TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 08/24 pp1368–1389
DOI: 10.26599/TST.2023.9010140
Volume 29, Number 5, October 2024

Domain Knowledge Used in Meta-Heuristic Algorithms for the Job-Shop Scheduling Problem: Review and Analysis

Lin Gui, Xinyu Li*, Qingfu Zhang, and Liang Gao

Abstract: Meta-heuristic algorithms search the problem solution space to obtain a satisfactory solution within a reasonable timeframe. By combining domain knowledge of the specific optimization problem, the search efficiency and quality of meta-heuristic algorithms can be significantly improved, making it crucial to identify and summarize domain knowledge within the problem. In this paper, we summarize and analyze domain knowledge that can be applied to meta-heuristic algorithms in the job-shop scheduling problem (JSP). Firstly, this paper delves into the importance of domain knowledge in optimization algorithm design. After that, the development of different methods for the JSP are reviewed, and the domain knowledge in it for meta-heuristic algorithms is summarized and classified. Applications of this domain knowledge are analyzed, showing it is indispensable in ensuring the optimization performance of meta-heuristic algorithms. Finally, this paper analyzes the relationship among domain knowledge, optimization problems, and optimization algorithms, and points out the shortcomings of the existing research and puts forward research prospects. This paper comprehensively summarizes the domain knowledge in the JSP, and discusses the relationship between the optimization problems, optimization algorithms and domain knowledge, which provides a research direction for the meta-heuristic algorithm design for solving the JSP in the future.

Key words: domain knowledge; job-shop scheduling problem; meta-heuristic algorithm

1 Introduction

Optimization problems can be regarded as finding the optimal solution from a finite or infinite set of solutions. Intuitively, the optimal solution can always be found if we enumerate all the solutions and compare them. However, even for a problem with finite solutions, it is usually unacceptable because of the cost by simple enumeration. Therefore, an optimization algorithm needs to be designed that combines the characteristics of the specific problem (such as the structure of the solution space) so as to achieve the purpose of implicit enumeration; that is, only part of solutions are compared for the optimal solution. Here, we regard the characteristics of the optimization problem as domain knowledge, a kind of prior knowledge of the specific problem. The "*No Free Lunch*" theorems[1] also state that combining domain knowledge of a specific problem can improve the performance of the algorithm. Although domain knowledge means different things in different problems, this paper refers to the properties of an individual solution or the relationships between solutions, which can be used in the algorithm to improve its effectiveness and efficiency.

• Lin Gui, Xinyu Li, and Liang Gao are with State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China. E-mail: whguilin@163.com; lixinyu@mail.hust.edu.cn; gaoliang@mail.hust.edu.cn.
• Lin Gui and Qingfu Zhang are with Department of Computer Science, City University of Hong Kong, Hong Kong, China. E-mail: qingfu.zhang@cityu.edu.hk.
* To whom correspondence should be addressed.
Manuscript received: 2023-07-18; revised: 2023-11-11; accepted: 2023-11-13

This paper focuses on the job-shop scheduling problem (JSP), which widely exists in the field of manufacturing[2, 3]. JSP is a classic combinatorial optimization problem. To solve this problem, many different types of optimization algorithms have been proposed, such as heuristic rules[4, 5], exact algorithms[6], approximate algorithms[7], meta-heuristic algorithms[8], expert systems[9], neural networks[10] and different kinds of artificial intelligence method[11−13]. Among them, the meta-heuristic algorithm is the most effective one, because it has the ability to jump out of the local optimum during its search[14].

In the past decades, most of the researchers have preferred to use domain knowledge when designing meta-heuristic algorithms. However, to the best of our knowledge, it is still a fuzzy concept in the existing research, which usually be obtained and used according to the intuition and experience of researchers[15]. Therefore, the main task of this paper is to summarize, classify and analyze the domain knowledge used in the design of meta-heuristic algorithms for the JSP. This paper firstly surveys the development of methods for solving the JSP. After that, the domain knowledge which can be used in the meta-heuristic algorithms for the JSP is summarized and classified, and their applications and effectiveness are analyzed. Finally, the shortcomings of the existing research are pointed out and promising research directions are proposed. Obviously, this review article is not only suitable for researchers of shop scheduling problems, but also has reference value for researchers of other optimization problems and optimization methods.

The main contents of this paper are as follows: In Section 2, the development of methods is summarized. Section 3 illustrates the domain knowledge in the JSP and classifies it into four types. Section 4 mainly reviews the applications and effectiveness of domain knowledge in meta-heuristic algorithms for the JSP. Section 5 proposes some promising research directions. Section 6 concludes this paper.

2 Development of Methods for Solving the Job-Shop Scheduling Problem

To the best of our knowledge, Akers Jr and Friedman studied a JSP with two jobs and four machines in 1955[16], and the term "*job shop*" was first formally proposed in the research literature by Sisson in 1959[17]. In nearly 70 years of development, the JSP has been greatly extended by considering the transportation time[18], distributed[19], setup time[20, 21], machine flexibility[22, 23], dynamic events[24], fuzzy processing time[25, 26], multi-objective[27−29] and other characteristics[30, 31] to meet the needs of actual production. So far, there have been many literature reviews related to the JSPs, including the study of the models[32], the solving methods[33], and the JSP with other constraints[34], etc. Different from other reviews, the focus of this paper is not to review the development of the problem or the solution methods, but to summarize and analyze the domain knowledge in the problem that can be applied to algorithm design. It is hoped that this review can provide a basis for the problem solving, and also hope to guide the research ideas of meta-heuristic algorithm design. In this section, we give a brief review of the development of methods for the JSP to emphasize the importance of meta-heuristic algorithms, and show that the domain knowledge used in different types of algorithms is not the same.

JSP can be easily described as follows: there are n jobs, and each job needs to be processed on m different machines with predetermined sequences. Each job can only be processed on one machine at one time, and each machine can only process one job at one time. Preemption is not permitted when one job is processing on a machine[32]. The essential research in this field is the emergence of computational complexity theory in the 1970s. It makes researchers realize that most of the JSPs are NP-hard, which is almost impossible to design an algorithm to obtain the optimal solution within an acceptable time. After that, scholars preferred to find the near-optimal solution through some approximate methods. Parker named the period before the study of computational complexity theory as B.C. (Before Complexity), and the period after it as A.D. (Advanced Difficulty)[35], as shown in Fig. 1. The following will introduce the primary methods for the JSP in these two periods.

