processes

Article

An Improved Line-Up Competition Algorithm for Unrelated Parallel Machine Scheduling with Setup Times

Yuting Xu and Bin Shi *

Department of Chemical Engineering, School of Chemistry, Chemical Engineering and Life Sciences, Wuhan University of Technology, Wuhan 430070, China
* Correspondence: shibin@whut.edu.cn; Tel.: +86-189-8619-5381

Abstract: It is well known that with the development of economic globalization and increasing competition in the market, enterprises are facing a huge challenge in the unrelated parallel machine scheduling problem with setup time (UPMST). Determining the processing order of all jobs and assigning machines to production scheduling has become more complex and has research implications. Moreover, a reasonable production scheduling scheme can not only complete the production plan efficiently but also contribute to reducing carbon emissions. In this paper, a mathematical model with the goal of the shortest completion time is studied for the UPMST problem. An improved line-up competition algorithm (ILCA) is proposed to solve this model, and the search accuracy and rate of the algorithm are improved by the newly proposed heuristic workpiece allocation rules and variation strategies. From the perspective of evaluation purposes, the effectiveness and stability of the method are significantly superior to other methods, and it is competitive in solving the UPMST problem.

Keywords: unrelated parallel machine scheduling; line-up competition algorithms; heuristic rules; variant policies; production scheduling

check for updates

Citation: Xu, Y.; Shi, B. An Improved Line-Up Competition Algorithm for Unrelated Parallel Machine Scheduling with Setup Times. *Processes* **2022**, *10*, 2676. https://doi.org/10.3390/pr10122676

Academic Editor: Jingtao Wang

Received: 8 November 2022
Accepted: 7 December 2022
Published: 12 December 2022

1. Introduction

Production scheduling is the work of an enterprise to produce jobs in strict accordance with the requirements of a production plan [1]. When jobs can be processed on multiple machines with similar functions and neither interfere nor disturb each other during simultaneous processing, such a problem is defined as a parallel machine scheduling problem [2]. There are a large number of manufacturing and service problems in modern industrial production, such as chemical, metallurgical, textile, mechanical, semiconductor, and logistics fields [3]. In this type of practical production, the processing times of artifacts on different machines are generally different, and the processing times on different machines are not correlated with each other, which is the uncorrelated parallel machine scheduling (UPM) problem that widely exists in manufacturing [4]. However, in actual production, many constraints have to be considered, such as storage space, raw material storage, release time, machine preparation time, etc. Generally, common constraints are the ready time, and UPM problems with such constraints are called UPMST problems, which are NP-hard problems.

Solutions to this kind of problem can be roughly divided into exact algorithms and heuristic algorithms. The exact algorithm can obtain the optimal solution, but as the scale of the problem becomes larger, it becomes almost impossible for the exact algorithm to find and obtain the optimal solution. Rakovitis et al. [5] developed a novel mathematical formulation to solve the large-scale instances that existing models fail to solve, and it takes less computation time and consumes less energy. The problem of scheduling parallel machines is solved. In [6], a new logic-based exact algorithm for Benders decomposition was proposed to deal with common extra constraints and various min or max objectives to solve a wide class of parallel machine scheduling problems. A new three-stage algorithm based on an exact mathematical method has been shown to have a competitive advantage in dealing with unrelated parallel machine scheduling with specific and different resources [7]. To abate the problem of the excessive computation time of exact algorithms, promising progress has been made through research. Yunusoglu et al. [8] developed a constraint programming (CP)-based exact solution with the addition of lower bound restrictions and redundancy constraints to enrich the proposed CP model, thereby achieving a reduction in computation time.

Considering the solution time spent, more heuristic algorithms for the solution of this problem can be found in the literature because they are more efficient and stable in finding feasible scheduling solutions within an acceptable time frame. In [9] A new mixed integer linear programming (MILP) was proposed for the single-machine scheduling problem with release time and workpiece sequence conversion time, and developed a beam search heuristi to solve the problem. Finally, a higher-quality solution was obtained with a lower computational cost. Laha et al. [10] proposed a cuckoo search algorithm to solve the identical parallel machine scheduling problem, which is faster and less computationally intensive than the other algorithms mentioned in this article. Arnaout et al. [11] presented a two-stage ant colony optimization (ACO) algorithm that solved a non-preemptive non-correlated parallel machine scheduling problem with computational results that outperformed the taboo search (TS) algorithm. New worm optimization algorithms similar to ACO have also been proposed to solve similar problems [12]. Ying et al. [13] proposed a restricted simulated annealing (RSA) algorithm that incorporates a restricted search strategy to effectively reduce the search effort required to find the best neighborhood solution by eliminating invalid job moves. Avalos-Rosales et al. [14] proposed a metaheuristic algorithm based on a multi-start algorithm and a variable neighborhood descent metaheuristic, which contributes to solving large-scale UPMTS.

In recent years, larger UPM instances have also been solved. Santos et al. [15] studied four different stochastic local search (SLS) methods that computed 1000 large benchmark instances, 901 of which yielded the latest and best solution. Lin et al. [16] proposed a population-based simulated annealing algorithm that considered the practical need for a burn-in (B/I) procedure and finally obtained a better solution than the schedule actually used. Chen et al. [17] used a mixed-taboo search (TS) algorithm to solve the proposed mixed-integer programming (MIP) model and obtained the maximum number of processing jobs in practice with an average of 8 s. Huang et al. [18] proposed an improved firefly algorithm with a summation-learning firefly algorithm with competitive performance in processing UPMSP with sequence-dependent setup times. Fang et al. [19] proposed a hybrid metaheuristic based on LA-ALNS and taboo search (LA-ALNS-TS) and a dynamic perturbation scheme that helps the algorithm get rid of local optimization. The effectiveness and efficiency of the proposed algorithm were evaluated experimentally. It is shown that the hybrid algorithm has the advantages of each and is valuable for the study of large-scale UPM.

Of course, a growing number of scholars consider more than just the proposed new algorithm and further improve the search strategy. In [20], the divided semi-definite programming (D-SDP) algorithm and fast divided semi-definite programming (FD-SDP) algorithm with a tailored local search (LS) strategy were proposed, and solved the parallel machine scheduling problem independent of the release time. Lei et al. [21] proposed a multi-subgroup artificial bee colony (MABC) algorithm to solve problems with the objective of minimizing the makespan and total latency, verifying that the new strategy proposed by MABC is effective. Moser et al. [22] proposed several variants of the simulated annealing algorithm for solving large-scale instances that arise in practice, and also investigated an innovative heuristic move selection strategy for problems with the objective of minimizing the total tardiness and working time of uncorrelated parallel machine scheduling problems have some certain meaningful guidance.

The UPMST problem that has been the concern of scholars in the aforementioned literature is the problem we are concerned with in this paper. Focusing on this UPMST problem, which conforms to the modern production model, we have done the following work. A mathematical model of the UPM problem with setup time is established, which takes the traditional scheduling optimization objective, that is, minimizing the maximum completion time, as the objective function. Based on the original line-up competition algorithm (LCA) [23], a new heuristic artifact assignment rule is proposed to expand the search range of the solution, and the variational strategy is also improved to avoid falling into the local optimum so that the local search and global search are performed simultaneously and the algorithm converges quickly to obtain the approximate global optimum. We have solved the model with the improved line-up competition algorithm (ILCA) and compared the results calculated by examples with the previous ones. Experimentally, it is proved that the proposed algorithm outperforms the other algorithms listed in terms of solution quality and other generalizations, and it is verified that the ILCA's have good effectiveness and population diversity for the UPMST problem. The contributions of this study can be summarized as follows.

(1) An improved queuing competition algorithm is proposed to handle the UPMST problem.
(2) A new heuristic artifact allocation rule is proposed for the research problem.
(3) Different mutation strategies are proposed to execute different mutation strategies by giving different mutation probabilities for families and parents in different positions.
(4) Its effectiveness and performance are verified by improving the algorithm to improve its search accuracy and speed.

