Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2023.028339

ARTICLE

An Effective Neighborhood Solution Clipping Method for Large-Scale Job Shop Scheduling Problem

Sihan Wang, Xinyu Li and Qihao Liu*

School of Mechanical Science and Engineering, Huazhong University of Science and Technology, Wuhan, 430074, China

*Corresponding Author: Qihao Liu. Email: lllqh@hust.edu.cn

Received: 13 December 2022 Accepted: 29 January 2023 Published: 28 June 2023

ABSTRACT

The job shop scheduling problem (JSSP) is a classical combinatorial optimization problem that exists widely in diverse scenarios of manufacturing systems. It is a well-known NP-hard problem, when the number of jobs increases, the difficulty of solving the problem exponentially increases. Therefore, a major challenge is to increase the solving efficiency of current algorithms. Modifying the neighborhood structure of the solutions can effectively improve the local search ability and efficiency. In this paper, a genetic Tabu search algorithm with neighborhood clipping (GTS_NC) is proposed for solving JSSP. A neighborhood solution clipping method is developed and embedded into Tabu search to improve the efficiency of the local search by clipping the search actions of unimproved neighborhood solutions. Moreover, a feasible neighborhood solution determination method is put forward, which can accurately distinguish feasible neighborhood solutions from infeasible ones. Both of the methods are based on the domain knowledge of JSSP. The proposed algorithm is compared with several competitive algorithms on benchmark instances. The experimental results show that the proposed algorithm can achieve superior results compared to other competitive algorithms. According to the numerical results of the experiments, it is verified that the neighborhood solution clipping method can accurately identify the unimproved solutions and reduces the computational time by at least 28%.

KEYWORDS

Job shop scheduling; makespan; Tabu search; genetic algorithm

1 Introduction

Job shop scheduling problem (JSSP) is one of the most important combinatorial optimization problems in the field of operational research and management science [1]. As one of the most classic scheduling problems, it widely exists in the fields of aerospace, transportation, and automotive processing. JSSP can be briefly described as a set of jobs to be processed on a set of machines, and each job has to go through all machines in a certain order. It is a well-known NP-hard problem, when the number of jobs increases, the difficulty of solving the problem exponentially increases. In the past few decades, JSSP has been studied by a significant number of researchers. But even the most efficient state-of-the-art method cannot ensure the acquisition of the optimal solution to the JSSP problem and

becomes time-consuming as the size of the problem increases. Hence, there is still much work to be done on addressing large-scale JSSP.

Currently, many effective methods use local search methods with embedded neighborhood structures to enhance the local search capability of the methods, such as biased random-key genetic algorithm (BRKGA) [2], Tabu search/path relinking algorithm (TSPR) [3], Tabu search algorithm with a new neighborhood structure (TSED) [4], Knowledge-Based Multiobjective Memetic Algorithm (MOMA) [5], and variable neighborhood descent hybrid genetic algorithm (VND-hGA) [6]. Some of these algorithms are currently the most effective methods, such as BRKGA and TSPR, which improved the upper bounds of benchmark instances. From the existing literature, the results of the benchmark have been updated with the proposal of effective neighborhood structures, meaning that the neighborhood structure of JSSP plays a crucial role in the local search method. During the development of the neighborhood structure, it seems that the size of the neighborhood structure has a significant impact on the efficiency of local search. Large-size neighborhood structures may reduce the efficiency of the local search approach, while the properly-size neighborhood structure will improve the efficiency of the method. For instance, in comparison with N4 [7], N5 [8], and N6 [9], the exploitation and efficiency of neighborhood structure N1 [10] are inhibited because it spends a lot of time making unfruitful moves [7]. On the other hand, after erasing several unfruitful movements from N4, N6 is more constrained than N4 and is currently one of the most effective and efficient neighborhood structures [4]. Therefore, reducing the number of unfruitful moves to improve the quality of the neighborhood solution set has a positive impact on improving the efficiency of the algorithm.

To obtain a more efficient local search procedure, a neighborhood solution clipping method is proposed based on the domain knowledge of JSSP. It can reduce the computational cost of the algorithm by avoiding the search for unfruitful moves. Besides, a genetic Tabu search algorithm with neighborhood clipping (GTS_NC) is developed to solve large-scale JSSP. The main contributions of this paper are as follows:

(1) This work mined the domain knowledge of the neighborhood structure of JSSP that provided a deeper exploration of the characteristics of the JSSP.

(2) Based on the domain knowledge of JSSP, a neighborhood solution clipping method is developed to improve the efficiency of the local search method by avoiding the generation of unimproved neighborhood solutions.

(3) A feasible neighborhood solution determination method is proposed to distinguish feasible neighborhood solutions from infeasible ones.

The remainder of this paper is organized as follows. Section 2 reviews relevant literature. Section 3 is the problem formulation. Section 4 proposes the detail of GTS_NC. Section 5 shows the computational results and comparisons. Finally, Section 6 summarizes the research and discusses future research directions.

2 Literature Review

JSSP is a kind of typical machine scheduling problem that a huge amount of literature has been published within the last six decades [1]. In a general way, algorithms for JSSP can be grossly classified into two categories: exact algorithms and approximation algorithms. Since exact algorithms have encountered difficulties in solving JSSP with more than 250 operations in a reasonable time, approximation algorithms provide a quite good alternative for the JSSP [4]. More recently, swarm intelligence and evolutionary algorithms are widely used to solve JSSP, such as genetic algorithm

[11,12], Differential evolution algorithm [13], Tabu search algorithm [3,4,14], Coral Reef optimization [15], Memetic algorithm [16,17], Artificial bee colony algorithm [18], Artificial algae algorithm [19], Jaya algorithm [20], and Hybrid algorithms [21–23].

From the existing literature, it appears that the increase in the size of the problem leads to an increase in the computational consumption of the algorithm, thus reducing its efficiency of the algorithm. Hence, local search methods based on neighborhood structure, which can obtain high-quality local optimal solutions, have received increasing attention for their advantage in improving the exploration capability of the algorithm. For example, a new neighborhood structure with adaptive GA is developed in [22], in which crossover probability (P_c) and mutation probability (P_m) can be adjusted based on the dispersion of the fitness of the population in the evolution. In [24], an artificial bee colony algorithm adopted with five neighborhood structures is proposed for solving a profit-oriented and energy-efficient disassembly sequencing problem. Mohanmmad et al. [25] developed a hybrid algorithm that combines global equilibrium search, path relinking, and Tabu search to solve the JSSP. The neighborhood structure N6 is applied in the Tabu search framework. Cheng et al. [26] presented a Hybrid evolutionary algorithm which incorporated a Tabu search with neighborhood structure N7. The approach identifies a better upper bound of two instances, SWV06 and SWV08. Nagata et al. [27] presented a local search-based method that works in partial solution space for solving the JSSP. The local search procedure applied with neighborhood structure N6 is performed in a partial solution space where the current solution is represented as a partial schedule. Zobolas et al. [28] presented a hybrid metaheuristic method consisting of differential evolution, variable neighborhood search, and genetic algorithm.

In addition to the solving of JSSP, neighborhood structure-based local search methods are also widely used to solve other types of scheduling problems. For example, Li et al. [29] developed a new neighborhood search method for solving the permutation flow shop scheduling problem (PFSP). Zhang et al. proposed a variable neighborhood search using four neighborhood structures is proposed to solve the dynamic flexible job shop scheduling problems (DFJSP) [30]. Fan et al. [31] developed an improved genetic algorithm with a modified k-insertion neighborhood structure for solving flexible job shop scheduling problem (FJSP) considering reconfigurable tools with limited auxiliary modules. They also proposed a Tabu search embedded with three neighborhood operators to solve FJSP [32]. In fact, every approach that improves the best-known solutions of famous benchmark instances is developed with neighborhood structures, such as BRKGA [2], TSPR [3], and novel memetic algorithms [16]. The BRKGA, which combined with neighborhood structure N5, improved the best-known solution for 57 instances of well-known benchmarks. The TSPR algorithm, which is embedded with neighborhood structure N7, improved the best-known solution for 49 instances of well-known benchmarks. Besides, the neighborhood structure N7 is also used in the novel memetic algorithm [16], which improved the best-known solutions for 3 instances of well-known benchmarks. Thus, it is obvious that the neighborhood structures of JSSP play an essential role in the process of finding a high-quality optimal solution.