In the early stage, scholars put forward many heuristic rules, which became the basis of classical scheduling theory, for solving the scheduling problems in actual production. Heuristic rules for the JSP include Jackson's rule[36], shortest processing time (STP) rule, and the sum of weighted completion times (SWPT)[37], etc. And then, different priority dispatch rules are also proposed[38−41]. Although these heuristic rules can solve the JSP quickly, the results obtained by using these heuristic rules are not satisfactory as the size of

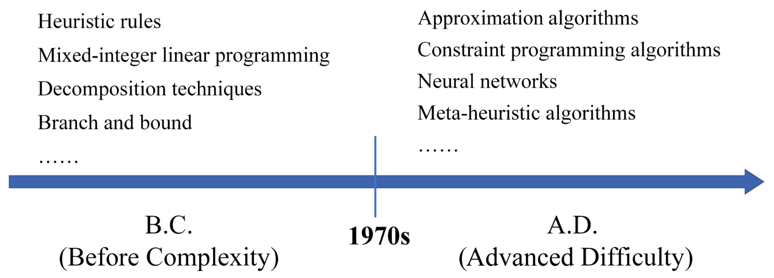

Fig. 1 Two periods of the research for the JSP.

problems increases. Subsequently, many scholars studied how to use the enumeration method, such as the branch-and-bound technology, to solve the shop scheduling problem for the optimal solution. For the JSP, researchers designed active schedule generation branching or used the disjunctive graph to construct a search tree to realize implicit enumeration of all feasible solutions[6, 42]. Although branch-and-bound technology can obtain the optimal solution to the problem, due to the long calculation time required by this method, it can only be applied to the scheduling problem that the number of operations is less than 250, so it is difficult to apply to the actual production problem.

As mentioned before, the 1970s, the time of the emergence of complexity theory research, was the most critical period for the development of shop scheduling problems, which directly brought a significant turn in scholars' research direction. In 1979, Garey and Johnson summarized 320 NP problems, including shop scheduling problems[43]. Later, it was also proved that only a few JSP with special conditions could be solved in polynomial time, and the other general problems are NP problems[44]. After that, people shift the research focus back to the approximation methods, which mainly includes approximation algorithms, constraint programming algorithms, neural networks, heuristic algorithms, and so on. The approximation algorithm gives the multiple of the optimal solution obtained by the algorithm in the worst case by the worst-case analysis or probabilistic analysis. Approximation algorithms are usually designed for parallel machine scheduling problems and flow-shop scheduling problems but are relatively rare for the JSP. There is only some research for special JSP with only two machines or two jobs[45, 46, 47]. The constraint programming algorithm is to reduce the search space through constraint propagation and finally obtain a feasible solution[48]. However, since the ultimate goal of the constraint programming algorithm is to obtain a feasible solution, the results obtained by constraint programming are not ideal. In addition, the expert system and the neural network have been used to solve the JSP in the 1990s[49–51], but they have yet to be further developed due to the poor effect.

The most effective approximate methods for the JSP are meta-heuristic algorithms. The local search is an essential meta-heuristic algorithm, and the first time it proposed for a shop scheduling problem was by Nicholson[52]. However, due to the limitations of the computing power of computers and the fact that it seemed too simple to be worth studying, this method did not receive enough attention until the simulated annealing algorithm was proposed by Kirkpatrick et al.[53] and the tabu search algorithm proposed by Glover[54, 55]. The booming development of this method originated in 1988 when a heuristic algorithm proposed by Adams et al. succeeded in finding the optimal solution of a classical instance (FT 10) of the JSP[56]. So far, many scholars have proposed or improved local search algorithms which can solve the JSP well[57–61]. In addition to the local search algorithms, there are many meta-heuristic algorithms, such as the population-based evolutionary algorithm, which can also solve the JSP very well. Fisher and Rinnooy Kan[62] emphasized the important principles for generating good heuristic techniques: design, analysis and implementation. The common population-based meta-heuristic algorithms for the JSP are genetic algorithms[63, 64], ant colony optimization algorithms[65], particle swarm optimization algorithms[66, 67], various hybrid algorithms[68, 69], and so on. These algorithms guide the evolution of individuals by simulating the behavior of populations of different organisms in nature. Of course, some scholars analyzed the fitness landscape of the JSP[70–72] and designed individual evolutionary strategies in the algorithm according to the characteristics of the fitness landscape[61, 73]. Due to its superior performance in solving the JSP, more and more different types of meta-heuristic algorithms have been designed, such as the chemical-reaction optimization[74], the intelligent water drops algorithm[75], hybrid fruit fly optimisation algorithm[76], the hybrid election campaign optimization algorithm[77] and the imperialist competition algorithm[78]. However, some scholars point out that most of these meta-heuristic algorithms have the same optimization framework but just analogy these algorithms to different biological or human social behaviors, which is not innovative enough[79].

In recent years, with the development of machine learning technology, more and more scholars have begun to study how to solve the shop scheduling problem directly or indirectly by machine learning[80–83]. This method has great advantages for solving dynamic shop scheduling problems because it can achieve offline learning and online optimization, making it respond to the demand of dynamic shop scheduling problems quickly. However, except for this, the meta-heuristic algorithm is still the most advantageous method.

We searched articles with the keyword "*job shop scheduling*" in the title and the literature type "*Article*" in Scopus, among which there were 1731 relevant articles from 2001 to 2022. We classify the literature that uses algorithms to solve problems into two categories: the literature using meta-heuristic algorithms and the literature using other methods. The changing trend of the publication of these two types of literature in the past 22 years is shown in Fig. 2. Although the results retrieved in this paper are not necessarily accurate, they can reflect the development trend of solving methods for the JSP; that is, the meta-heuristic algorithm has been the mainstream method for the last 20 years.

The meta-heuristic algorithm is the most popular method to solve the JSP. One of the important reasons is that a large amount of related domain knowledge can guarantee the optimization performance of the algorithm. It can also be seen that domain knowledge contained in the problem is not universal for all types of algorithms because of the essentially different optimizations of different algorithm types. For example, heuristic rules construct a solution through several choices, so the domain knowledge should be related to make each choice as best as possible. The branch and bound algorithm obtains the optimal solution of a problem by means of implicit enumeration, so the domain knowledge should be related to reducing the number of solutions to improve the optimization efficiency. In this paper, we focus on the meta-heuristic algorithm, which is an algorithm that searches in the solution space of the problem, so the domain knowledge should be related to solutions.

3 Domain Knowledge for Designing Meta-Heuristic Algorithms for the JSP

The "*No Free Lunch*" theorem states that no one algorithm is better than another one on all optimization problems, which indicates that it is important to incorporate domain knowledge into the algorithm for a