The remainder of this paper is organized as follows. Section 2 presents the problem statement and the establishment of the mathematical model. Section 3 describes the proposed solution approach in detail, and Section 4 illustrates the computational experiments, demonstrating the potential of the approach for the UPMST scheduling problem. Finally, Section 5 provides the conclusions and presents future research directions.

2. Problem Formulation

2.1. Problem Description

There are N (i = 1, 2, 3, . . . , N) jobs and M (m = 1, 2, 3, . . . , M) machines that must be processed. Each job goes through S (j = 1, 2, 3, . . . , S) stages. Each processing procedure has at least one machine, and at least one processing procedure has a parallel machine. The processing time of each job on each machine is different; that is, the processing capacity of each machine is different. At each stage, each job must complete a stage, but each processing procedure for each job can be processed. Processed on any machine in the corresponding stage. Knowing the processing time ($PT_{i,m}$) of each job on each machine and the waiting time ($SPT_{i,j,m}$) between jobs on different machines in different stages, it is required to determine the processing sequence of all jobs and the allocation of machines to minimize certain scheduling indicators, such as the maximum completion time (C_{max}). An illustration of this problem is given in Figure 1. In actual production, a workpiece often requires multiple processes for processing, and there are unrelated parallel machines for each process, making this production scheduling problem NP-hard.

In conjunction with the above brief description of the UPMST scheduling problem, the production machines for each job in each process are selected sequentially according to their processing order, thereby determining the start time and completion time of the job on the machine and optimizing the given scheduling objectives. This determines the optimal job plan for the final stage, including the assignment, sequencing, and timing of jobs. Prior to modeling, reasonable assumptions are made based on UPMST production realities.

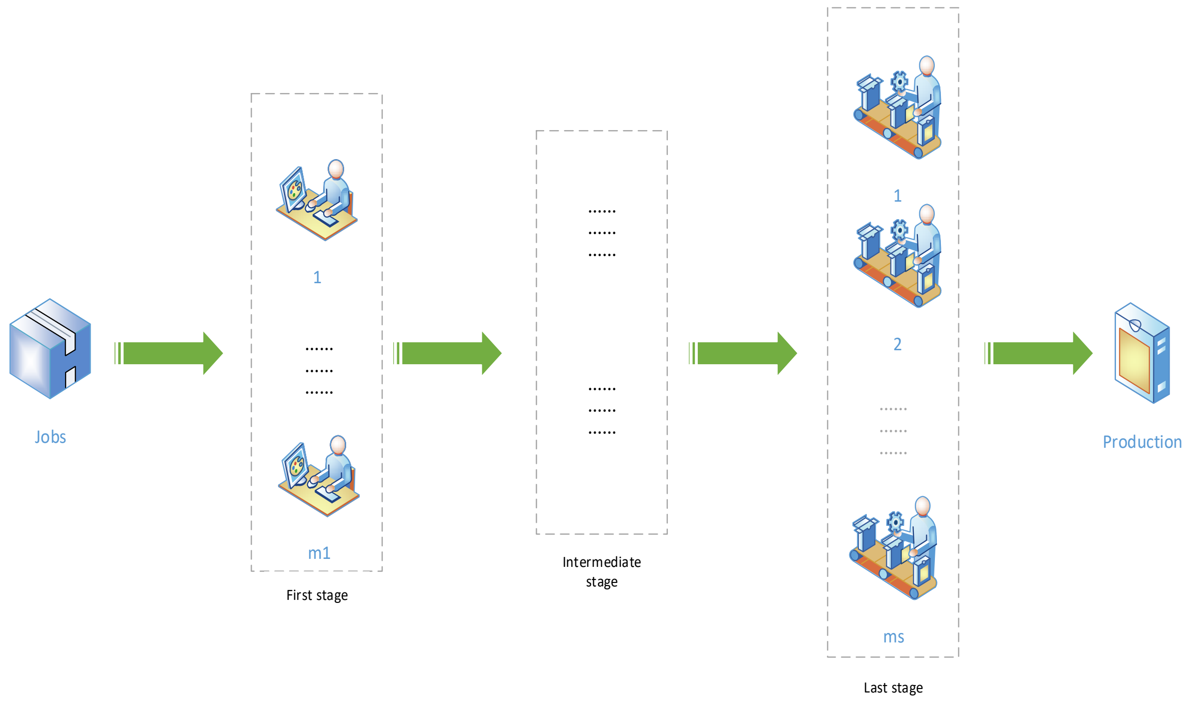


Figure 1. UPMST illustration.

The assumptions are presented as follows:

(1) All processing parameters are deterministic.
(2) The setup time in a machine is independent of job order and properties.
(3) Machine malfunction will not occur.
(4) Unlimited storage capacity between stages.
(5) There are multiple machines in each stage, and the production capacity of each machine is different, which means that the actual processing time of each job is different.
(6) The job cannot be interrupted once machining has started.
(7) A machine can handle only one job at a time.
(8) A job can only be processed on one machine at a time.

2.2. *Mathematical Model*

2.2.1. Objective Function

In this subsection, we provide a mathematical model for the UPMST scheduling problem. Note that this model is an adapted version from [24]. We first need to define symbols to simplify the displayed model. Table 1 shows the symbols used in the mathematical model and explains each symbol.

Table 1. Symbol description.

Nomenclature	
Sets	
N	All jobs
M	Set of all machines
S	Set of operations that jobs need to be processed
M_i	Set of machines that can process i
$M_{s,i}$	Set of machines that can process i at stage s

Table 1. *Cont.*

Nomenclature	
Indices	
i	Jobs
k	Machines
j	Stages
K'	Machines selected with heuristic job assignment rules
Parameters	
MCT_k	Completion time of the most recently processed job on machine k
MCP_k	The last processed job number on machine k
$PUBT_{j,i}$	Virtual release time of job i at stage j
OR_{i}	Release time of job i
$PT_{i,k}$	Processing time of job i on machine k
$SPT_{i,i'}$	Relative sequential transition time between jobs i and i'
$STP_{i,k'}$	Completion time of job i on machine k'
$STA_{i,k'}$	Start time of job i on machine k'
$C_{s,i}$	Completion time of workpiece i on stage s
Variables	
r_s	Job allocation rules used in stage s
P	Production sequence of job
P'	Production sequence of jobs sorted by completion time
$f_{i,k'}$	The most suitable machine is obtained by evaluating each machine of a job according to a job assignment rule

In order to optimize the given scheduling objective, this paper takes minimizing the maximum completion time of the job as the objective function of the uncorrelated parallel machine scheduling problem.

$$min\{max(C_{s,1}, C_{s,2}, \cdots, C_{s,i})\} \tag{1}$$

Equation (1) is the objective function, the maximum completion time is the time that each machine completes the last artifact on the final stage.

2.2.2. Heuristic Order Allocation Rules

In this paper, a series of job allocation rules is proposed to achieve a reasonable allocation of machines according to the characteristics of actual production so that the scheduling goal can be achieved better and faster. The specific description is as follows:

(1) Select the machine with the shortest job sequence change over time.
(2) Select the machine with the shortest job processing time.
(3) Select the machine that can process jobs at the earliest.
(4) Select the machine with the largest processing time.
(5) Select the machine with the largest current time.
(6) Select the machine with the smallest sum of job processing time and current time of the machine.
(7) Select the machine with the maximum sum of job processing time and machine current time.
(8) Select the machine that has the smallest sum of job sequence transition time and processing time.
(9) Select the machine that has the smallest sum of job sequence transition time and current machine time.
(10) Select the machine with the smallest job completion time. In the job allocation process, if a certain rule is used to obtain multiple machinable machines, any one of them can be selected to process the current job.