The development of neighborhood structures can be summarized as follow. In 1992, it is verified that the permutation of non-critical operations cannot improve the objective function and may create infeasible solutions. Subsequently, researchers developed neighborhood structure N1, which brought the attention of researchers to the study of the critical path of the scheduling solution. Combined simulated annealing with N1, the algorithm can find a better optimal solution than other effective approximation approaches at that time. As N1 includes a great number of unimproved moves, it is observed that unless the job-predecessor of operation u or the job successor of operation v is on the critical path, the interchange containing u and v cannot reduce the makespan [33]. This theory

strongly supports the development of neighborhood structures. For instance, neighborhood structure N4 is developed based on N1 by erasing the interchange of adjacent pair of critical operations and inserting an operation to either the front or the rear of the critical block. It combined with the Tabu search algorithm improves the best-known upper bound for 5 of the seven open benchmark problems. Similarly, the neighborhood structure N5 is designed based on N4, which only concerns the exchange of operations in the front and the back of the critical block. Computational experiment shows it is competitive in terms of computational consumption. To date, neighborhood structure N6 is the most effective neighborhood structure. It is properly optimized on the basis of N4, which maintains an abundant set of neighborhood solutions compared with N5. The algorithm embedded with N6 obtains the optimal solution of LA27 (unknown before) and improved most of the TD and LA instances. After the development of N6, researchers are still focusing on developing efficient neighborhood structures, such as Zhang et al. [4] proposed neighborhood structure N7, which extends N6 by moving either the first or the last operation of the critical block into the internal operation within the block. Zhao [34] proposed a multi-operation joint movement neighborhood structure, it can exchange up to 3 pairs of operations simultaneously. Xie et al. [35] developed a new neighborhood structure N8, which considers the movement of critical operations outside the critical block on the basis of N7.

The development of neighborhood structure reveals that the size of neighborhood structure is closely related to the quality and efficiency of local search. Large-scale neighborhood structure enables adequate exploration of the solution space, but the computational cost increases at the same time. Even the most efficient neighborhood structure N6 still contains a number of neighborhood solutions that are generated by invalid operations movements. Therefore, it is necessary to develop a JSSP domain knowledge-based method that can recognize and clip the unimproved neighborhood solutions to improve the efficiency of the algorithm.

In summary, this research proposed an algorithm named GTS_NC for solving large-scale JSSP. The details of the proposed method will be given in the following sections.

3 Problem Formulation

Job shop scheduling problem can be described as n jobs processed on m machines; each job has to go through all machines in a certain order. A job's processing on a machine indicates an operation of the job, and its duration is a given constant. It is an NP-hard problem. The constraints of JSSP are as follows: (i) the processing sequence of operations of each job is prescribed. (ii) each machine can process at most one job at a time. (iii) each operation can be processed on at most one machine at a time. The objective is to minimize the makespan.

It is useful to formulate JSSP with disjunctive graph $G = (N, A, E)$ including node set N, conjunctive arc set A (directed), and disjunctive arc set E (undirected) [4]. N is contained by all operations nodes, a dummy start node s, and a dummy end node e. The directed arc (i, j) of A shows the sequence constraints of operations where the length depends on the duration of operation i. The selected direction of the arc (i, j) of E decides the processing order of jobs on each machine. S denotes the set of selections of edge set E, which will instead of E get a solution $D_s = (N, A \cup S)$. A feasible solution D_s corresponds to an acyclic set S, that no directed cycle exists in the directed graph. Furthermore, if $L(u, v)$ denotes the longest path from u to v, then the $L(s, e)$ of D_s is equal to the makespan of the schedule. Thus, in the disjunctive graph of JSSP, the objective function is to find the acyclic set S that minimizes the longest path from s to e.

4 Proposed GTS_NC Method

4.1 Text Layout Neighborhood Solution Clipping Method

The neighborhood structure is a specific perturbation that can obtain a set of similar solutions. It usually considers a pair of operations u and v processed on the same machine, and both of them belong to the critical path of the solution (v processed after u). When u is moved right after v, we will call the perturbation a *forward interchange*. When v is moved before u, we will call the perturbation a *backward interchange*. As neighborhood structures focus on the design of movement rules, the lack of control over the quality of neighborhood solutions reduces the local search efficiency. Therefore, this paper proposes a neighborhood solution clipping method that can improve the efficiency of the local search method by avoiding the generation of unimproved neighborhood solutions.

To demonstrate our properties clearly, related symbols are identified as follows: For any operation u, denoting $\alpha(u)$ as the job-predecessor operation and $\gamma(u)$ as the job-successor operation, respectively [9]. Also, $\beta(u)$ means the machine-predecessor operation, and $\delta(u)$ means the machine-successor operation of operation u, respectively.

A neighborhood solution clipping method is well-designed in this section. We state these methods with proof as follows:

Proposition 1. For feasible scheduling, if the movement between a pair of critical operations of a solution cannot reduce the start time of at least one critical block, then the makespan of the solution will not reduce.

Proof. Suppose there are n critical blocks in solution S, and the start processing time of each critical block i is denoted as c_i with length l_i, thus the makespan of S is equal to $c_n + l_n$. Solution S' is obtained after the interchange of solution S. Since no critical block is advanced and the length l_i of each critical block will not change after a within-the-block interchange. Therefore, the longest path of S' from s to e is at least equal to $c_n + l_n$, which means the makespan of S' cannot be less than S.

Then we explore conditions, where an interchange on operation u and v is guaranteed, but cannot reduce the makespan.

Proposition 1.1. For a feasible scheduling S, v is the first operation in critical block, if $\alpha(u)$ is completed after the starting time of v, then the backward interchange of operation u before v cannot reduce the makespan.

Proof. Suppose solution S' is obtained after the backward interchange of S. The makespan of S can be described as $L(s, v) + p_v + L(v, e)$ or $L(s, u) + p_u + L(u, e)$. Then move u to the front of v, the start time of u is the completion time of $\alpha(u)$ due to the sequence constraints. Thus, the makespan of S' is $\max\{L(s, v)' + p_v + L(v, e)', L(s, u)' + p_u + L(u, e)'\}$. Where $L(s, u)' = L(s, \alpha(u)) + p_{\alpha(u)}$, $L(u, e)' = p_v + L(v, e)'$, $L(v, e)' = L(v, e) - p_u$, so $L(s, u)' + p_u + L(u, e)' = L(s, \alpha(u)) + \mathrm{p}_{\alpha(u)} + p_u + p_v + L(v, e) - p_u = L(s, \alpha(u)) + \mathrm{p}_{\alpha(u)} + p_v + L(v, e)$. Since $\alpha(u)$ is completed after the starting time of v in S, $L(s, v) \leq L(s, \alpha(u)) + p_{\alpha(u)}$. Hence $L(s, \alpha(u)) + p_{\alpha(u)} + p_v + L(v, e) \geq L(s, v) + p_v + L(v, e)$, which means a new longest path is created and the makespan of S' cannot be less than S confirms the Proposition 1.1 (see Fig. 1).

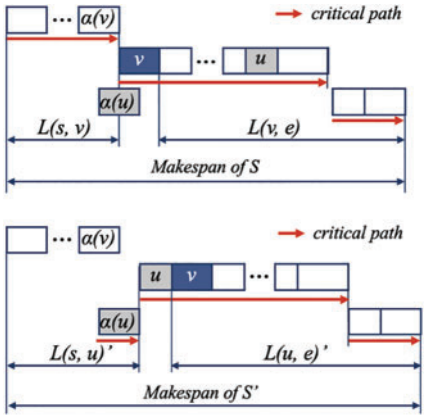

Figure 1: The backward interchange of operation u

Proposition 1.2. For a feasible scheduling S, v is the last operation in critical block, if $\gamma(u)$ processed on the same machine before $\gamma(v)$, then the forward interchange of operations u in critical block cannot reduce the makespan.