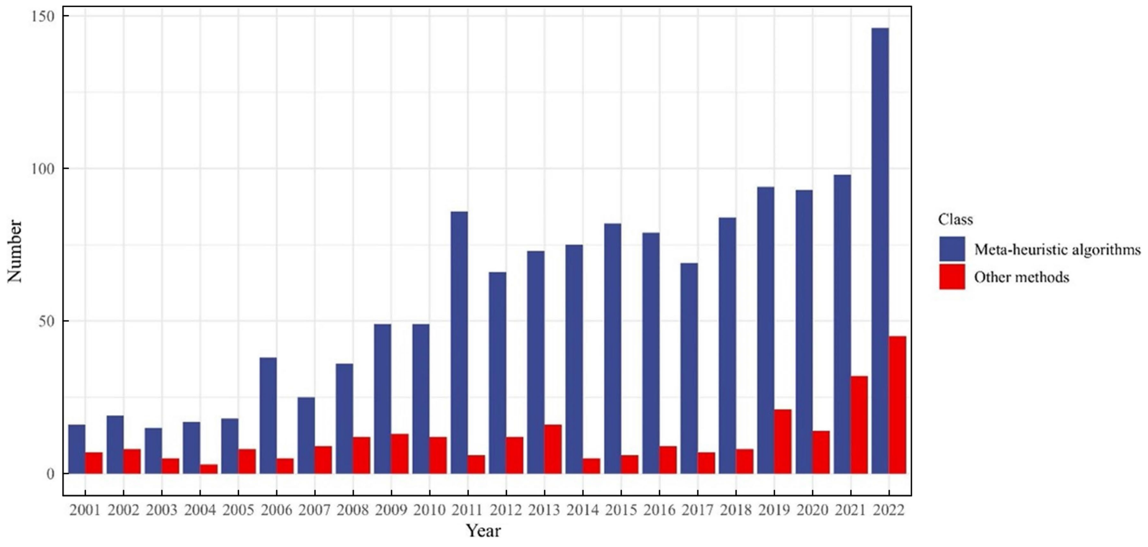

Fig. 2 Trend of the methods for the JSP in the last 20 years.

specific problem[1]. In fact, domain knowledge can be regarded as a kind of prior knowledge of the problem, and the performance of the optimization algorithm can be improved by using this knowledge. This section summarizes the domain knowledge in the JSP used for designing meta-heuristic algorithms and classifies this domain knowledge according to the optimization process of this algorithm. Generally, when it uses the meta-heuristic algorithm to solve the JSP, the problem needs to be mapped to the coding space first, and then the algorithm searches in the space through some neighborhood structures or search strategies. Finally, the final optimization result is obtained by decoding the searched solution. Therefore, the domain knowledge applied to the meta-heuristic algorithm can be divided into four types according to the execution process of the algorithm: 1) Problem representation; 2) Global characteristics of the solution space; 3) Local characteristics of the solution space; 4)Types of solutions. The relationship between domain knowledge and meta-heuristic algorithms can be shown in Fig. 3.

3.1 Problem representation

In fact, problem representation can be understood as the process of modeling or coding, and the search space of the algorithm is determined by choice of problem representation. The most common problem representation is a mathematical programming model, which can represent any optimization problem in terms of decision variables, constraints, and objective functions. However, this representation can only be solved by mathematical programming methods. In the meta-heuristic algorithms, the commonly used problem representations for the JSP are the Gantt chart and the disjunctive graph. Of course, other representations, like the permutation-disjunctive graph, have been proposed by some scholars, but these are also essentially combinations of representations of Gantt and the disjunctive graph. In this part, we use simple examples to show the Gantt chart and the disjunctive graph and briefly introduce the history of these two representations.

3.1.1 Gantt chart

Gantt chart is a kind of bar chart to show the start and end time of each task, and it can intuitively represent the time relationship between all tasks so that resources can be correctly allocated. Gantt chart has been widely used in various task arrangement activities, including the JSP. A simple example is given for it. Suppose there are three jobs ($O_{1,1}$, $O_{1,2}$,$O_{1,3}$), ($O_{2,1}$, $O_{2,2}$, $O_{2,3}$), ($O_{3,1}$, $O_{3,2}$, $O_{3,3}$), where the notation $O_{j,i}$ means the i-th operation of the j-th job. These operations are processed on three machines, and [$m_{j,i}$, $p_{j,i}$] represents the machine and time of the operation $O_{j,i}$. The processing information of these operations are ([1, 3], [2, 3], [3, 2]), ([1, 3], [3, 2], [2, 2]), and ([1, 3], [3, 2], [2, 3]). The JSP needs to decide the processing sequence of the jobs on each machine to minimize the makespan of the whole scheduling scheme. With the information given above, we can draw a Gantt chart by determining the processing sequence of the jobs on each machine, as shown in Fig. 4.

The Gantt chart has a long history of development. As early as the mid-1890s, Polish engineer Karel Adamiecki proposed a Harmonogram to represent the workflow network diagram, which could show the start and end time of the execution of different tasks[83]. In 1912, Swiss engineer Herman Schurch introduced a graphical and quantitative chart called “*bauprogramm*” for management projects. He used a histogram to represent the workload of different jobs and a shadow effect to represent different types of jobs. In 1913, Henry Laurence Gantt published the book *Work*, *Wages*, *and Profits*[84], which summarized his management

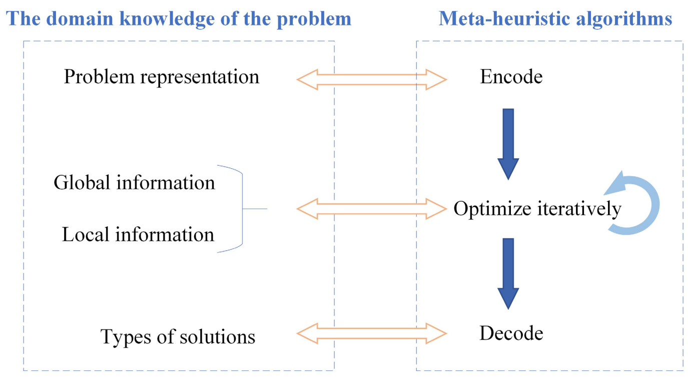

Fig. 3 Relationship between domain knowledge and meta-heuristic algorithms.

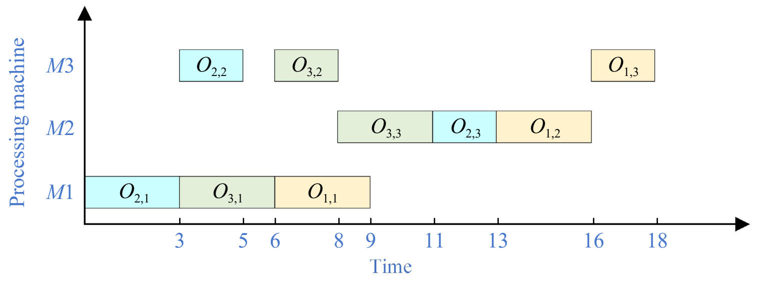

Fig. 4 Gantt chart of a solution for the JSP.

experience from the production practice, including the use of the Gantt chart. In later work, Gantt used this chart to discuss scheduling problems. It is worth mentioning that Henry Laurence Gantt did not name this kind of graph as a Gantt chart in the book. Three years after Mr. Gantt's death, Clark published the book *The Gantt chart: A working tool of management*[85] to commemorate Mr. Gantt's contribution to management science and officially named this kind of chart the Gantt chart. Later, the Gantt chart was widely used in the shop scheduling problem to express the processing time and production sequence of jobs.