2.2.3. Job-Machine Allocation Steps for UPMST Scheduling Problems

Based on the above job distribution rules and scheduling objectives, the machines for each job at each stage are selected. Since each production stage is both independent and sequentially related, the multi-stage UPMST scheduling problem is reduced to multiple interconnected single-stage UPMST scheduling problems. We give the processing sequence and scheduling target of the job in the first stage in advance and get a series of scheduling results, such as processing machines, start time, and completion time for each job in each stage, by using the allocation rules. The following Algorithm 1 shows the pseudo-code of the order-machine assignment process.

Algorithm 1: Job-machine assignment process

```
$MCT_k = PRET_k, \forall_k \in M;$
$MCP_k = 0, \forall_k \in M;$
$PUBT_{1,i} = OR_i, \forall_i \in N;$
for j = 1 : S
    for i = 1 : N
        switch($r_s$)
            case1 : $f_{P_i,k'} = min\{SPT_{MCP_k,P_i}\}, \forall_k \in M_{j,P_i};$
            case2 : $f_{P_i,k'} = min\{PT_{P_{i,k}}\}, \forall_k \in M_{j,P_i};$
            case3 : $f_{P_i,k'} = min\{MCT_k\}, \forall_k \in M_{j,P_i};$
            case4 : $f_{P_i,k'} = max\{PT_{P_i,k}\}, \forall_k \in M_{j,P_i};$
            case5 : $f_{P_i,k'} = max\{MCT_k\}, \forall_k \in M_{j,P_i};$
            case6 : $f_{P_i,k'} = min\{PT_{P_i,k} + MCT_k\}, \forall_k \in M_{j,P_i};$
            case7 : $f_{P_i,k'} = max\{PT_{P_i,k} + MCT_k\}, \forall_k \in M_{j,P_i};$
            case8 : $f_{P_i,k'} = min\{SPT_{MCP_k,P_i} + PT_{P_i,k}\}, \forall_k \in M_{j,P_i};$
            case9 : $f_{P_i,k'} = min\{SPT_{MCP_k,P_i} + MCT_k\}, \forall_k \in M_{j,P_i};$
            case10 : $f_{P_i,k'} = min\{SPT_{MCP_k,P_i} + PT_{P_i,k} + MCT_k\}, \forall_k \in M_{j,P_i};$
        end
        $STP_{P_i,k'} = max\left(SPT_{MCP_k,P_i} + MCT_k, PUBT_{j,P_i}\right) + PT_{P_i,k'};$
        $STA_{P_i,k'} = STP_{P_i,k'} - PT_{P_i,k'};$
        $MCT_{k'} = STP_{P_i,k'};$
        $MCP_{k'} = P_i;$
        $C_{j,P_i} = STP_{P_i,k'};$
    end
    $P' = \text{sort}\left(C_{j,1}, C_{j,2}, C_{j,3}, \cdots, C_{j,N}\right);$
    $P = P';$
    $PUBT_{j+1,i} = C_{j,i}, \forall_i \in N;$
end
```

The distribution of jobs for production processes with unrelated parallel machines can be achieved as follows:

(1) At the initial moment, the current available time of each machine is 0, and the job number of all machines currently completed is set to be 0, and all jobs are in the first production stage;
(2) The production sequence P of the job is taken, and the job P_i to be processed is processed in order. According to the selected job allocation rules, the machine set k' that can be used in the current production stage j of the job is obtained so as to select the most suitable machine k to process the job;
(3) Determine the start processing time and completion time of job P_i on machine k;
(4) The completion time of job P_i on machine k is recorded as the current available time of machine k;
(5) Record the completion time of the current production stage of the job P_i;
(6) Recording the current completed job number on machine k;

(7) Judging whether P_i is the last job of the production sequence, if so, continue the following steps to complete the scheduling process, and if not, return to step (2), and then arrange the production of the next job;
(8) Arrange the completion time of the jobs in the current stage j in ascending order to obtain the production sequence of the jobs in the next stage, and the completion time of the job at this stage j is regarded as the release time of the virtual job at the beginning of the next process;
(9) Determine whether the current production stage j is the last production process; if so, complete the job allocation, otherwise return to (2), and continue to arrange the job production in the next stage.

According to the above scheduling method, any given production process and job allocation rules of each stage can obtain a reasonable scheduling scheme that satisfies production constraints.

The key to solving the scheduling optimization model proposed in this paper is to determine the optimal production sequence and set of job assignment rules, which is essentially a typical combinatorial optimization problem. In this paper, we use the improved line-up competition algorithm (ILCA) to solve the model.

3. Proposed Improved LCA

3.1. Original LCA

LCA is a stochastic optimization algorithm inspired by biology to simulate the process of biological evolution [23]. However, it is different from classical population intelligence optimization algorithms, such as genetic algorithms. Each family evolves independently and parallelly and generates its offspring through asexual reproduction. In addition, the competition forms include the survival competition of each generation in the vertical family and the status competition of each horizontal family. It is widely applied to complex optimization problems, such as mixed integer nonlinear programming problems and combinatorial optimization problems. The input to LCA is the solution for the multiple groups of individuals being evaluated. Once the fitness value of the individual is evaluated, the parent generation is selected to generate a new generation of offspring through the mutation strategy. The main feature of the proposed line-up competition algorithm is that the fine local search and the rough global search are carried out simultaneously so that the search speed is fast and an optimal or satisfactory solution can be obtained. In the following section, we describe the proposed ILCA in detail.

For the UPMST scheduling problem, the basic steps of LCA are as follows:

(1) Use random or other methods to generate n groups of mixed sequences consisting of all job production sequences and jobs using job assignment rules at each stage to form n initial populations;
(2) Calculate the scheduling objective function corresponding to each mixed sequence, and arrange each mixed sequence in ascending or descending order according to the fitness value;
(3) According to the sorting result, separate the production sequence and the job assignment rule set in the mixed sequence to implement the mutation strategy, and obtain the corresponding m sub-generation mixed sequences;
(4) Comparing the fitness values of the mixed sequences of the descendants of each family and the mixed sequences of the parents, in turn, and retaining the optimal individual as the parent to participate in the next round of competition for a platooning position;
(5) Judge whether the evolution reaches the given termination condition; if not, return to step (2); if it is reached, stop the evolution.

The flow chart of the calculation is shown in Figure 2.

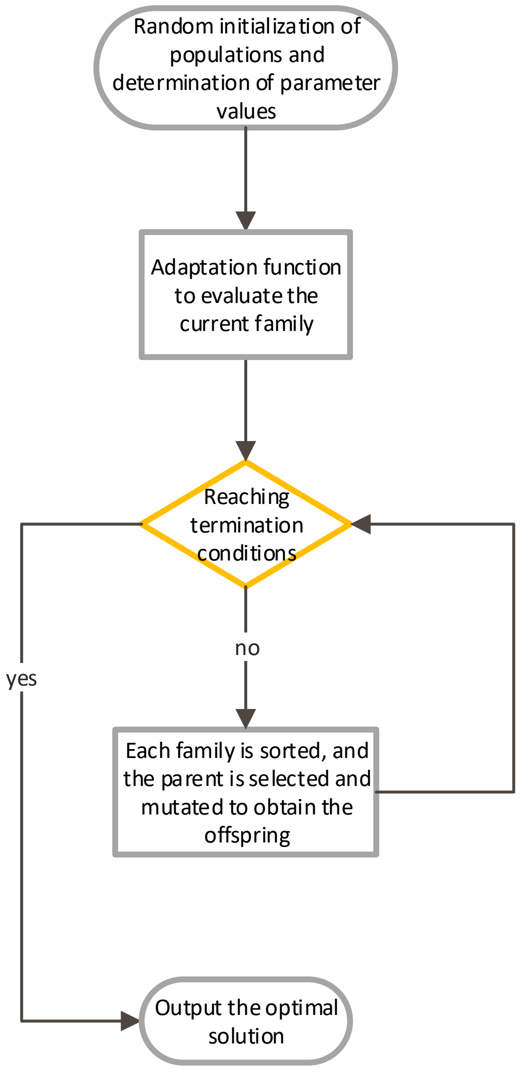

Random initialization of populations and determination of parameter values
Adaptation function to evaluate the current family
Reaching termination conditions
no
yes
Each family is sorted, and the parent is selected and mutated to obtain the offspring
Output the optimal solution

Figure 2. Flowchart of implementation process of LCA.