Proof. Suppose solution S' is obtained after the backward interchange of S. The makespan of S can be described as $L(s, v) + p_v + L(v, e)$. Then move u right after v that obtain solution S' which makespan is $\max\{L(s, v)' + p_v + L(v, e)', L(s, u)' + p_u + L(u, e)'\}$, where $L(s, v)' = L(s, v) - p_u$, $L(v, e)' = L(v, e)$, $L(s, u)' = L(s, v)' + p_v$, because $\gamma(u)$ processed on the same machine before $\gamma(v)$, so the processing of $\gamma(u)$ is delayed due to the delay of u, $L(u, e)' \geq p_{\gamma(u)} + L(v, e)$. Hence $L(s, u)' + p_u + L(u, e)' \geq L(s, v) - p_u + p_v + p_u + p_{\gamma(u)} + L(v, e) = L(s, v) + p_v + p_{\gamma(u)} + L(v, e) \geq L(s, v) + p_v + L(v, e)$. It means a new operation is added and the makespan of S' cannot be less than S confirms the Proposition 1.2 (see Fig. 2).

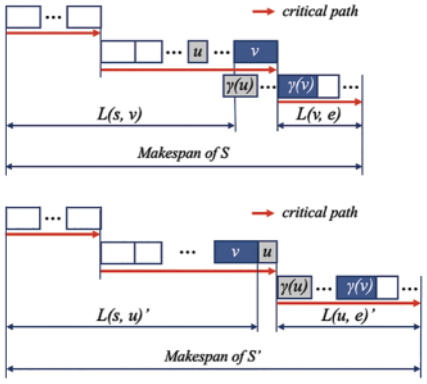

Figure 2: The forward interchange of operation u

Both proposition 1.1 and 1.2 can be used to decrease the size of N6 and N7. In addition, consider N7 has more perturbations than N6 with the forward interchange of the first operation and the backward interchange of the last operation in each critical block, another two propositions should be concerned as follows.

Proposition 1.3. For a feasible schedule S, u is the first operation in the critical block, if $\alpha(\delta(u))$ (if it exists) is completed after the starting time of u, then the forward interchange of operation u cannot reduce the makespan.

Proof. Suppose solution S' is obtained after the backward interchange of S. After the backward interchange of u, $\delta(u)$ will be the first operation of the critical block of S'. If $\alpha(\delta(u))$ (if it exists) is completed after the starting time of u, the start time of the critical block will be delayed due to the sequence constraints between $\alpha(\delta(u))$ and $\delta(u)$. Because the interchange will not change the length of the critical block, all the critical blocks behind will be delayed. According to Proposition 1, the makespan of S' cannot be less than S (see Fig. 3).

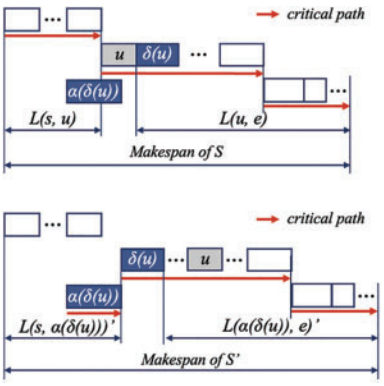

Figure 3: The forward interchange of operation u

Proposition 1.4. For an active scheduling S, u is the last operation in the critical block, if $\gamma(\beta(u))$ (if it exists) is processed on the same machine before $\gamma(u)$ (if it exists), then the backward interchange of operations u in the critical block cannot reduce the makespan.

Proof. Suppose solution S' is obtained after the backward interchange of S. After the forward interchange of u, $\beta(u)$ will be the last operation of the critical block of S'. If $\gamma(\beta(u))$ (if it exists) is processed on the same machine before $\gamma(u)$, all the operations between $\gamma(\beta(u))$ and $\gamma(u)$ will be added into the critical block containing $\gamma(u)$, the start time of $\gamma(u)$ will be delayed due to the sequence constraint of $\beta(u)$ and $\gamma(\beta(u))$. Considering no operation is removed from the critical path, the makespan of S' cannot be less than S (see Fig. 4).

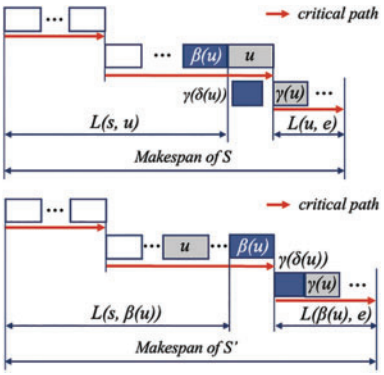

Figure 4: The backward interchange of operation u

4.2 Feasible Neighborhood Solution Determination Method

The determination of feasible neighborhood solutions involves the efficiency of local search methods because infeasible neighborhood solutions cannot provide a referable recommendation for the scheduling scheme. In order to distinguish a feasible neighborhood solution from an infeasible one, two propositions proposed by Balas et al. [9] are shown as follows:

Proposition 2. If two operations u and v to be performed on the same machine are both on the critical path $P(s, e)$, and $L(v, e) \geq L(\gamma(u), e)$, then a forward interchange on u and v yields an acyclic complete selection.

Proposition 3. If two operations u and v to be performed on the same machine are both on the critical path $P(s, e)$, and $L(s, u) + p_u \geq L(s, \alpha(v)) + p_{\alpha(v)}$, then a backward interchange on u and v yields an acyclic complete selection.

Although both of the propositions can ensure the feasible of the obtained neighborhood solutions, it was found to be inadequate for obtaining all acyclic solutions. As shown in Fig. 5a, the scheduling problem consists of 5 jobs and 4 machines ($J_{i,j}$ means the j-th operation of job i). The critical path of each solution is signed with red bold line. Solution S' is a neighborhood solution of S by backward interchange on $J_{2,2}$ and $J_{4,3}$. According to proposition 2, the backward interchange will lead to an infeasible solution because the $L(J_{4,3}, e) = p_{4,4}$, $L(J_{2,3}, e) = p_{2,4}$, $L(J_{4,3}, e) \leq L(J_{2,3}, e)$. In fact, however, the backward interchange corresponding to a feasible solution, and the Gantt chart of S' is shown in Fig. 5b. The reason is the length between the start node and end node to each operation cannot fully map the sequential connection relationship of the operations. Therefore, in this paper, a new theorem is developed to determine the feasible neighborhood solutions (see Fig. 6). For an operation u, we define a front operation set FS_u and back operation set BS_u, separately. The FS_u includes operation u's $\alpha(u)$, $\beta(u)$, $\mathrm{FS}_{\alpha(u)}$, and $\mathrm{FS}_{\beta(u)}$ (if exits). The BS_u includes operation u's $\gamma(u)$, $\delta(u)$, $\mathrm{BS}_{\gamma(u)}$ and $\mathrm{BS}_{\delta(u)}$ (if exits). Consider a feasible solution S:

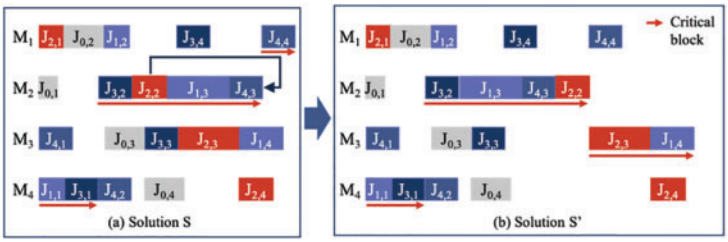

Figure 5: A counter-example for proposition 2. (a) Gantt chart of solution S. (b) Gantt chart of solution S'

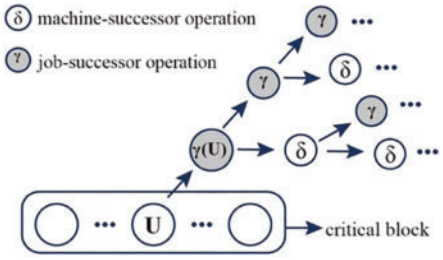

Figure 6: Example of the back operation set $\mathrm{BS}\gamma(u)$ of an individual

Proposition 4. If a critical path $P(s, e)$ containing u and v, and $v \notin BS_{\gamma(u)}$, then move u to the back of v yields an acyclic complete selection.