With the Gantt chart, it is easy to know about the start and end time of each operation, and people can also use this Gantt chart to improve the schedule by different methods. In the very beginning, it is a common way to look at the Gantt chart to see if the scheduling scheme can be improved by changing the order of operations on certain machines. With the increase in the number of machines and jobs, the method of manual observation has become no longer effective. It is necessary to give the processing sequence of each machine through an appropriate algorithm. In the meta-heuristic algorithm, a scheduling scheme can be expressed by a permutation for each machine, which corresponds to the processing sequence of each machine in the Gantt chart.

3.1.2 Disjunctive graph

The disjunctive graph is a kind of network graph that uses vertices to represent the operations in jobs, connecting arcs to represent the relationships between operations in the same jobs, and disjunctive arcs to represent the relationships between operations that should be processed on the same machine. By deciding the direction of each disjunctive arc (that is, deciding the processing sequence of two operations on the same machine), the final processing sequence on each machine can be determined, and finally, the scheduling scheme can be obtained. A disjunctive graph can be represented by a triple $G=(V, A, E)$, where V denotes the set of vertices in G, A denotes the set of connecting arcs in G, and E denotes the set of disjunctive arcs in G. For the convenience of representation, two virtual vertices, s and e, which are used to represent the starting point and the ending point of all the jobs, respectively, are generally added to the disjunctive graph, as shown in Fig. 5a.

In the disjunctive graph, the length of the arc or the weight of the vertex can be expressed as the processing time of the corresponding operation. For ease of understanding, the weight of the vertex is set to the processing time of the operation in this paper, and the weight of the virtual vertex (s and e) and the length of all arcs are 0. By determining the direction of the disjunctive arc and calculating the longest path from the starting vertex to the end vertex, the maximum completion time of the scheduling scheme can be obtained, as shown in Fig. 5b. For a more concise picture, some redundant disjunctive arcs have been removed, such as the arc from $O_{2,1}$ to $O_{1,1}$. Deletions

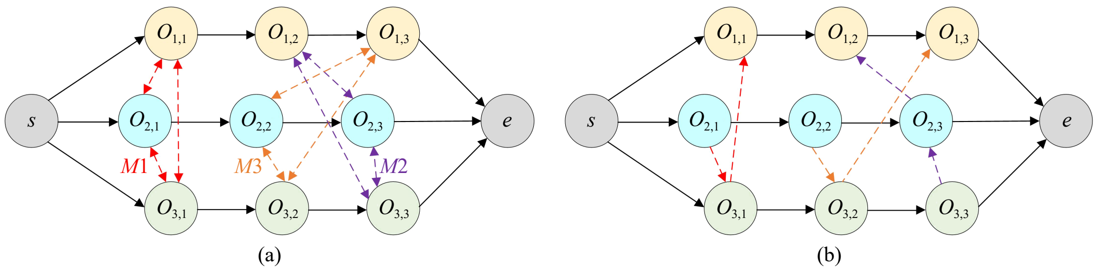

Fig. 5 Disjunctive graph of the JSP.

like this will also be adopted in later disjunctive graphs and will not be elaborated on in the following. In fact, each disjunctive arc in the disjunctive graph can be regarded as a (0,1) variable, and the direction of the disjunctive arc is determined by the value of each variable. However, if it is not properly decided, it will make the disjunctive graph form cycles, as shown in Fig. 6a. It is obvious that the solution represented by the disjunctive graph with a cycle is infeasible because it is impossible to determine which operation starts first among these several operations in the cycle. In fact, infeasible solutions will also appear if the processing sequence of the jobs is not properly changed in the Gantt chart. However, they cannot be displayed in the Gantt chart, so the infeasible solutions in the JSP are generally discussed in the disjunctive graph.

Roy and Sussman[86] first proposed the use of a disjunctive graph to represent the JSP. Balas[6] designed an exact algorithm based on the disjunctive graph, and then more and more scholars used it to solve the JSP. White and Rogers[87] described the extensions and limitations of the disjunctive graph. They successfully extended the disjunctive graph model to represent the JSP with assembly and disassembly operations, due dates, scheduled maintenance, setup time, priority tasks, and so on. But they also pointed out that it may be difficult to apply the model to industrial situations because parallel machines cannot be modeled directly. Gui et al.[88] used a disjunctive graph to represent the hybrid flow-shop scheduling problem, which means the parallel machines can also be represented by the disjunctive graph. Bazewicz et al.[89] pointed out that the disjunctive graph was more popular than the Gantt graph to represent the JSP. Since the development of the network graph is more mature than the scheduling problem, using the disjunctive graph to represent the JSP can make full use of the existing knowledge in the network graph, such as the critical path. At the same time, the disjunctive graph can be more conducive to the analysis of the problem, such as the feasibility of a solution. However, there will be a large number of infeasible solutions when using the disjunctive graph to make a decision on the problem. These infeasible solutions stem not only from the fact that operations on different machines in the disjunctive graph jointly produce cycles, but also from the fact that operations on the same machine produce cycles, as shown in Fig. 6b. This makes it impossible to represent the processing sequence of operations on the same machine by a permutation. To this end, Nowicki and Smutnicki proposed the permutation-graph model[90], which essentially combines the advantages of the Gantt graph and disjunctive graph to express the job shop scheduling problem, and now it has become the most commonly used expression form.

3.2 Global characteristics of the solution space

Global characteristics of the solution space refers to the indexes or properties used to describe the overall trend of optimization space, such as linearity/non-linearity and convexity/non-convexity in continuous optimization problems. With the appropriate global characteristics, it can effectively guide the iteration direction of the algorithm to find the optimal solution. For example, suppose a continuous optimization problem is convex. In that case, the optimal solution of the problem must be at the point that the derivative is 0 or the edge of the solution space, and the optimal solution can be found along the direction of the derivative descent, which provides great convenience for the optimal solution of convex optimization problems.

For combinatorial optimization problems such as the JSP, we usually use the fitness landscape to describe the global characteristics of a problem. The concept of fitness landscape was proposed by Wright for the field of biological evolution[91], and then this concept was gradually adopted in the field of optimization to represent the solution space of a problem. In the existing research, there is not too much research on the fitness

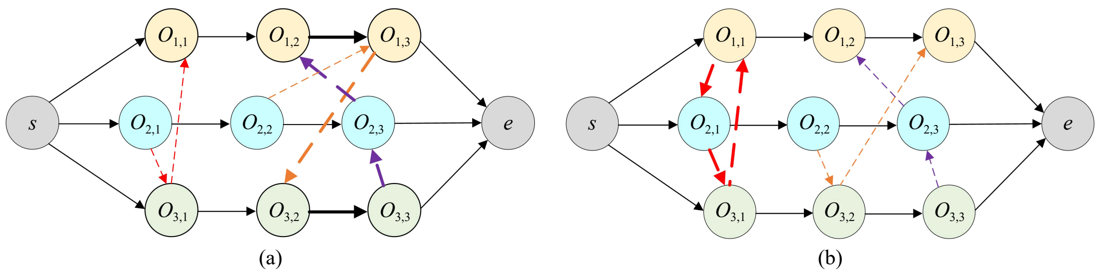

Fig. 6 Infeasible solutions for the JSP.

landscape analysis of the JSP. On the one hand, the definition of some parameters required to describe the fitness landscape of the problem is not uniform, such as the definition of distance between two solutions, which may lead to different conclusions for the same problem. On the other hand, the fitness landscape of this problem is very complex, so it is difficult to find out the rule directly to guide the search. Mattfeld et al.[70] analyzed the fitness landscape of the JSP and showed that the inherent characteristics of the problem had an impact on the heuristic algorithm based on local search. Bierwirth et al.[71] showed that the fitness landscape of the JSP is irregular. Streeter and Smith[72] showed that the fitness landscape of the JSP is related to the problem scale, and it was verified that JSP has "*big valley*" terrain in different instances. The description of the fitness landscape for this problem can provide some guidance for the design of the algorithm, such as the path relinking used in the algorithm for "*big valley*" .