3.2. Improved LCA

In this paper, the random generation method is used to generate the initial population because random generation can expand the scope of the global search and avoid prematurely falling into the local optimal solution. In the heuristic artifact assignment rules proposed in the past, assignment rules are generally proposed in the form of minimization. This rule often has a small search space in practical solutions. Therefore, in order to reduce the occurrence of the above situation, new rules, such as selecting the machine with the maximum processing time, are added to the job allocation rules proposed in this paper. In addition, a suitable mutation strategy can generate excellent individuals to increase the search rate, thus speeding up the iterative convergence speed and obtaining the production scheduling plan faster. In previous research work, two jobs in an individual are randomly selected to exchange their production sequence positions to generate new solutions. However, the variation scale of this method is relatively small, which is not enough to jump out of the local optimum range. Furthermore, it may reduce the search range of the original suboptimal solution, which is not conducive to a global search. Therefore, this work proposes a mutation strategy based on these mutation strategies. See Figure 3 for details.

First, two position numbers of the production sequence are randomly selected to prepare for the population mutation step in the algorithm. We proposed five forms of variation, which are described as follows. Strategy 1: The two position numbers are interchanged. Strategy 2: The production sequence between the two position numbers and the sequences, including the two position numbers, is arranged in reverse order. Strategy 3: The sequence from the second position number sequence to the end is placed first, the sequence that includes the first position number before the first position number succeeds, and the sequence between the two positions is placed last. Strategy 4: Put the two position numbers taken out at the first position of the original production sequence, and place the remaining production sequences in turn. Strategy 5: Randomly shuffle the sequence rearrangement. The new job processing order is obtained by the above variation strategy, and the corresponding job assignment rules are updated randomly so that the new order of jobs is processed with the new allocation rules.

The initial solution obtained is sorted in ascending order, and the result of the obtained sort is subjected to the following variation operation. The top 20% of individuals in the top 20% of the family remained unchanged; strategy 1 was implemented for individuals ranked 21% to 40%; Strategy 2 was adopted for individuals ranked 41% to 60%; Strategy 3 was applied for individuals ranked 61% to 80%; and Strategy 4 was used for individuals ranked 81% to 100%. The top 20% of individuals in families ranked 21% to 40% adopted strategy 1; individuals ranked 21% to 40% implemented Strategy 2; individuals ranked 41% to 60% adopted Strategy 3; individuals ranked 61% to 80% applied Strategy 4; and individuals ranked 81% to 100% used Strategy 5. Strategy 1 was applied to the top 20% of individuals in families ranked 41% to 60%; Strategy 2 was implemented for individuals ranked 21% to 40%; Strategy 3 was applied for individuals ranked 41% to 60%; Strategy 4 was applied for individuals ranked 61% to 80%; and Strategy 5 was used for individuals ranked 81% to 100%. The top 20% of individuals in families ranked 61% to 80% adopted Strategy 1; individuals ranked 21% to 40% implemented Strategy 2; individuals ranked 41% to 60% adopted Strategy 3; individuals ranked 61% to 80% applied Strategy 4; and individuals ranked 81% to 100% used Strategy 5. Strategy 1 was adopted for the top 20% of individuals in the family ranked 81% to 100%, Strategy 2 was implemented for individuals ranked 21% to 40%, Strategy 3 was adopted for individuals ranked 81% to 100%, Strategy 4 was applied for individuals ranked 61% to 80%, and Strategy 5 was used for individuals ranked 81% to 100%.

New families are obtained by the above mutation strategy. Top-ranked families are searched in a smaller space by implementing smaller variants, thus increasing local search ability and speeding up convergence. The global search capability is increased by implementing larger variants for lower-ranked families to perform a larger spatial search. In this way, the local search and global search of the algorithm are synchronized.

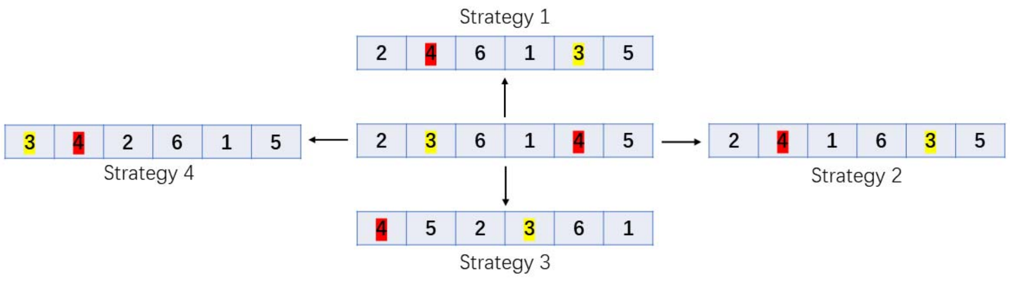


Figure 3. Schematic diagram of the mutation strategy.

4. Computational Experiments

In this section, to evaluate the effectiveness and efficiency of the proposed method, we demonstrate the following two test experiments. The proposed ILCA in the previous section is implemented in a high-level matrix language and run on a PC equipped with an Intel Core i5-6300HQ (2.3 GHz) CPU and 12 G RAM. The results obtained using the proposed ILCA are compared with the results reported in the literature.

4.1. Single-Stage UPMST Scheduling

The first experiment is a single-stage UPMST scheduling problem that considers the optimization objective of the traditional scheduling problem, i.e., the objective function is to minimize the maximum completion time. The performance of ILCA is also compared with several approaches previously reported in the literature for unrelated parallel machine problems and the same scheduling objective. For this benchmark test, there are other researchers who have done the same experiments, and they all proposed optimization algorithms to solve the UPMST scheduling problem. For example, the taboo search (TS)-based meta-heuristic algorithm proposed by Helal et al. [25], the randomized priority search meta-heuristic (Meta-RaPS) method proposed by Rabadi et al. [26], and the ant colony optimization (ACO) method proposed by Arnaout et al. [27]. Among them, the TS method applies a two-stage perturbation scheme, that is, an in-machine perturbation to optimize the work order on the machine and an out-of-machine perturbation to balance the work-to-machine assignment. The ACO method solves the UPMST problem by applying machine assignment and then sequencing the process.

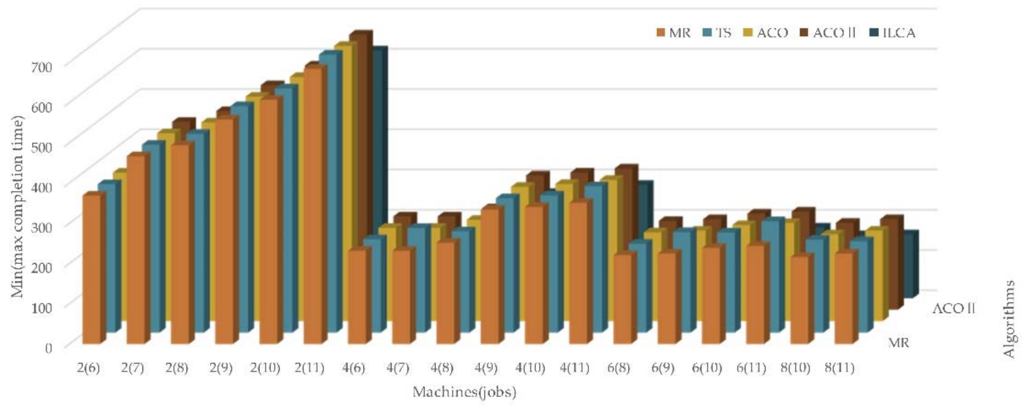
The test problems used in this study are obtained from the Scheduling Research website for the test problem dataset and the solution files obtained from previous studies [28]. Stated differently, the processing and setup times are randomly generated from two uniform distributions, U [50,100] and U [125,175]. When the processing and setup times are balanced (denoted by B), both the processing and setup times are obtained from U [50,100]. When the processing time dominates (denoted by P), they are separately generated from U [125,175] and U [50,100]. When the setup time is dominant (denoted by S), the processing time and setup time are obtained from [50,100] and [125,175], respectively. The resulting test problem dataset was obtained. This work focuses on problems with 6 to 11 artifacts and 2 to 8 machines. A total of 54 problem combinations are obtained with different number of artifacts, number of machines, and different dominance of processing time and preparation time. Each machine–job combination generated another 15 problem instances for testing. Thus, we make 810 $\times$ (54 $\times$ 15) instances for the problem set. The choice of parameters may affect the quality of the computational results. For our proposed ILCA, based on extensive computational testing, the following parameters are used for the experiments reported in this paper: an initial population of 200 and a number of iterations of 500. On top of this, 15 independent runs are also performed for each instance. All experiments terminate the algorithm computation after satisfying the termination condition, which is the number of iterations. ILCA is compared with four existing algorithms for solving the problem: TS by Helal et al. [25], Meta-Raps (MR) by Rabadi et al. [26], ACO by Arnaout et al. [27,29], and ACOII.