Proof. By contradiction: suppose there is a feasible solution S, if $v \notin BS_{\gamma(u)}$ for operation u and v on the critical path, then moving u to the back of v creates a cycle C in the disjunctive graph. Because the interchange of u and v deletes the direct arc from u to v on the machine, creating a direct arc (v, u) and a cycle, $(v, u) \in C$, there exists a path in D_s from u to v. Hence D_s exists a direct arc from u to another operations $l_1, \ldots, l_k$ connect with v. In this case, v must belong to $BS_{\gamma(u)}$ which is contrary to the assumption.

Proposition 5. If a critical path $P(s, e)$ containing u and v, and $u \notin FS_{\alpha(v)}$, then moving v to the front of u yields an acyclic complete selection.

Proof. By contradiction: suppose there is a feasible solution S, if $u \notin FS_{\alpha(v)}$ for operation u and v on the critical path, then moving v to the front of u creates a cycle C in the disjunctive graph. Because the interchange of u and v deletes the direct arc from u to v on the machine, creating a direct arc (v, u) and a cycle, $(v, u) \in C$, there exists a path in D_s from u to v. Hence D_s exists a direct arc from v to another operations $l_1, \ldots, l_k$ connect with u. In this case, u must belong to $FS_{\alpha(v)}$ which is contrary to the assumption.

The procedure of distinguishing feasible neighborhood solutions by existing method and the proposed method is shown in Fig. 7.

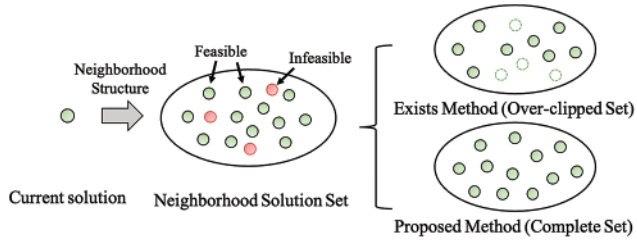

Figure 7: The procedure of filtering feasible neighborhood solutions

4.3 Framework of the Proposed GTS_NC Algorithm

In this paper, the hybrid algorithm [36] is combined with the proposed neighborhood solution clipping method for solving JSSP. The workflow and details of GTS_NC can be seen in Algorithm 1 which is composed of selection, crossover, mutation, and Tabu search. Each solution is encoded by a vector whose size is equal to the total number of operations and generated randomly. Besides, two-generation selection operators are adopted, an elitist selection scheme and a tournament selection scheme. In the elitist selection scheme, the best $p_r \times popsize$ solutions will be selected from solutions randomly chosen. In the tournament selection scheme, a number of individuals ($p_t \times popsize$) will be maintained through randomly chosen t_n solutions from the population and keep the one with the best fitness. Then, the crossover scheme contains precedence operation crossover (POX) and job-based crossover (JBX) [36]. The mutation scheme contains swapping mutation and neighborhood mutation methods [36]. The Tabu search algorithm [4] is adopted as local search algorithm. To enhance the diversity of the Tabu search and avoid premature convergence of the algorithm, we improve the Tabu search algorithm with a memory pool to save the best-unchosen neighborhood solution in each iteration. In addition, a random solution will be generated if the size of the neighborhood solution set is less than two. The workflow of the Tabu search algorithm is shown in Fig. 8.

Algorithm 1: Pseudocode of GTS_NC.

Initialization: Generate the initial population randomly
Set *gen*=1, *gen* is the current generation
Evaluation: Evaluate every individual in the population by the objective
While $gen<maxGen$ **do**
    // Select $p_r \times popsize$ individuals in the population by elitist selection scheme
    // Select $p_t \times popsize$ individuals in the population by tournament selection scheme.
    // Generate $(1 - p_r - p_t) \times popsize$ individuals by crossover operator POX or JBX with equal probability
    for $i = 1$ to *popsize* **do**
     //Generate the individual ind_i by swapping mutation or neighborhood mutation with probability p_m
    //Apply the improved Tabu search algorithm to improve the quality of ind_i
        if ($fitness(ind_i) < fitness(bestind)$) **then**
            Set $bestind \leftarrow ind_i$ and $fitness(bestind) \leftarrow fitness(ind_i)$
        endif
    endfor
    $gen \leftarrow gen + 1$
endwhile
return *bestind*

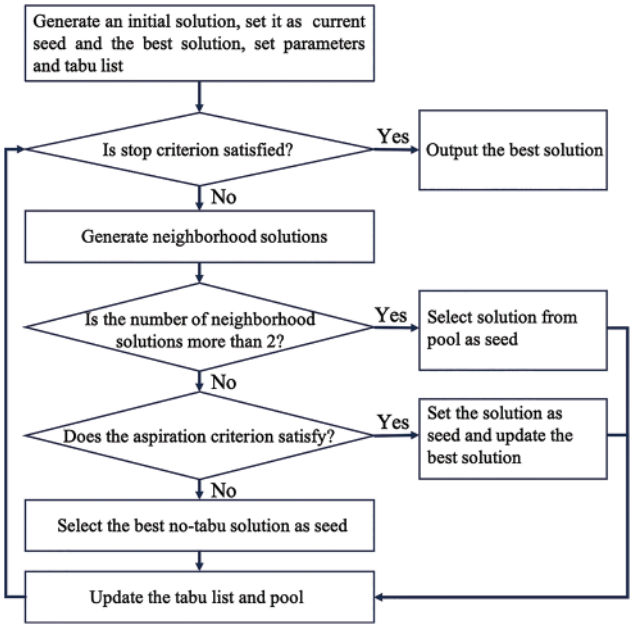

Figure 8: Tabu search workflow

5 Results and Discussion

In this section, three experiments are set to verify the proposed feasible neighborhood solution determination method, the neighborhood solution clipping method, and the proposed GTS_NC algorithm, respectively.

Experiment 1 compares the proposed feasible neighborhood solution determination method with the state-of-the-art feasible solution definition method [4] according to the average quantity of feasible neighborhood solutions obtained for each benchmark instance problem. Experiment 2 tests the effectiveness of the proposed neighborhood solution clipping method by the local search algorithm. Experiment 3 tests the performance of the proposed GTS_NC algorithm by comparing it with other algorithms. All the best solutions are recorded. The famous LA benchmark instance [37] was adopted to evaluate the performance of the proposed algorithm. The objective of this paper is to minimize makespan.

The proposed GTS_NC algorithm was coded in C++ and implemented on a computer with an Apple M1 with 16.0 GB of RAM memory.

5.1 Parameters Setting

The parameters of GTS_NC are the same as their basic papers, respectively (see Table 1) [4,36].

Table 1: Parameter settings of GTS_NC

Parameter	Value
RunTime	20
Max generation	200
Population	200
P_c (crossover rate)	0.9
P_m (mutation rate)	0.1
P_r (elitist individual rate)	0.005
Tournament size	2

The maximum iteration size *maxTS* was 10000 and a dynamic tabu list was used [4], which length dynamically changed from *Lmin* to *Lmax* ($L_{min} = l$, $L_{max} = [1.4 - 1.5l]$, where if $n \leq 2m$, then the maximal extreme value $L_{max} = [1.4l]$. Otherwise, $L_{max} = [1.5l]$). The parameters of GTS_NC were shown in Table 1 ($n \times m$ means the problem contains n jobs and m machines).

5.2 Experiment 1

To validate the performance of the suggested feasible neighborhood solution determination method (denoted as FDM), it is compared with the basic version of the proposition proposed by Zhang et al. [4]. The comparison is performed in terms of the average quantity of feasible neighborhood solutions (N_{avg}). The experimental can be described as follows.

Firstly, for each problem, 2000 feasible initial solutions were randomly generated, and their neighborhood solutions were generated by neighborhood structure N7. Then, for each initial solution, record the number of its feasible neighborhood solutions selected by feasible solution definition methods. Finally, the average quantity of feasible neighborhood solutions for each problem was compared to show the determination ability of each method. In particular, each selected neighborhood solution was decoded to verify its feasibility.

The reported results are demonstrated in Table 2. The outcomes demonstrated that the proposed method is more competitive than the proposition proposed by Zhang et al. [4] for all twelve benchmark

test problems. It means that the proposed feasible neighborhood solution determination method has obvious advantages in mapping the JSSP.