3.3 Local characteristics of the solution space

Local characteristics of the solution space can be considered as the neighborhood characteristics, which indicates the relationship between the neighborhood solutions obtained from the current solution by small perturbation. In the JSP, the usage of the local characteristics can be divided into three categories: 1) How to obtain a neighborhood solution that is better than the current solution; 2) How to obtain feasible neighborhood solutions; 3) How to use the characteristics of the current solution to obtain the objective function value of the neighborhood solution. These can be summarized as the quality of neighborhood solutions, the feasibility of neighborhood solutions, and the evaluation of neighborhood solutions. By entirely using these three kinds of local characteristics, the quality and efficiency of the algorithm can be significantly improved. In this paper, we mainly introduce some properties in local characteristics of the JSP, and detailed proof can be obtained from the corresponding literature. In the JSP, the local disturbances are usually generated by selecting one operation and inserting it before or after the other operations, which are processed on the same machine. Of course, some disturbances choose multiple operations and change their processing sequences simultaneously, but these disturbances can be obtained by the former through multiple steps, so the relevant information on neighborhood solutions of multiple disturbances is not involved in this paper. Next, this paper will elaborate on the specific local characteristics in the JSP.

3.3.1 Quality of neighborhood solutions

It is known that most of the neighborhood solutions generated by perturbation are worse than the current ones. These neighborhood solutions not only make no sense for the iteration of the algorithm but also incur a huge computational cost. Therefore, how to avoid generating solutions that do not improve the current solution is the key to effectively improving the quality of the neighborhood solution, and the critical path method is the most important method. The critical path method is a network graph method in which the critical path refers to the longest path between two points in the network graph. It was proposed by Kelley Jr and Walker[92] in 1959 to solve the project scheduling problem, and Conway et al.[93] introduced the critical path method into the JSP, which was used to represent the longest distance from the start vertex to the end vertex in the disjunctive graph (that is, the maximum completion time of the solution). Since then, most of the research on the neighborhood solution of the JSP has been based on the critical path. However, although the critical path was first applied to the concept of the disjunctive graph, it does not intuitively express the relationship between the critical path and other operations in the whole scheduling scheme. On the contrary, this relationship can be clearly found in the Gantt chart, so the Gantt chart is mainly used to express and explain the related concepts or properties in this part.

Potts[94] pointed out the role of the critical path in the JSP; that is, when the processing sequence of operations on the critical path does not change, the total completion time will not decrease. This points out that the optimization bottleneck of the JSP is the processing sequence of the operations on the critical path, and changing the processing sequence of other operations will not improve the current solution. This finding dramatically reduces the number of neighborhood solutions in the JSP. Grabowski et al.[95] proposed the concept of the critical block based on the critical path, a block composed of operations processed on the same machine and with adjacent processing time in the critical path. The Gantt chart in Fig. 4 identifies the critical path as shown in Fig. 7, which contains four critical blocks.

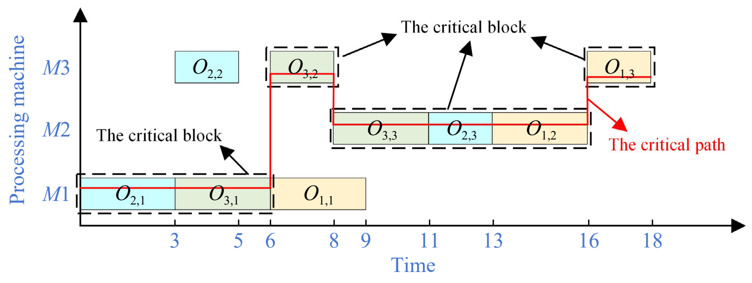

Fig. 7 Critical path and critical blocks in the Gantt chart.

Based on the above, scholars improve the quality of neighborhood solutions by further analyzing the critical path blocks. Matsuo et al.[8] pointed out that if the processing sequence of operations is only changed inside the critical block, the total makespan will not decrease. Bazewicz et al.[89] pointed out that if other operations were inserted before the first operation in the first critical block, the total completion time would not decrease, and if other operations were inserted after the last operation in the last critical block, the total completion time would not decrease. In addition, there is some other domain knowledge that can improve the quality of neighborhood solutions. Although they do not appear in the literature, they are also introduced in this paper, and the corresponding proof is given. To better describe this domain knowledge, the definitions of symbols are given below:

• x: one operation of the schedule;

• m_x: the machine used to process x;

• p_x: the processing time of the x;

• jp[x]: the operation in the same job as x, and processing just before x;

• js[x]: the operation in the same job as x, and processing just after x;

• mp[x]: the operation processed on the same machine as x, and processing just before x;

• ms[x]: the operation processed on the same machine as x, and processing just after x;

• $F(x)$: the maximum weight sum of a path from s to x in the disjunctive graph (x is not included);

• $R(x)$: the maximum weight sum of a path from x to e in the disjunctive graph (x is not included);

Assuming that the operations on one critical block are (O_1, O_2, ..., O_k), as shown in Fig. 8. Therefore, it is easy to know that if the first operation or the last operation on this block is not changed, the makespan of neighborhood solutions will not be reduced. In the following, we will give four other properties which can use constraints to judge whether one neighborhood solution is worse than the current one so as to reduce the evaluation times of neighborhood solutions.

Proposition 1 Assuming that the operation O_t belongs to $(O_2, \ldots, O_{k-1})$. If $F(\text{jp}[O_t])+p_{\text{jp}}[O_t] \geqslant F(O_1)$, the neighborhood solution obtained by moving O_t to the position just before O_1 will not be better than the current one.