The simulation results are shown in Figure 4, where (a) indicates when setup time and processing time are balanced, (b) indicates when processing time is dominant, and (c) indicates the solution results of different algorithms when setup time is dominant. The values in the graph represent the minimum of the results of 15 independent simulations. The horizontal coordinates indicate the combination of different numbers of machines and workpieces. 2(6) indicates that there are two parallel machines, and the number of orders to be processed is 6 pieces. From the figure, it is clear that the optimal values obtained by ILCA are better than other algorithms.

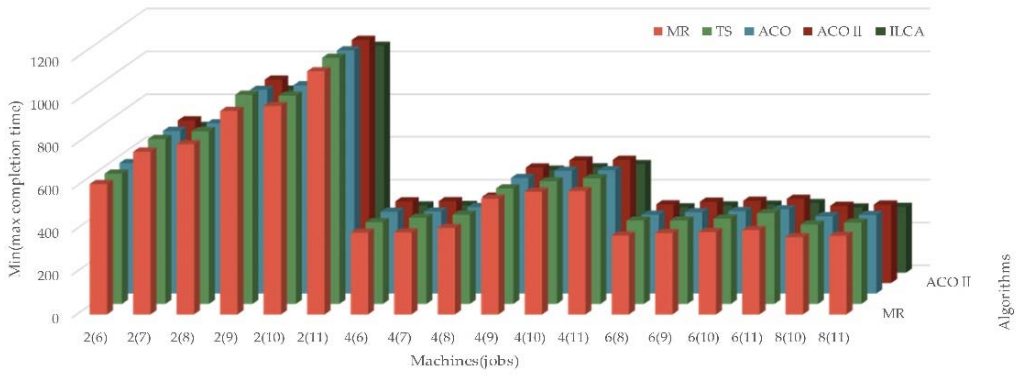
For the given problem instance, the performance of the proposed algorithm is evaluated in this work by means of the average deviation δ. The average deviation of the existing algorithm from the ILCA is calculated as follows:

$$\delta = \frac{A - B}{B} \tag{2}$$

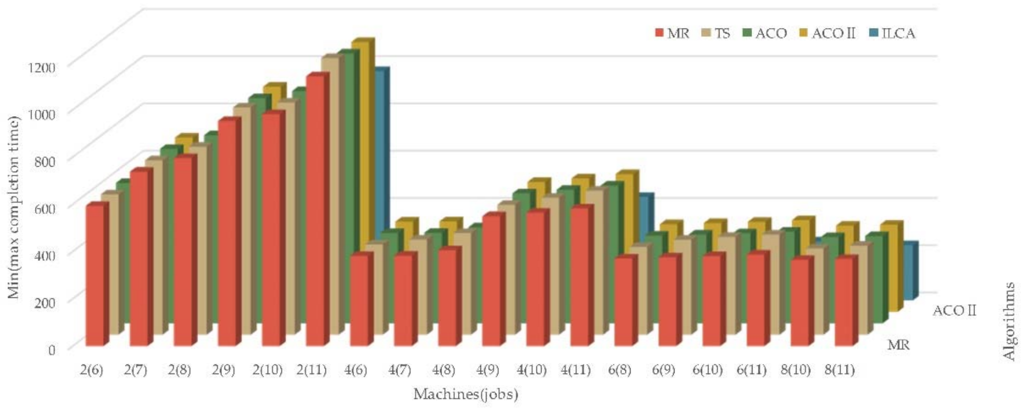
where A is the optimal objective value found for a problem instance using other optimization methods, and B is the optimal value solved by the ILCA proposed in this work. It is worth noting that, in essence, the δ metric is similar to the optimality gap usually reported in optimization software. Table 2 shows the δ values for the test instances of the problem.



(**a**)



(**b**)



(**c**)

Figure 4. The comparison of solutions. (**a**) represents the maximum completion time obtained by using different algorithms for different benchmark cases in the case of Balanced; (**b**) represents the case of Processing; similarly (**c**) represents the case of Setup.

Table 2. Average deviation of the existing algorithm from the ILCA.

M	**N**	$PT_{i,k}$,$SPT_{i,i'}$ **Balanced**				$PT_{i,k}$ **Dominant**				$SPT_{i,i'}$ **Dominant**			
		MR	**TS**	**ACO**	**ACOII**	**MR**	**TS**	**ACO**	**ACOII**	**MR**	**TS**	**ACO**	**ACOII**
2	6	0.212	0.217	0.212	0.212	0.145	0.149	0.145	0.145	0.306	0.310	0.306	0.306
	7	0.165	0.177	0.165	0.165	0.103	0.107	0.103	0.103	0.223	0.230	0.223	0.223
	8	0.122	0.131	0.121	0.121	0.106	0.109	0.106	0.106	0.240	0.246	0.240	0.240
	9	0.120	0.131	0.120	0.120	0.095	0.100	0.095	0.095	0.186	0.196	0.186	0.186
	10	0.108	0.124	0.108	0.108	0.075	0.081	0.075	0.075	0.192	0.199	0.192	0.192
	11	0.410	0.449	0.410	0.410	0.069	0.075	0.069	0.069	0.142	0.154	0.142	0.142
4	6	0.386	0.458	0.386	0.386	0.223	0.237	0.223	0.223	0.593	0.633	0.593	0.593
	7	0.386	0.458	0.386	0.386	0.222	0.257	0.222	0.222	0.561	0.604	0.561	0.561
	8	0.392	0.428	0.392	0.392	0.215	0.226	0.215	0.215	0.558	0.600	0.558	0.558
	9	0.292	0.290	0.289	0.289	0.141	0.139	0.141	0.141	0.360	0.361	0.360	0.360
	10	0.262	0.270	0.262	0.262	0.149	0.154	0.149	0.149	0.332	0.337	0.332	0.332
	11	0.234	0.261	0.234	0.234	0.136	0.164	0.138	0.136	0.324	0.346	0.324	0.324
6	8	0.420	0.467	0.420	0.420	0.224	0.240	0.224	0.224	0.606	0.634	0.606	0.606
	9	0.411	0.477	0.411	0.411	0.222	0.252	0.222	0.222	0.593	0.637	0.593	0.593
	10	0.383	0.459	0.386	0.383	0.213	0.249	0.213	0.213	0.585	0.650	0.585	0.585
	11	0.391	0.521	0.393	0.391	0.209	0.272	0.209	0.209	0.552	0.623	0.552	0.552
8	10	0.394	0.430	0.394	0.394	0.208	0.229	0.209	0.208	0.583	0.601	0.583	0.583
	11	0.416	0.435	0.416	0.416	0.215	0.242	0.215	0.215	0.587	0.618	0.587	0.587
Average		0.299	0.338	0.299	0.299	0.162	0.179	0.162	0.162	0.408	0.433	0.408	0.408

For the above test example problem, the average performance of ILCA is significantly better than other algorithms as can be seen in the results presented in the table. Note that a positive value of δ indicates that ILCA performs better, and a negative value indicates that ILCA performs worse than the listed algorithms. For the three different dominant cases, it can be seen that the performance of ILCA is better than the other two cases when $SPT_{i,i'}$ is dominant, that is, when the value of the conversion time is greater than the processing time, the minimum completion time obtained by ILCA is better. Specifically, if the rule is based on average performance, then for this problem, ILCA ranks first, followed by MR, ACO, and ACOII, and TS ranks last.