Table 2: Average quantity of feasible neighborhood solutions in different methods

Instance	$n \times m$	Zhang et al. [4]	FDM	Improvement (%)
		N_{avg}	N_{avg}	
LA02	10 × 5	12.53	16.02	**27.85**
LA19	10 × 10	10.28	12.24	**19.07**
LA21	15 × 10	17.68	24.21	**36.93**
LA24	15 × 10	14.47	20.11	**38.98**
LA25	15 × 10	17.63	27.42	**55.53**
LA27	20 × 10	23.72	35.54	**49.83**
LA29	20 × 10	20.78	32.88	**58.23**
LA36	15 × 15	14.01	23.84	**70.16**
LA37	15 × 15	17.44	26.94	**54.47**
LA38	15 × 15	16.7	24.27	**45.33**
LA39	15 × 15	15.53	22.64	**45.78**
LA40	15 × 15	17.92	24.46	**36.50**

Note: N_{avg} means the average quantity of neighborhood solutions selected by each method.

5.3 Experiment 2

To validate the performance of the proposed neighborhood solution clipping method, the Tabu search embedded and unembedded neighborhood solution clipping method is compared. The widely used neighborhood structure N6 was adopted here. LA instances which contain at least 150 operations are considered in this paper. The maximum generation of Tabu search is 10000. Each of the problems is tested 10 times independently.

The comparison results are demonstrated in Table 3. Table 3 represents, name of problem instances, size of the problem that represents number of jobs × number of machines, the makespan of the best known solutions (C_{max}) obtained by TSN6 (TS unembedded with neighborhood solution clipping method), the average makespan of the best known solutions (C_{avg}) obtained by TSN6, the shortest (T_{min}) and average computational cost (T_{avg}) of TSN6, and the values of the best known solutions obtained by TSN6_NC (TS embedded with neighborhood solution clipping method).

Table 3: Comparison of neighborhood clipping method and Tabu search with neighborhood structure N6

Problem	$n \times m$	TSN6				TSN6_NC			
		C_{max}	C_{avg}	T_{min}	T_{avg}	C_{max}	C_{avg}	T_{min}	T_{avg}
LA21	15 × 10	1058	1071.5	12.873	13.587	1055∗	1080.6	7.938∗	8.436
LA22	15 × 10	939	950.3	12.031	12.326	939	1921.9	7.504∗	7.791
LA23	15 × 10	**1032**	1033.55	0.217	3.312	**1032**	1032	0.074∗	0.925
LA24	15 × 10	949	966.75	9.759	10.231	949	1941.2	5.153∗	7.274
LA25	15 × 10	992	1020.1	10.938	12.620	991∗	1007.9	7.981∗	8.239
LA26	20 × 10	**1218**	1225.6	1.738	16.268	**1218**	1220.55	0.676∗	7.451
LA27	20 × 10	1265	1293.5	23.587	26.636	1254	1294.05	11.631∗	14.209
LA28	20 × 10	1216	1233.25	10.026	24.869	1216	2484.76	3.987∗	12.871
LA29	20 × 10	1208	1278.05	27.953	35.337	1207	1257.55	12.415∗	15.199
LA30	20 × 10	**1355**	1440.65	2.272	23.709	**1355**	1429.2	0.818∗	14.071
LA31	30 × 10	**1784**	1795.45	0.988	76.367	**1784**	1791.9	0.454∗	30.468
LA32	30 × 10	**1850**	1876.1	1.350	192.550	**1850**	1866.55	0.763∗	39.788
LA33	30 × 10	**1719**	1730.5	0.821	74.160	**1719**	1725.5	0.330∗	16.326
LA34	30 × 10	**1721**	1785	3.506	264.340	**1721**	1782.55	0.738∗	94.541
LA35	30 × 10	**1888**	1911.8	2.164	184.198	**1888**	1894.75	0.614∗	33.777
LA36	15 × 15	1312	1322.5	29.655	33.615	1291∗	1326.45	18.735∗	22.052
LA37	15 × 15	1450	1466.35	28.503	32.613	1450	1465.45	16.312∗	17.458
LA38	15 × 15	1236	1260.4	29.413	30.783	1220∗	1265.43	14.828∗	16.910
LA39	15 × 15	1251	1278.55	27.310	30.187	1251	1282.45	15.592∗	16.746
LA40	15 × 15	1239	1248.95	27.681	31.167	1233∗	1243.21	14.542∗	17.184
Number of better solutions		0		0		**5**		20	

Note: The values marked with ∗ are the better results.

In Table 3, the bolded results are the optimal C_{max}, others marked by ∗ are the better C_{max} and better T_{min}. As shown in Table 3, the TSN6_NC obtained seven better optimal solutions than TSN6 (21, 25, 27, 29, 36, 38 and 40). Moreover, TSN6_NC gets 34 problems with lower computational cost. For instance, TSN6 require 27.9 s on solving LA29, but TSN6_NC requires 12.4 s to get better solution. The convergence curves of TSN6 and TSN6_NC are shown in Fig. 9. From the reported results in Table 3, it is obvious that the suggested algorithm TSN6_NC produces more efficient results in terms of best makespan and computational time. In Particular, the advantage of the proposed algorithm in terms of computational time become more pronounces as the size of the problem increases, as shown in Table 4. The obtained results show that proposed neighborhood solution clipping method produces more efficiency results as compared to the Tabu search unembedded with it.