Proof The makespan of the current solution can be calculated. by the formulation: $C = F(O_1)+\sum_{i=1}^{k} p(O_i)+R(O_k)$. If a neighborhood solution obtained by moving O_t to the position just before O_1, the makespan of this neighborhood solution is satisfied with the inequality: $C' \geqslant F(\text{jp}[O_t]) + p_{\text{jp}[O_t]} + \sum_{i=1}^{k} [p(O_i)] + R(O_k) \geqslant C$. ■

Proposition 2 Assuming that the operation O_t belongs to $(O_2, \ldots, O_{k-1})$. If $F(\text{jp}[O_2])+p_{\text{jp}[O_2]} \geqslant F(O_1)$, the neighborhood solution obtained by moving O_1 to the position just after O_t will not be better than the

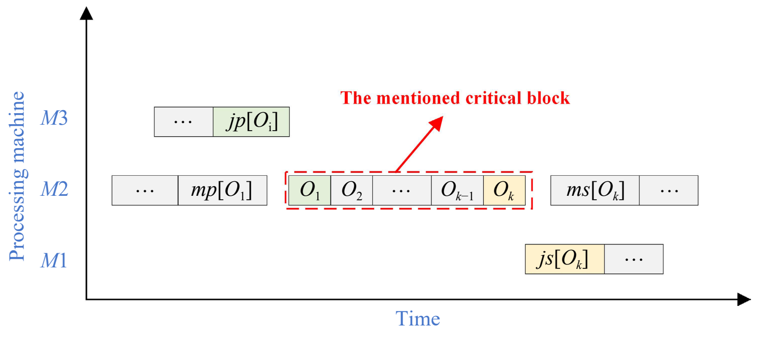

Fig. 8 Illustration of the critical block mentioned above.

current one.

Proof The makespan of the current solution can be calculated. by the formulation: $C = F(O_1)+\sum_{i=1}^{k} p(O_i)+ R(O_k)$. If a neighborhood solution obtained by moving O_1 to the position just after O_t, the makespan of this neighborhood solution is satisfied with the inequality:$C' \geqslant F(\text{jp}[O_2]) + p_{\text{jp}[O_2]} + \sum_{i=1}^{k} [p(O_i)] + R(O_k) \geqslant C$. ■

Proposition 3 Assuming that the operation O_t belongs to $(O_2, \ldots, O_{k-1})$. If $R(js[O_t])+p_{js[O_t]} \geqslant R(O_k)$, the neighborhood solution obtained by moving O_t to the position just after O_k will not be better than the current one.

Proof Similar to the proof of Proposition 1. ■

Proposition 4 Assuming that the operation O_t belongs to $(O_2, \ldots, O_{k-1})$. If $R(\text{js}[O_{k-1}])+p_{\text{js}[O_{k-1}]} \geqslant R(O_k)$, the neighborhood solution obtained by moving O_k to the position just before O_t will not be better than the current one.

Proof Similar to the proof of Proposition 2. ■

3.3.2 Feasibility of neighborhood solutions

When changing the processing sequence of the operations on the critical path block, there are generally two methods to avoid the infeasible neighborhood solutions: 1) avoid the infeasible solutions by using some constraints; 2) transform the infeasible into the feasible when the neighborhood solution is found to be an infeasible solution. This paper will not discuss which method is better, but only describe the relevant domain knowledge.

There are few feasibility studies on the JSP. Van Laarhoven et al.[57] found that the neighborhood solution obtained by exchanging the processing sequence of any two adjacent operations on any critical path block must be feasible and proved its correctness. Balas and Vazacopoulos[60] proposed a set of constraint conditions and proved that the neighborhood solution generated by inserting the operation before or after the other operation must be the feasible solution if the conditions were met. It is assumed that u and v are operations processed on the same machine, and u is processed before v in the current solution. When the constraint conditions $R(v)+p_v \geqslant R(\text{js}[u])+p_{\text{js}[u]}$ are satisfied, the neighborhood solution obtained by inserting u into the position just after v is a feasible solution. When the constraint conditions $F(u)+p_u \geqslant F(\text{jp}[v])+p_{\text{jp}[v]}$ are satisfied, the neighborhood solution obtained by inserting v into the position just before u is a feasible solution. Balas's work gives constraints on the generation of feasible solutions by arbitrary insertion, not just adjacent swapping. However, since the constraint conditions proposed by Balas are sufficient but not necessary conditions for feasible solutions, that is, although such constraints can ensure that the neighborhood solutions obtained are feasible, some feasible neighborhood solutions will also be deleted as infeasible solutions. Gui et al. gave the necessary and sufficient conditions for feasible solutions by analyzing the necessary and sufficient conditions for infeasible solutions[96]. That is, it is assumed that u and v are operations processed on the same machine, and u is processed before v in the current solution. When there is no path from js(u) to v in the disjunctive graph, the neighborhood solution obtained by inserting u into the position just after v is feasible. When there is no path from u to jp(v) in the disjunctive graph, the neighborhood solution obtained by inserting v into the position just before u is feasible. By giving the necessary and sufficient conditions for feasible solutions of the JSP, all feasible neighborhood solutions near the current solution can be searched, making the local search more adequate.

3.3.3 Evaluation of neighborhood solutions

The evaluation of neighborhood solutions is realized by using the information of the current solution to estimate the function value of neighborhood ones to reduce the number of re-decoding of neighborhood solutions. The evaluation of neighborhood solutions needs to ensure the accuracy of the estimated value and the computational speed, which needs to make full use of the local information in the problem.

Taillard proposed an evaluation method for exchanging two adjacent operations on critical path blocks to generate neighborhood solutions[97]. Suppose that u is processed before v. Since only the processing order of these two operations changes in the disjunctive graph of the neighborhood solution, the other operations unrelated to u and v in the graph will not undergo any change. For the operations that are related to u and v, if these operations are preceded by u and v, the longest path from the start vertex to these operations will not change, and if these operations follow u and v, the longest path from these operations to the end vertex does not change. This unchanged information can be used to compute the longest path from the start vertex to v and u, and the longest distance from v and u to the end vertex. We can then compute the longest path that goes through v or u from the start vertex to the end vertex. The maximum value among them is used as the evaluation value of the neighborhood solution. The specific calculation formula is: $C'=\max\{F'(v)+p_v+R'(v),\ F'(u)+p_u+R'(u)\}$, where $F'(x)=\max\{F'(\mathrm{jp}[x])+p_{\mathrm{jp}}[x], F'(\mathrm{mp}[x])+p_{\mathrm{mp}[x]}\}$ and $R'(x)=\max\{R'(\mathrm{js}[x])+p_{\mathrm{js}[x]},\ R'(\mathrm{ms}[x])+p_{\mathrm{ms}[x]}\}$. In the paper, the authors prove that this way of estimating the neighborhood solution is the lower bound of the neighborhood solution. Balas et al.[60] proposed the evaluation method of generating neighborhood solutions by inserting an operation before or after other operations on the critical path block. The overall idea of this method is similar to that proposed by Taillard, that is, using the information of the current solution to calculate the length of the longest path, which simultaneously goes through the start vertex, the end vertex, and any operation that the processing order is changed. The calculation is as follows: $C'=\max\{F'(v)+p_v+R'(v),\ F'(u)+p_u+R'(u),\ F'(x)+p_x+R'(x)\}$, where x belongs to the operations between u and v in the critical path block. Nowicki and Smutnicki[61] improved the method of Taillard; that is, in addition to calculating the longest path in the neighborhood solution that passes through v and u, it also needs to calculate the longest path that does not pass through u and v, and finally take the maximum value among them as the function value of the neighborhood solution. The authors mentioned in the paper that if the longest path through u and v is greater than the function value of the current solution, it is not necessary to calculate the longest path without u and v. Since the longest path length from the start vertex to the end vertex is the function value of the solution, it is known that the evaluation value obtained by the method of Nowicki et al. is the exact function value of the neighborhood solution. However, this method is not widely used because it is too complex to calculate the longest path without passing through u and v. It needs to transform the disjunctive graph of the neighborhood solution into a topological sort and calculate the longest path of all operations before v and after u. Gui et al.[98] improved Balas et al.'s method and proved that it only needs to calculate the longest path through u and v in the neighborhood solution to obtain the same effect as Balas and Vazacopoulos's method[60]. With this property, the makespan can be calculated as $C' = \max\{F'(v)+p_v+R'(v),\ F'(u)+p_u+R'(u)\}$, where the operations between u and v in the critical path block do not need to be calculated again.