Additionally, to test for significant differences between algorithm performance, a paired *t*-test is performed for all problems to determine the best algorithm with a 95% confidence interval (see Table 3). The results indicate that the differences in the means of the mean differences of (TS-ACOII), (MR-ILCA) and (ACO-ILCA) are statistically significant (small *p*-value, $p < 0.05$). On the other hand, it is indicated that the differences in the means of the mean differences of (ACOII-MR) are not statistically significant ($p > 0.05$).

Table 3. Paired *t*-tests with 95% CI.

	Paired *t* Tests	**Mean Difference (MD)**	**SD**	***t* Stat**	**Two-Tailed *p***	**95% CI on MD**
$PT_{i,k}, SPT_{i,i'}$ balanced	**TS-ACOII**	7.448	9.543	12.825	1.060×10^{-29}	[6.305,8.592]
	ACOII-MR	−0.067	0.778	−1.408	0.160	[−0.160,0.027]
	MR-ILCA	70.463	5.410	214.015	1.81×10^{-302}	[69.815,71.111]
	ACO-ILCA	70.452	5.328	217.295	3.10×10^{-304}	[69.814,71.090]

Table 3. *Cont.*

	Paired t Tests	**Mean Difference (MD)**	**SD**	**t Stat**	**Two-Tailed p**	**95% CI on MD**
$PT_{i,k}$ dominant	**TS-ACOII**	6.759	9.710	11.438	5.960×10^{-25}	[5.596,7.923]
	ACOII-MR	−0.015	0.243	−1.000	0.318	[−0.044,0.014]
	MR-ILCA	72.848	6.933	172.662	1.44×10^{-277}	[72.017,73.679]
	ACO-ILCA	72.911	6.899	173.653	3.14×10^{-278}	[72.084,73.738]
$SPT_{i,i'}$ dominant	**TS-ACOII**	8.063	9.922	13.353	1.510×10^{-31}	[6.874,9.252]
	ACOII-MR	−0.015	0.243	−1.000	0.318	[−0.044,0.014]
	MR-ILCA	146.993	8.432	286.460	0.000	[145.982,148.003]
	ACO-ILCA	146.996	8.436	286.325	0.000	[145.986,148.007]

Tables 2 and 3 adequately demonstrate the average performance of the algorithm, but these values do not provide a detailed representation of the distribution of the computational results. To visualize the distribution of the results, this study summarizes the computational results in a box-line plot. Figure 5 box plots for all problem instances (including all instances and computed results). 1 to 4 on the horizontal coordinates indicate the mean deviations of MR, TS, ACO, ACO II, and ILCA when $PT_{i,k}$ and $SPT_{i,i'}$ are in equilibrium, respectively, and similarly 5 to 8 indicate the case when $PT_{i,k}$ is dominant, and similarly 9 to 12 would indicate the dominance of $SPT_{i,i'}$. Where δ values greater than 1.5 times the interquartile range are defined as outliers and are indicated by separate data points, no outliers can be seen in the graph. In the equilibrium case, the upper median of the mean deviation indicates the average level of this data sample, which is also indicated by the high value of δ. The trend direction of the variation seems to follow the same trend as the average performance. In other words, regardless of who is dominant in $SPT_{i,i'}$ and $PT_{i,k}$, the change in the mean deviation values of TS and ILCA is the largest, and the change in the difference between MR, ACO, ACOII, and ILCA is significantly slower in relative terms. This can be summarized by saying that the better the average performance, the lower the variability. This again verifies the validity and stability of the ILCA.

4.2. Multi-Stage UPMST Scheduling

The second experiment is about a multi-stage UPMST scheduling problem time. In this subsection, the green optimization objective of this scheduling problem is considered for the rational scheduling of the continuous casting and rolling production process in the aluminum industry [30], with the objective function of minimizing the energy loss and furnace waiting processing time due to the interruption of continuous casting and rolling. It is well known that the aluminum industry is the metallurgical process with the highest integrated energy consumption per unit. The continuous casting and rolling line in the aluminum production process means that the aluminum liquid coming from the refining furnace must be cast into aluminum ingots of corresponding shapes and specifications on the casting machine within a certain period of time according to the order properties, and then sent to the rolling mill for rolling after a very short heating time. In this production line, there is more than one aluminum refining furnace, continuous casting machine, and continuous rolling unit working in parallel, so it is an unrelated parallel machine scheduling problem. Between the melting and casting process, the aluminum liquid needs preparation time, which means waiting for processing time. Similarly, the aluminum ingots cast in the continuous casting machine have a preparation time from the continuous casting machine to the rolling mill, which is the UPMST scheduling problem studied in this paper. In addition, improperly sequenced production of workpieces in each process can lead to manufacturing interruptions, so we try to achieve zero waiting in the production process to reduce energy losses and thus achieve the goal of green scheduling as much as possible.

In this study, the scheduling solution obtained by the ILCA solution is compared with the results of previous studies, and it is proved that the solution obtained by ILCA is better than the solution in the compared literature.

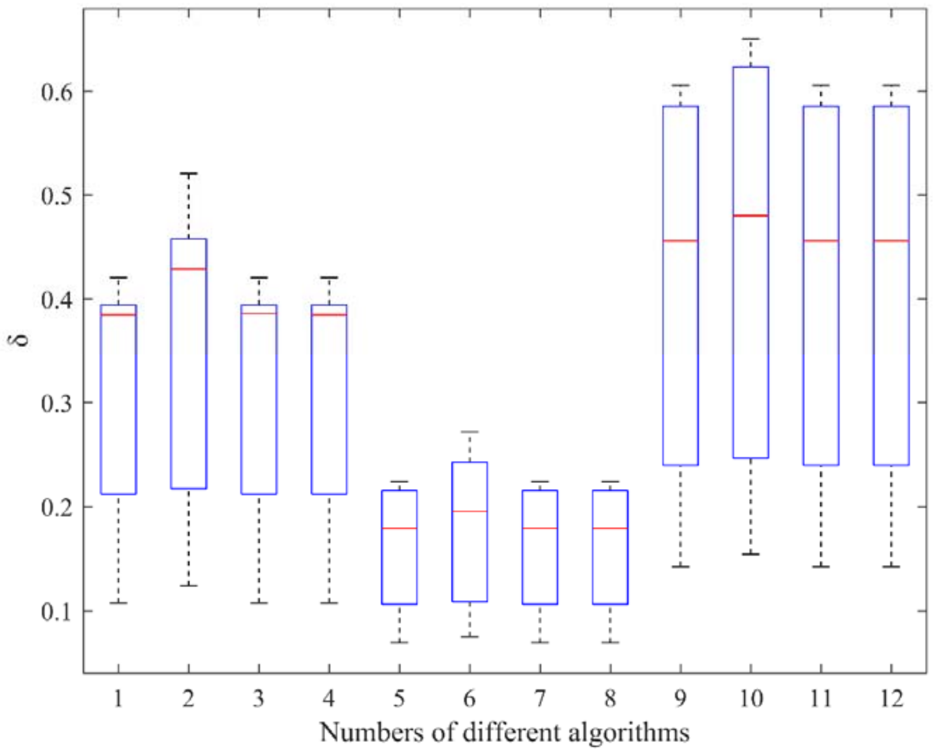


Figure 5. Box-and-whisker plot for average δ deviation from ILCA.