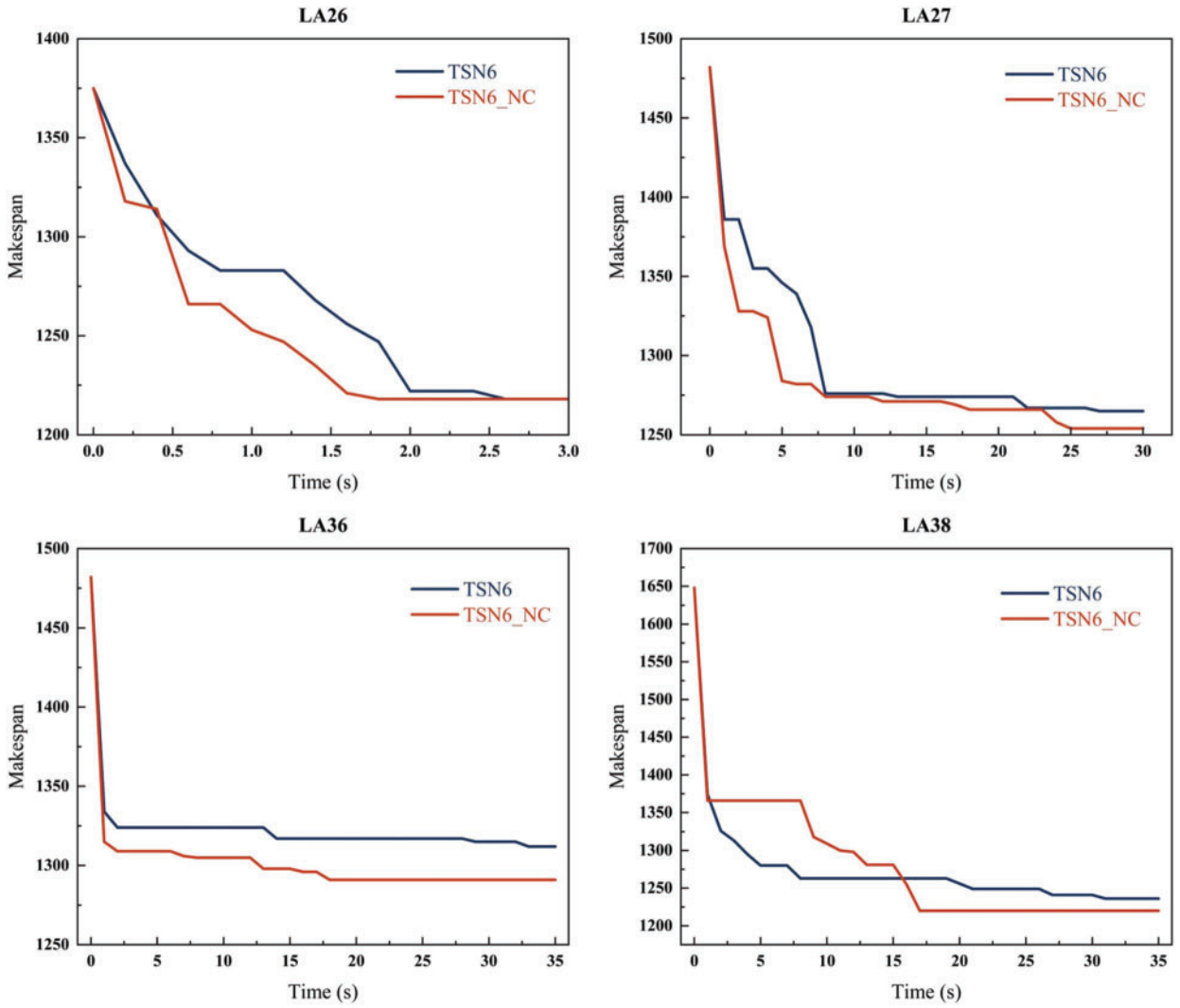

Figure 9: Convergence curves of the TSN6 and TSN6_NC for LA problems

Table 4: Comparison based upon average computational cost

Algorithm	15 × 10	15 × 15	20 × 10	30 × 10
TSN6	9.16	28.51	13.11	1.76
TSN6_NC	6.53	16	5.9	0.58
Improvement (%)	**28.71**	**43.88**	**55.00**	**67.05**

5.4 Experiment 3

To validate the efficiency of the proposed algorithm GTS_NC (applied with neighborhood structure N6), LA instances that contain at least 150 operations are considered in this paper. The

obtained results are compared with the following significant approaches for solving JSSP as available in the literature:

1. teaching-learning-based optimization (TLBO) algorithm [38],
2. upper-level algorithm (UPLA) [39],
3. genetic algorithm with improved local search techniques (mXLSGA) [11],
4. mXLSGA with Frequency Analysis (GIFA) Operator (GIFA-mXLSGA) [12],
5. the proposed approach unembedded with the neighborhood solution clipping method (GTS).

The comparison results with all the considered algorithms are depicted in Table 5. Table 5 represents the name of problem instances, the size of the problem, the upper bound of the problem, the makespan value (C_{max}) of the best solution obtained by each algorithm, the CPU time of GTS and GTS_NC, and relative percentage error (RPE) with respect to C_{max}. Here, PRE is calculated using the following Eq. (1):

$$RPE = 100 \times (C_{max} - \text{UB})/\text{UB} \tag{1}$$

The reported results for LA instances for TLBO, UPLA, mXLSGA, GIFA-mXLSGA, GTS and GTS_NC are shown in Table 5. From the reported results in Table 5, it is obvious that the suggested algorithm GTS_NC produces more efficient results in terms of best makespan and RPE in comparison with TLBO, UPLA, mXLSGA, GIFA-mXLSGA, and GTS. Besides, GTS_NC has an obvious advantage in computational time.

Table 5: Comparison between GTS_NC and other approaches on LA instances

Problem	UB	TLBO (2013)		UPLA (2019)		mXLSGA (2020)		GIFA-mXLSGA (2022)		GTS		GTS_NC		CPU time	
		C_{max}	RPE	C_{max}	RPE	C_{max}	RPE	C_{max}	RPE	C_{max}	RPE	C_{max}	RPE	GTS	GTS_NC
LA21	1046	**1046**	0.00	1052	0.57	1059	1.24	1052	0.57	**1046**	0.00	**1046**	0.00	2093	1034∗
LA22	927	**927**	0.00	**927**	0.00	935	0.86	**927**	0.00	**927**	0.00	**927**	0.00	5399	1155∗
LA23	1032	**1032**	0.00	**1032**	0.00	**1032**	0.00	**1032**	0.00	**1032**	0.00	**1032**	0.00	1.94	0.60∗
LA24	935	**935**	0.00	941	0.64	946	1.18	940	0.53	**935**	0.00	**935**	0.00	29062	6107∗
LA25	977	**977**	0.00	982	0.51	986	0.92	984	0.72	978	0.10	**977**	0.00	35167	20241∗
LA26	1218	**1218**	0.00	**1218**	0.00	**1218**	0.00	**1218**	0.00	**1218**	0.00	**1218**	0.00	114	63∗
LA27	1235	**1235**	0.00	1256	1.70	1269	2.75	1261	2.11	1247	0.97	**1235**	0.00	72097	40051∗
LA28	1216	**1216**	0.00	**1216**	0.00	1239	1.89	1239	1.89	**1216**	0.00	**1216**	0.00	1311	1853∗
LA29	1152	1171	1.65	1191	3.39	1201	4.25	1190	3.30	1179	2.34	1161	0.78	93400	47970∗
LA30	1355	**1355**	0.00	**1355**	0.00	**1355**	0.00	**1355**	0.00	**1355**	0.00	**1355**	0.00	82	3.82∗
LA31	1784	**1784**	0.00	**1784**	0.00	**1784**	0.00	**1784**	0.00	**1784**	0.00	**1784**	0.00	6.98	3.90∗
LA32	1850	**1850**	0.00	**1850**	0.00	**1850**	0.00	**1850**	0.00	**1850**	0.00	**1850**	0.00	5.46	2.41∗
LA33	1719	**1719**	0.00	**1719**	0.00	**1719**	0.00	**1719**	0.00	**1719**	0.00	**1719**	0.00	14.31	2.88∗
LA34	1721	**1721**	0.00	**1721**	0.00	**1721**	0.00	**1721**	0.00	**1721**	0.00	**1721**	0.00	253.92	42.55∗
LA35	1888	**1888**	0.00	**1888**	0.00	**1888**	0.00	**1888**	0.00	**1888**	0.00	**1888**	0.00	123	6.47∗
LA36	1268	**1268**	0.00	1278	0.79	1295	2.13	1295	2.13	1281	1.03	**1268**	0.00	95218	57719∗
LA37	1397	1407	0.72	1407	0.72	1415	1.29	1407	0.72	1415	1.29	1401	0.29	96434	51830∗
LA38	1196	1215	1.59	1215	1.59	1246	4.18	1246	4.18	1203	0.59	1202	0.50	74217	42402∗

(Continued)

Table 5 (continued)

Problem	UB	TLBO (2013)		UPLA (2019)		mXLSGA (2020)		GIFA-mXLSGA (2022)		GTS		GTS_NC		CPU time	
		C_{max}	*RPE*	C_{max}	*RPE*	C_{max}	*RPE*	C_{max}	*RPE*	C_{max}	*RPE*	C_{max}	*RPE*	GTS	GTS_NC
LA39	1233	1248	1.22	1250	1.38	1258	2.03	1258	2.03	1241	0.65	**1233**	0.00	74716	45003∗
LA40	1222	1229	0.57	1229	0.57	1243	1.72	1243	1.72	1229	0.57	1227	0.41	89090	46045∗
Mean(RPE)			0.29		0.59		1.22		0.99		0.38		0.10		
NOS		1		10		8		9		12		16		0	20

Note: The values marked with ∗ are the better results.

Further, to analyze the convergence characteristics of the proposed GTS_NC algorithm, it is applied to the LA25 instance and the average makespan is recorded during the generations and computational time. The obtained results are depicted in Fig. 10. The proposed GTS_NC algorithm converges very fast and finds the optimal solution very rapidly as shown in Fig. 10. The Gantt chart of the best solution of LA25 obtained by the proposed approach is shown in Fig. 11.

