, pp. 1941–1959, 2022.

[19] Z. Zhao, S. Liu, M. Zhou, X. Guo, and J. Xue, "Iterated greedy algorithm for solving a new single machine scheduling problem," in *2019 IEEE 16th Int. Conf. on Networking, Sensing and Control (ICNSC)*, pp. 430–435, 2019.

[20] Z. Zhao, M. Zhou, S. Liu, X. Guo, and H. Liu, "A lexicographic bi-objective scheduling problem from steel production systems," *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 158–163, 2020.

[21] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.