This example comes from [31] and the details include a pour with 12 furnaces (jobs) that undergo 3 processes of refining, continuous casting and rolling to obtain the final aluminum product. The entire line has 4 refining furnaces, 3 continuous casters, and 2 rolling mills, and each machine has a different processing capacity. The initial population number is taken to be 200, the maximum number of iterations is 200, the algorithm is run independently 15 times, and the termination condition is set to the maximum number of iterations. To verify the efficiency of ILCA method for solving continuous casting and rolling lines, several tests were conducted. A set of test results is shown in Figure 6, which shows the iterative convergence process of minimizing the energy loss due to continuous casting and rolling interruptions and the furnace waiting processing time for solving the continuous casting and rolling line scheduling problem for aluminum with ILCA. From the figure, it can be seen that ILCA has a strong ability to overcome premature convergence and avoid falling into the local optimum. When iterating from 50 generations to 150 generations, the optimal solution remains unchanged. A new optimal solution was obtained when iterating to 150 generations, which means that the population diversity of ILCA does not become worse as the number of iterations gradually goes deeper. And the search accuracy and search efficiency are significantly enhanced, and the global search ability is better. Figure 7 shows the final scheduling solution obtained by solving it with ILCA. We can also know from the diagram that every machine is used reasonably, and the connection between jobs is also very compact, which meets our actual production requirements. The rolling process is not interrupted, the continuous casting process is only interrupted by 1 time unit, and all jobs wait for 532 time units during the whole production process, which maximally meets the objective function of reducing interruptions in the continuous casting and rolling line. The experimental results show that the structure of the mathematical model of UPMST is realistic and reasonable, and that the program implemented according to the mathematical model can meet the technical requirements. ILCA quickly obtains the optimal scheduling solution, and the application of the scheduling model for continuous casting and rolling lines with unrelated parallel machines is practical and competitive.

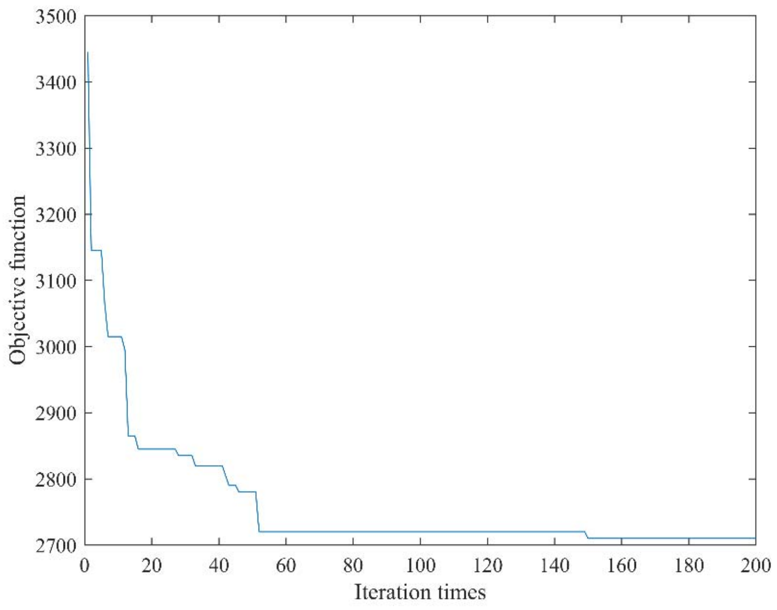


Figure 6. Best feasible solutions for multi-stage unrelated parallel machine scheduling problems from ILCA.

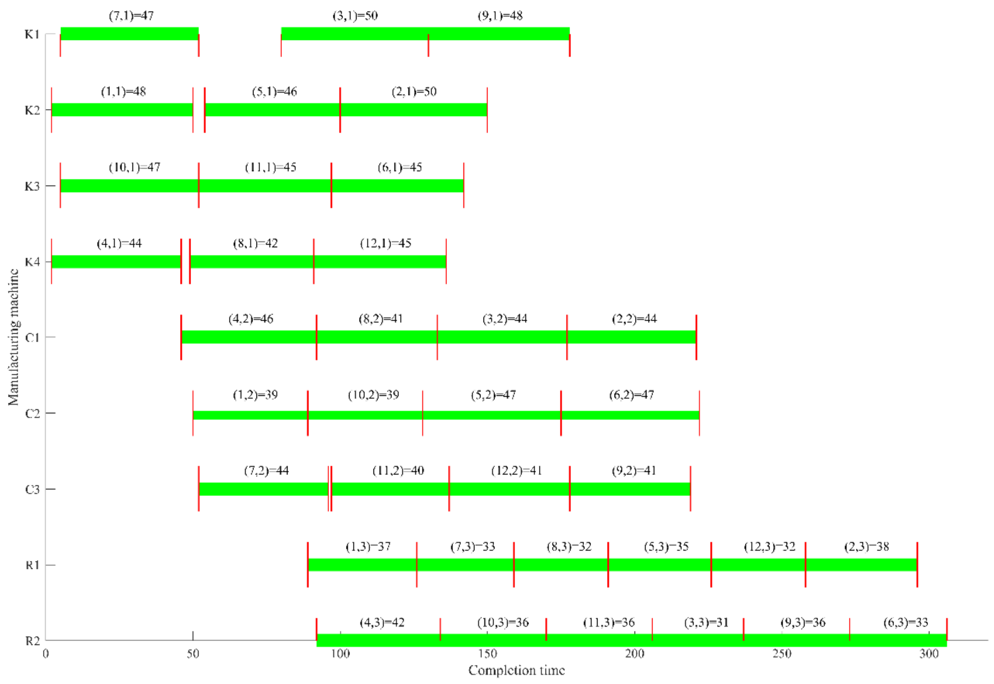


Figure 7. Optimal schedule from the proposed ILCA.

5. Conclusions

In this study, we propose a stochastic optimization algorithm also known as an improved queuing competition algorithm to solve the unrelated parallel machine scheduling problem with setup time. ILCA retains the advantages of LCA with few control parameters, simple programming, and stable operation, and proposes some new artifact allocation rules and population variation strategies to alleviate the phenomenal difficulty of getting into local extremes to some extent. The new assignment rules and different variation strategies according to different probabilities are applied to perform local and global simultaneous searches, which improves search accuracy and rate. In the first benchmark test experiment, it is shown that the proposed algorithm outperforms the final results of all the algorithms compared with it, and the average performance is better than the listed algorithms, verifying the effectiveness and stability of ILCA. The second example shows that ILCA can find a new optimal solution in the later stage of the iteration, thus jumping out of the local optimum. Moreover, the idle time of the machine is relatively short, which proves that the algorithm obtains a scheduling scheme that satisfies complex production constraints. Overall, ILCA with the new mutation strategy and the new heuristic artifact allocation rule has a better performance in solving the benchmark case of UPMST. In addition, when solving complex, actual production cases, a scheduling scheme that satisfies production constraints is also obtained. In future research, the large-scale UPMTS needs to be further explored to demonstrate the effectiveness and efficiency of ILCA in solving the large-scale UPMST. We will apply triangulation to verify the experimental data. Of course, it is necessary to consider the green optimization objective to help the enterprise reduce carbon emissions in production, so we will consider both the economic efficiency and carbon emissions for the aluminum continuous casting and rolling line. Therefore, we will optimize the solution for the aluminum continuous casting and rolling line, considering both economic efficiency and carbon emission, and develop a scheduling scheme that can provide some guidance for production planning.

Author Contributions: Conceptualization, B.S. and Y.X.; methodology, B.S.; software, B.S. and Y.X.; validation, Y.X.; formal analysis, Y.X.; investigation, Y.X.; resources, B.S. and Y.X.; writing-original draft preparation, Y.X.; writing-review and editing, B.S.; project administration, B.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China, founding number 21878238.