3.4 Types of solutions

In the JSP, when the processing sequence of the operations on each machine is determined, it can also obtain infinite different scheduling schemes because infinite different idle-time can be inserted between the operations. It is clear that scheduling schemes that insert idle time are meaningless, so researchers have defined several different types of solutions to avoid mostly meaningless scheduling schemes. Although some of these types were proposed to be applicable to earlier heuristic rules, the analysis of these solution types is beneficial for us to deepen our understanding of the JSP. These different scheduling types have a certain subsumption relationship with each other, which is described in detail as follows.

3.4.1 Non-delayed schedule

The non-delayed schedule refers to a solution where the corresponding machine is idle when there is no job waiting. This type of solution was first proposed by Jackson[4], and it was called availability schedule at that time, and later renamed as non-delayed scheduling by Nugent[99] in his doctoral thesis. Using the Gantt chart in Fig. 4 as an example, it is obvious that this solution is not a non-delayed schedule because the operation $O_{2,3}$ waits from time 5, and the machine $M2$ is idle. If we change the processing sequence of $O_{3,3}$ and $O_{2,3}$, then it is a non-delayed schedule, although the start time of $O_{2,3}$ is delayed. As it is shown in Fig. 9a.

3.4.2 Active schedule

The active schedule refers to a solution in which no operation can advance its start time by moving to the

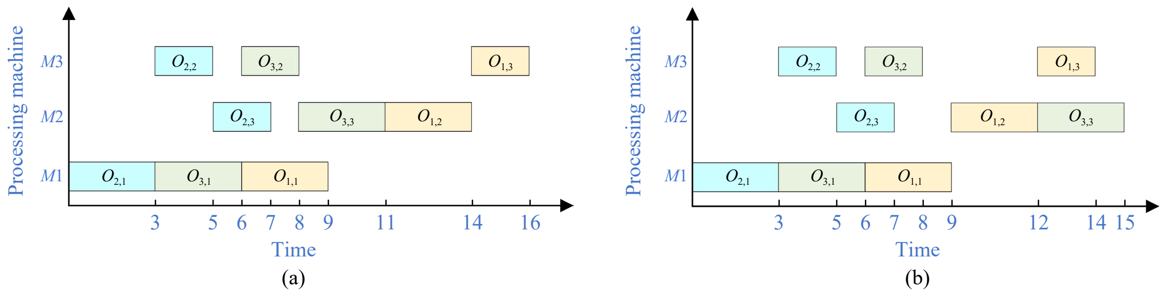

Fig. 9 Gantt chart of a non-delayed schedule and a full-active schedule.

left in a scheduling scheme on the premise of not delaying the start processing time of other operations. This solution type was first introduced by Giffler et al.[38] and proved that the optimal scheduling scheme must be in the set of active schedules of the problem. Using the Gantt chart in Fig. 4 as an example, it is obviously that this solution is an active schedule, because no matter which operation starts earlier, there is at least one operation should delay its start time.

3.4.3 Semi-active schedule

The semi-active schedule refers to a solution in which no operation can be processed earlier when the processing sequence of the jobs is not changed. This solution type first introduced by Nugent[99], which makes the processing time of each operation a unique value when the processing sequence of the processes on the machine is determined. Using the Gantt chart in Fig. 4 as an example, it is obviously that this solution is a semi-active schedule, because there is no idle time inserted into the solution. With the semi-active schedule, it is reasonable for us to represent a solution by a sequence. It is easy to know that the optimal must be the semi-active schedule.

3.4.4 Full-active schedule

The full-active schedule refers to a solution in which no operation can be moved to the left or right to make the scheduling scheme better on the premise that the start time of other operations is not delayed. This type of schedule was first proposed by Zhang et al.[100] and proved that the optimal scheduling scheme must be in the set of full-active schedule of the problem. Using the Gantt chart in Fig. 4 as an example, we can know that the solution is not a full-active schedule because when the operation $O_{2,3}$ moves right to the position just after $O_{1,2}$, the solution will be better and the start time of other operations is not delayed, as it is shown in Fig. 9b. From the definitions of different types of solutions, we can see that there is an inclusion relation between the types of these solutions, and the Venn diagram of this inclusion relation is shown in Fig. 10. The optimal solution in the figure is in the full-active scheduling solution, but as said above, for small-scale problems, the optimal solution may also be a non-delayed scheduling solution.

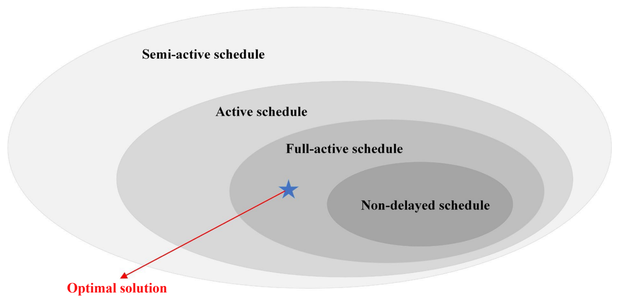

Fig. 10 Relationship between different types of solutions.

4 Analysis of the Relationship between Domain Knowledge and Meta-Heuristic Algorithms

The domain knowledge in the JSP has been summarized and classified above. This section will elaborate on the corresponding relationship between domain knowledge and meta-heuristic algorithms, as well as its application in meta-heuristic algorithms. The effectiveness of this domain knowledge in meta-heuristic algorithms is analyzed through classical literature. Finally, the relationship among optimization problems, domain knowledge and optimization methods is analyzed.

4.1 Applications of domain knowledge in meta-heuristic algorithms

The domain knowledge mentioned in Section 3 is classified according to the execution process of the algorithm. Since it is obvious to design the encoding and decoding of meta-heuristic algorithms by using the problem representations and the types of the solution, this part mainly introduces how to use the global information and local information in the JSP to design the optimization operator in the meta-heuristics.