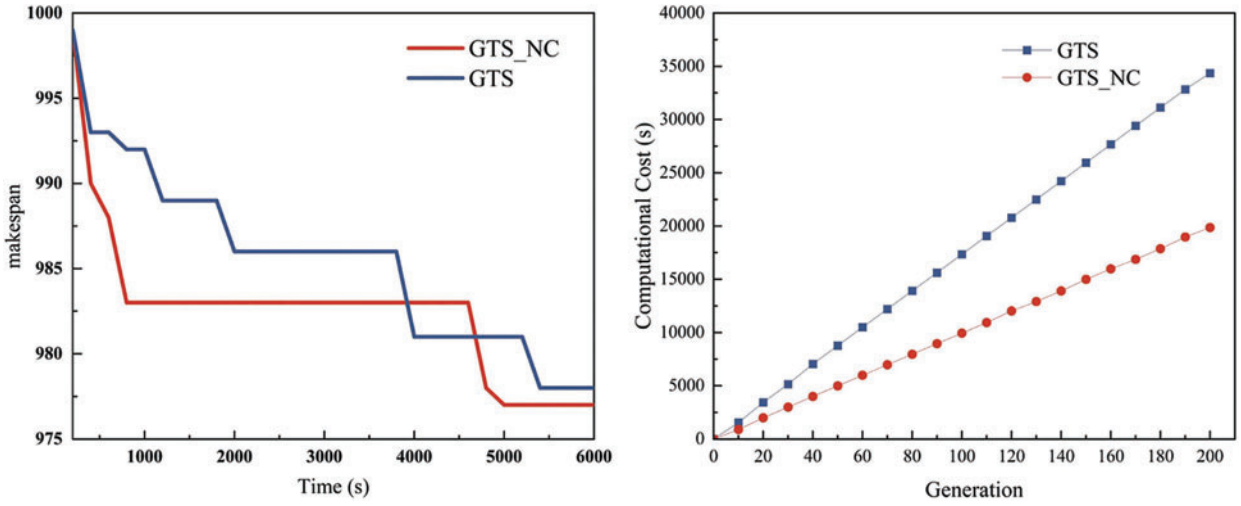

Figure 10: Convergence characteristics of LA25

5.5 Discussion

From the computational results, the proposed GTS_NC can obtain the best results for most problems. Embedded with the neighborhood solution clipping method, GTS_NC shows strong competitiveness in terms of computational time. And the proposed feasible neighborhood solution determination method was verified so that it can accurately distinguish feasible neighborhood solutions from infeasible ones, and achieves a better mapping relationship with the JSSP.

The reasons are as follows. Firstly, the neighborhood solution clipping method was proposed based on the characteristic of movement of operations in the critical block, so it can accurately avoid the calculation for unimproved neighborhood solutions to save computational time. Secondly, the proposed feasible neighborhood solution determination method was designed based on the precedence constraints between operations, which provides a more comprehensive view of all feasible solutions by

directly checking for constraint satisfaction. However, there is also a disadvantage of the proposed neighborhood solution clipping method, the decrease in the neighborhood solution set will limit the diversification of the local search procedure. Therefore, this method will perform better when combined with strategies that increase the diversity of the population.

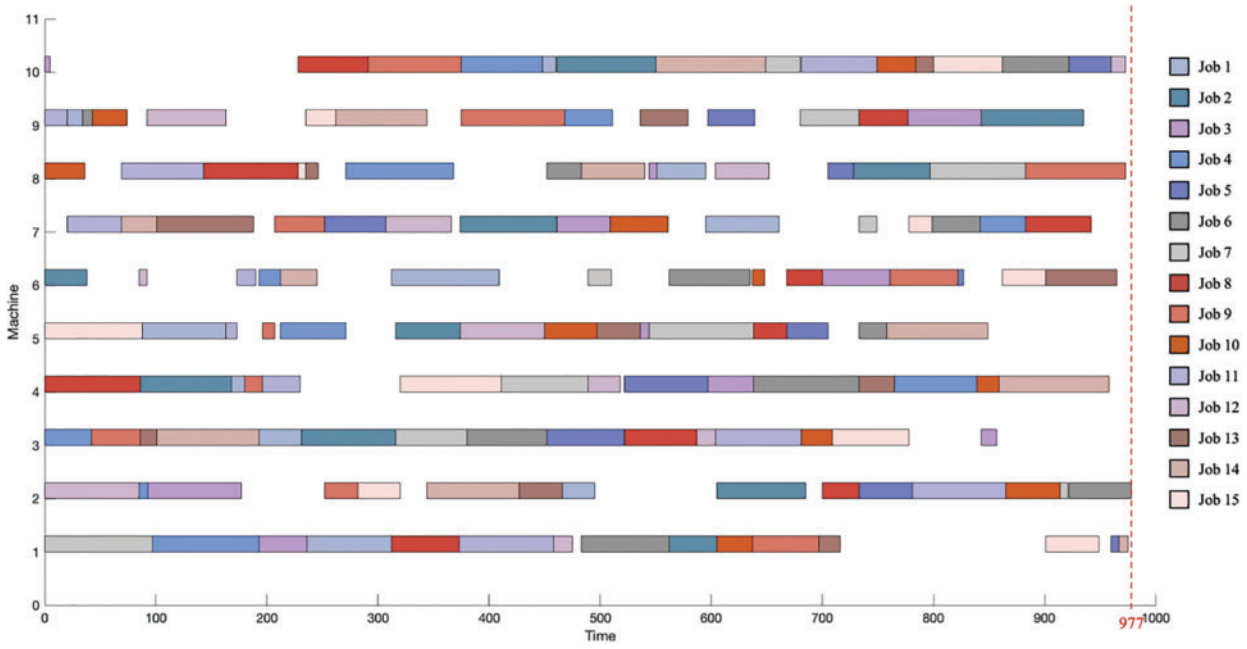

Figure 11: Gantt chart of the best solution of LA25

6 Conclusions

In this paper, a hybrid algorithm that consisted of a Genetic algorithm and Tabu search with a clipping method is proposed for solving large-scale JSSP. A neighborhood solution clipping method is developed and embedded in Tabu search, which can improve the efficiency of the local search by clipping the calculation for unimproved neighborhood solutions. Moreover, a feasible neighborhood solution determination method is developed, which can accurately distinguish feasible neighborhood solutions from infeasible ones. Both of the methods are based on the domain knowledge of JSSP.

The proposed GTS_NC is tested on LA benchmark instance and compared with several algorithms. The computational results obtained in experiments demonstrate the efficiency of the proposed GTS_NC algorithm, which is significantly superior to the other reported methods. In the future, the proposed algorithm can be extended to solve other scheduling problems, such as flexible job shop scheduling problems, dynamic scheduling problems, and so on.

Funding Statement: This work is supported by National Natural Science Foundation for Distinguished Young Scholars of China (under the Grant No. 51825502).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