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Wuhan, and approved by the Institutional Review Board of National Natural Science Foundation of China (protocol code 21878238 and date of approval).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Acknowledgments: Thanks for the support provided by the National Natural Science Foundation of China.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lapczynska, D.; Lapczynski, K.; Burduk, A.; Machado, J. Solving the problem of scheduling the production process based on heuristic algorithms. *J. Univers. Comput. Sci.* **2022**, *28*, 292–310. [CrossRef]
2. Goli, A.; Tirkolaee, E.B.; Aydın, N.S. Fuzzy Integrated Cell Formation and Production Scheduling Considering Automated Guided Vehicles and Human Factors. *Trans. Fuzy Syst.* **2021**, *29*, 3686–3695. [CrossRef]
3. Ezugwu, A.E. Advanced discrete firefly algorithm with adaptive mutation-based neighborhood search for scheduling unrelated parallel machines with sequence-dependent setup times. *Int. J. Intell. Syst.* **2022**, *37*, 4612–4653. [CrossRef]
4. Cheng, C.Y.; Pourhejazy, P.; Ying, K.C.; Li, S.F.; Chang, C.W. Learning-Based Metaheuristic for Scheduling Unrelated Parallel Machines With Uncertain Setup Times. *IEEE Access* **2020**, *8*, 74065–74082. [CrossRef]
5. Rakovitis, N.; Li, D.; Zhang, N.; Li, J.; Zhang, L.; Xiao, X. Novel approach to energy-efficient flexible job-shop scheduling problems. *Energy* **2022**, *238*, 121773. [CrossRef]
6. Tadumadze, G.; Emde, S.; Diefenbach, H. Exact and heuristic algorithms for scheduling jobs with time windows on unrelated parallel machines. *OR Spectr.* **2020**, *42*, 461–497. [CrossRef]
7. Fanjul-Peyro, L. Models and an exact method for the unrelated parallel machine scheduling problem with setups and resources. *Expert Syst. Appl. X* **2020**, *5*, 100022. [CrossRef]

8. Yunusoglu, P.; Topaloglu Yildiz, S. Constraint programming approach for multi-resource-constrained unrelated parallel machine scheduling problem with sequence-dependent setup times. *Int. J. Prod. Res.* **2022**, *60*, 2212–2229. [CrossRef]
9. Vélez-Gallego, M.C.; Maya, J.; Montoya-Torres, J.R. A beam search heuristic for scheduling a single machine with release dates and sequence dependent setup times to minimize the makespan. *Comput. Oper. Res.* **2016**, *73*, 132–140. [CrossRef]
10. Laha, D.; Gupta, J.N.D. An improved cuckoo search algorithm for scheduling jobs on identical parallel machines. *Comput. Ind. Eng.* **2018**, *126*, 348–360. [CrossRef]
11. Arnaout, J.; Musa, R.; Rabadi, G. Ant colony optimization algorithm to parallel machine scheduling problem with setups. In Proceedings of the 2008 IEEE International Conference on Automation Science and Engineering, Arlington, VA, USA, 23–26 August 2008; pp. 578–582.
12. Arnaout, J.-P. A worm optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *Ann. Oper. Res.* **2020**, *285*, 273–293. [CrossRef]
13. Ying, K.-C.; Lee, Z.-J.; Lin, S.-W. Makespan minimization for scheduling unrelated parallel machines with setup times. *J. Intell. Manuf.* **2012**, *23*, 1795–1803. [CrossRef]
14. Avalos-Rosales, O.; Angel-Bello, F.; Alvarez, A. Efficient metaheuristic algorithm and re-formulations for the unrelated parallel machine scheduling problem with sequence and machine-dependent setup times. *Int. J. Adv. Manuf. Technol.* **2015**, *76*, 1705–1718. [CrossRef]
15. Santos, H.G.; Toffolo, T.A.M.; Silva, C.L.T.F.; Vanden Berghe, G. Analysis of stochastic local search methods for the unrelated parallel machine scheduling problem. *Int. Trans. Oper. Res.* **2019**, *26*, 707–724. [CrossRef]
16. Lin, D.-Y.; Huang, T.-Y. A Hybrid Metaheuristic for the Unrelated Parallel Machine Scheduling Problem. *Mathematics* **2021**, *9*, 768. [CrossRef]
17. Chen, C.; Fathi, M.; Khakifirooz, M.; Wu, K. Hybrid tabu search algorithm for unrelated parallel machine scheduling in semiconductor fabs with setup times, job release, and expired times. *Comput. Ind. Eng.* **2022**, *165*, 107915. [CrossRef]
18. Huang, X.; Chen, L.; Zhang, Y.; Su, S.; Lin, Y.; Cao, X. Improved firefly algorithm with courtship learning for unrelated parallel machine scheduling problem with sequence-dependent setup times. *J. Cloud Comput.* **2022**, *11*, 9. [CrossRef]
19. Fang, W.; Zhu, H.; Mei, Y. Hybrid meta-heuristics for the unrelated parallel machine scheduling problem with setup times. *Knowl. Based Syst.* **2022**, *241*, 108193. [CrossRef]
20. Pei, Z.; Wan, M.; Wang, Z. A new approximation algorithm for unrelated parallel machine scheduling with release dates. *Ann. Oper. Res.* **2020**, *285*, 397–425. [CrossRef]
21. Lei, D.; Yang, H. Scheduling unrelated parallel machines with preventive maintenance and setup time: Multi-sub-colony artificial bee colony. *Appl. Soft Comput.* **2022**, *125*, 109154. [CrossRef]
22. Moser, M.; Musliu, N.; Schaerf, A.; Winter, F. Exact and metaheuristic approaches for unrelated parallel machine scheduling. *J. Sched.* **2022**, *25*, 507–534. [CrossRef]
23. Yan, L.; Ma, D. A new algorithm for global optimization search line-up competition algorithm (I) solving nonlinear programming and mixed-integer nonlinear programming problem. *CIESC J.* **1999**, *50*, 663–670.
24. Shi, B.; Yan, L. Scheduling of multi-purpose batch process with parallel units. *CIESC J.* **2010**, *61*, 2875–2880.
25. Helal, M.; Rabadi, G.; Al-Salem, A. A tabu search algorithm to minimize the makespan for the unrelated parallel machines scheduling problem with setup times. *Int. J. Oper. Res.* **2006**, *3*, 182–192.
26. Rabadi, G.; Moraga, R.J.; Al-Salem, A. Heuristics for the unrelated parallel machine scheduling problem with setup times. *J. Intell. Manuf.* **2006**, *17*, 85–97. [CrossRef]
27. Arnaout, J.-P.; Musa, R.; Rabadi, G. A two-stage Ant Colony optimization algorithm to minimize the makespan on unrelated parallel machines—Part II: Enhancements and experimentations. *J. Intell. Manuf.* **2014**, *25*, 43–53. [CrossRef]
28. Scheduling Research. Available online: http://schedulingresearch.com/ (accessed on 10 May 2022).
29. Arnaout, J.-P.; Rabadi, G.; Musa, R. A two-stage ant colony optimization algorithm to minimize the makespan on unrelated parallel machines with sequence-dependent setup times. *J. Intell. Manuf.* **2010**, *21*, 693–701. [CrossRef]
30. Larbi, R.; Abrar, K.; Nadjib, B. Scheduling aluminum billet casting lines: A case study. *J. Ind. Intell. Inform* **2016**, *4*, 257–262. [CrossRef]
31. Xu, X.; Yin, E.; Zou, F. Researches on optimal scheduling for aluminum industry continuous casting and rolling production. In Proceedings of the 2010 IEEE International Conference on Intelligent Computing and Intelligent Systems, Xiamen, China, 29–31 October 2010; pp. 348–352.