1. Xiong, H. G., Shi, S. Y., Ren, D. N., Hu, J. J. (2022). A survey of job shop scheduling problem: The types and models. *Computers & Operations Research, 142(2),* 105731. https://doi.org/10.1016/j.cor.2022.105731
2. Gonçalves, J. F., Resende, M. G. C. (2014). An extended Akers graphical method with a biased random-key genetic algorithm for job-shop scheduling. *International Transactions in Operational Research, 21(2),* 215–246. https://doi.org/10.1111/itor.12044
3. Peng, B., Lü, Z. P., Cheng, T. C. E. (2015). A tabu search/path relinking algorithm to solve the job shop scheduling problem. *Computers & Operations Research, 53(3),* 154–164. https://doi.org/10.1016/j.cor.2014.08.006
4. Zhang, C. Y., Li, P. G., Guan, Z. L., Rao, Y. Q. (2007). A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem. *Computers & Operations Research, 34(11),* 3229–3242. https://doi.org/10.1016/j.cor.2005.12.002
5. Lu, C., Zhang, B., Gao, L., Yi, J., Mou, J. H. (2021). A knowledge-based multiobjective memetic algorithm for green job shop scheduling with variable machining speeds. *IEEE Systems Journal, 16(1),* 844–855. https://doi.org/10.1109/JSYST.2021.3076481
6. Liu, Z. F., Wang, J. L., Zhang, C. X., Chu, H. Y., Ding, G. Z. et al. (2021). A hybrid genetic-particle swarm algorithm based on multilevel neighbourhood structure for flexible job shop scheduling problem. *Computers & Operations Research, 135(2),* 105431. https://doi.org/10.1016/j.cor.2021.105431
7. Amico, M. D., Trubian, M. (1993). Applying tabu-search to job-shop scheduling problem. *Annals of Operations Research, 41(3),* 231–252. https://doi.org/10.1007/BF02023076
8. Nowicki, E., Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management Science, 42(6),* 797–813. https://doi.org/10.1287/mnsc.42.6.797
9. Balas, E., Vazacopoulos, A. (1998). Guided local search with shifting bottleneck for job shop scheduling. *Management Science, 44(2),* 262–275. https://doi.org/10.1287/mnsc.44.2.262
10. Laarhoven, P. V., Aarts, E., Lenstra, J. (1992). Job shop scheduling by simulated annealing. *Operations Research, 40(1),* 113–125. https://doi.org/10.1287/opre.40.1.113
11. Monique, S. V., Orides, M. J., Rodrigo, C. (2020). A modified genetic algorithm with local search strategies and multi-crossover operator for job shop scheduling problem. *Sensors, 20(18),* 5440. https://doi.org/10.3390/s20185440
12. Monique, S. V., Orides, M. J., Rodrigo, C. (2022). A new frequency analysis operator for population improvement in genetic algorithms to solve the job shop scheduling problem. *Sensors, 22(12),* 4561. https://doi.org/10.3390/s22124561
13. Hu, R., Wu, X., Qian, B., Mao, J. L., Jin, H. P. (2022). Differential evolution algorithm combined with uncertainty handling techniques for stochastic reentrant job shop scheduling problem. *Complexity, 2022,* 9924163. https://doi.org/10.1155/2022/9924163
14. Ge, Y., Wang, A. M., Zhao, Z. J., Ye, J. R. (2019). A tabu-genetic hybrid search algorithm for job-shop scheduling problem. *E3S Web of Conferences, 95,* 04007. https://doi.org/10.1051/e3sconf/20199504007
15. Abdel-Kader, R. F. (2022). Modified coral reef optimization methods for job shop scheduling problems. *Applied Sciences, 12(19),* 9867. https://doi.org/10.3390/app12199867
16. Constantino, O. H., Segura, C. (2022). A parallel memetic algorithm with explicit management of diversity for the job shop scheduling problem. *Applied Intelligence, 53(1),* 1–13. https://doi.org/10.1007/s10489-021-02406-2
17. Gao, L., Zhang, G. H., Zhang, L. P., Li, X. Y. (2011). An efficient memetic algorithm for solving the job shop scheduling problem. *Computers & Industrial Engineering, 60(4),* 699–705. https://doi.org/10.1016/j.cie.2011.01.003
18. Sharma, N., Sharma, H., Sharma, A. (2018). Beer froth artificial bee colony algorithm for job-shop scheduling problem. *Applied Soft Computing, 68(2),* 507–524. https://doi.org/10.1016/j.asoc.2018.04.001

19. Abdelmonem, M. I., Mohanmed, A. T. (2022). An improved artificial algae algorithm integrated with differential evolution for job-shop scheduling problem. *Journal of Intelligent Manufacturing, 12(1),* 151. https://doi.org/10.1007/s10845-021-01888-8
20. He, L. J., Li, W. F., Chiong, R., Abedi, M., Cao, Y. L. et al. (2021). Optimising the job-shop scheduling problem using a multi-objective Jaya algorithm. *Applied Soft Computing, 111(1),* 107654. https://doi.org/10.1016/j.asoc.2021.107654
21. Alkhatteb, F., Abed-alguni, B. H., AI-rousan, M. H. (2022). Discrete hybrid cuckoo search and simulated annealing algorithm for solving the job shop scheduling problem. *The Journal of Supercomputing, 78(4),* 4799–4826. https://doi.org/10.1007/s11227-021-04050-6
22. Liang, Z. Y., Liu, M., Zhong, P. S., Zhang, C. (2021). Application research of a new neighbourhood structure with adaptive genetic algorithm for job shop scheduling problem. *International Journal of Production Reaserch, 61(2),* 362–381. https://doi.org/10.1080/00207543.2021.2007310
23. Liang, Z. Y., Zhong, P. S., Zhang, C., Liu, M., Liu, J. M. (2021). An improved adaptive genetic algorithm for job shop scheduling problem. *International Conference on Intelligent Equipment and Special Robots*, vol. 1212722. Qingdao, China.
24. Lu, Q., Ren, Y. P., Jin, H. Y., Meng, L. L., Li, L. et al. (2020). A hybrid metaheuristic algorithm for a profit-oriented and energy-efficient disassembly sequencing problem. *Robotics and Computer-Integrated Manufacturing, 61,* 101828. https://doi.org/10.1016/j.rcim.2019.101828
25. Mohanmmad, M. N., Farhad, K. (2012). A GES/TS algorithm for the job shop scheduling. *Computers & Industrial Engineering, 62(4),* 946–952. https://doi.org/10.1016/j.cie.2011.12.018
26. Cheng, T. C. E., Peng, B., Lü, Z. P. (2016). A hybrid evolutionary algorithm to solve the job shop scheduling problem. *Annals of Operation Research, 242(2),* 223–237. https://doi.org/10.1007/s10479-013-1332-5
27. Nagata, Y., Ono, I. (2018). A guided local search with iterative ejections of bottleneck operations for the job shop scheduling problem. *Computers & Operations Research, 90(3),* 60–71. https://doi.org/10.1016/j.cor.2017.09.017
28. Zobolas, G. I., Tarantillis, C. D., Ioannou, G. (2009). A hybrid evolutionary algorithm for the job shop scheduling problem. *Journal of Operational Research Society, 60(2),* 221–235. https://doi.org/10.1057/palgrave.jors.2602534
29. Li, Y., Li, X. Y., Gao, L., Fu, L., Wang, C. Y. (2022). An effective critical path based method for permutation flow shop scheduling problem. *Journal of Manufacturing Systems, 63(5),* 344–353. https://doi.org/10.1016/j.jmsy.2022.04.005
30. Zhang, C. J., Zhou, Y., Peng, K. K., Li, X. Y., Lian, K. L. et al. (2021). Dynamic flexible job shop scheduling method based on improved gene expression programming. *Measurement and Control, 54(7–8),* 1136–1146. https://doi.org/10.1177/0020294020946352
31. Fan, J. X., Zhang, C. J., Liu, Q. H., Shen, W. M., Gao, L. (2022). An improved genetic algorithm for flexible job shop scheduling problem considering reconfigurable machine tools with limited auxiliary modules. *Journal of Manufacturing Systems, 62(1),* 650–667. https://doi.org/10.1016/j.jmsy.2022.01.014
32. Fan, J. X., Shen, W. M., Gao, L., Zhang, C. J., Zhang, Z. (2021). A hybrid Jaya algorithm for solving flexible job shop scheduling problem considering multiple critical paths. *Journal of Manufacturing Systems, 60(1),* 298–311. https://doi.org/10.1016/j.jmsy.2021.05.018
33. Matsuo, H., Suh, C. J., Sullivan, R. S. (1989). A controlled search simulated annealing method for general job shop scheduling problem. *Annals of Operations Research, 21(1),* 85–108. https://doi.org/10.1007/BF02022094
34. Zhao, S. K. (2020). Research on multi-operation joint movement neighborhood structure of job shop scheduling problem. *Journal of Mechanical Engineering, 56(13),* 192–206. https://doi.org/10.3901/JME.2020.13.192

35. Xie, J., Li, X. Y., Gao, L., Gui, L. (2022). A new neighborhood structure for job shop scheduling problems. *International Journal of Production Research, 61(7),* 2147–2161. https://doi.org/10.1080/00207543.2022.2060772
36. Li, X. Y., Gao, L. (2016). An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. *International Journal of Production Economics, 174(19),* 93–110. https://doi.org/10.1016/j.ijpe.2016.01.016
37. Hurink, J. L., Jurisch, B., Thole, M. (1994). Tabu search for the job shop scheduling problem with multi-purpose machines. *Operations Research Spektrum, 15(4),* 205–215. https://doi.org/10.1007/BF01719451
38. Keesari, H. S., Rao, R. V. (2014). Optimization of job shop scheduling problems using teaching-learning-based optimization algorithm. *Opsearch, 51(4),* 545–561. https://doi.org/10.1007/s12597-013-0159-9
39. Pongchairesrks, P. (2019). A two-level metaheuristic algorithm for the job-shop scheduling problem. *Complexity, 2019,* 8683472.