Because there are many local optimal in the search space of the JSP, the global information is mainly used to guide algorithms to jump out of the local area and provide new areas worth further exploration. However, as can be seen from the above, the mining of the global characteristics of the JSP in the existing research is not sufficient. Only several pieces of literature proposed that the fitness landscape of the JSP has the characteristics of the big valley. Based on this, some scholars have proposed using path relinking to search the space between two local optimal points, which can effectively solve the optimization problem with many local optimal points. The experimental results of some literatures also show that path relinking can effectively solve the JSP[73, 89, 101].

The local characteristics in the JSP are generally used to design neighborhood structures of the algorithm and evaluation methods for neighborhood solutions. The use of the evaluation of neighborhood solutions has been elaborated in detail in Section 3, so the applications for neighborhood structures is mainly introduced in this part. The neighborhood structure is a perturbation rule on the current solution to produce a set of neighborhood solutions. The design of neighborhood structure in the JSP should follow two principles[102]: 1) Reduce the number of neighborhood solutions that do not improve the current solution; 2) The generated neighborhood solution must be feasible. This makes the neighborhood structure need to combine the local information in the JSP. Since the main purpose of this paper is to review the domain knowledge in the JSP rather than neighborhood structures, so this part only introduces the five most effective neighborhood structures and explains how domain knowledge is used in them. This paper will use the critical block to introduce them, where the operations in the critical block are (O_1, O_2, …, O_k), as shown in Fig. 11.

In order to compare the differences between these neighborhood structures, we use a table to represent the perturbation operation, the domain knowledge about quality and feasibility used, and the number of neighborhood solutions that can be obtained on a critical block for each neighborhood structure. The details are shown in Table 2.

(1) If the processing sequence of operations on the critical path does not change, the total makespan will not decrease;

(2) If the processing sequence of operations is only changed inside the critical block, the total makespan will not decrease;

(3) If operations in the first critical block move to the position before the first operation, the total completion time will not decrease; If operations in the last critical block move to the position after the last operation, the total completion time will not decrease;

(4) The neighborhood solution obtained by exchanging the processing sequence of any two adjacent operations of any one critical path block must be a feasible solution;

(5) Assuming that u and v are in the critical block, and u is processed before v. When the constraint conditions $R(v)+p_v \geqslant R(js[u])+p_{js[u]}$ are satisfied, the neighborhood solution obtained by inserting u into the position just after v is a feasible solution; When the constraint conditions $F(u)+p_u \geqslant F(jp[v])+p_{jp[v]}$ are satisfied, the neighborhood solution obtained by inserting v into the position just before u is a feasible solution.

From the above table, we can see that since different researchers have different understandings of the same domain knowledge, the designed neighborhood structure may be different even if the same domain knowledge is used. For example, most researchers achieve the purpose of (1) by changing the processing sequence inside the critical path block. However, Xie et al. proposed that the same effect can be achieved by moving the operations inside the critical path block to the outside. For another example, Nowicki and Smutnicki realized (2) by only exchanging the positions of the first two or the last two operations in the critical block, while Balas and Vazacopoulos proposed that (2) could also be realized by inserting the operations in the critical block to the position just before (after) the first (last) operation. On the basis of Balas and Vazacopoulos, Zhang et al. proposed that inserting the first (last) operation on the critical block after (before) other operations can also achieve (2).

4.2 Effectiveness of domain knowledge in meta-heuristic algorithms

In the existing research, almost all the meta-heuristic algorithms use the above domain knowledge to obtain good solutions in the JSP. In order to highlight the application effect of this domain knowledge, this part sorts out the research on the upper bound of refreshing instances in the benchmark of the JSP and analyzes the

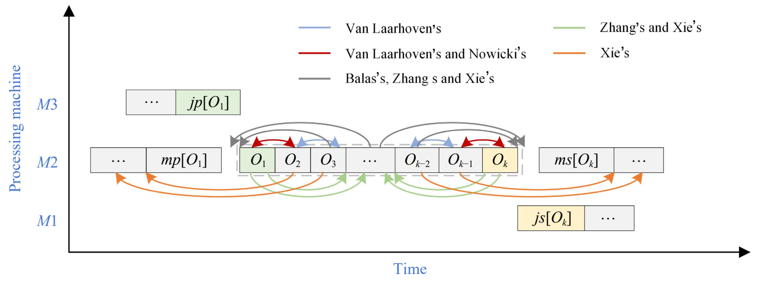

Fig. 11 Relationship between different types of solutions.

Table 1 The summary of the domain knowledge in the JSP.

	Type	Domain knowledge
Problem representation	Gantt chart	1. Marsh proposed the Harmonogram to represent the workflow network diagram[83]; 2. Gantt used this chart to discuss scheduling problems[84]; 3. Clark named this kind of chart the Gantt chart[85].
	Disjunctive graph	1. Roy and Sussmann first proposed the use of a disjunctive graph to represent the JSP[86]; 2. Balas designed an exact algorithm based on the disjunctive graph[6]; 3. White and Rogers described the extensions and limitations of the disjunctive graph[87]; 4. Bazewicz et al. pointed out that the disjunctive graph was more popular than the Gantt graph to represent the JSP[89]; 5. Nowicki et al. proposed the permutation-graph to represent the JSP[90].
Global information	Fitness landscape analysis	1. Mattfeld et al. showed that the inherent characteristics of the problem had an impact on the heuristic algorithm based on local search[70]; 2. Bierwirth et al. showed that the fitness landscape of the JSP is irregular[71]; 3. Streeter et al. showed that the fitness landscape of the JSP is related to the problem scale, and it was verified that JSP has “big valley” terrain in different instances[72].
Local information	Quality of neighborhood solutions	1. Kelley et al. proposed the critical path method for the project scheduling problem[92]; 2. Conway et al. introduced the critical path method into the JSP[93]; 3. Potts pointed out the role of the critical path in the JSP[94]; 4. Grabowski et al. proposed the concept of the critical path block based on the critical path[95]; 5. Matsuo pointed out that if the processing sequence of operations is only changed inside the critical block, the total makespan will not decrease[8]; 6. Nowicki et al. pointed out that if other operations were inserted before the first operation in the first critical block, the total completion time would not decrease, and if other operations were inserted after the last operation in the last critical block, the total completion time would not decrease[61].
	Feasibility of neighborhood solutions	1. Van Laarhoven et al. found that the neighborhood solution obtained by exchanging the processing sequence of any two adjacent operations on any critical path block must be feasible and proved its correctness[57]; 2. Balas et al. proposed a set of constraint conditions and proved that the neighborhood solution generated by inserting the operation before or after the other operation must be the feasible solution if the conditions were met[60]; 3. Gui et al. gave the necessary and sufficient conditions for feasible solutions by analyzing the necessary and sufficient conditions for infeasible solutions[96].
	Evaluation of neighborhood solutions	1. Taillard proposed an evaluation method for exchanging two adjacent operations on critical path blocks to generate neighborhood solutions[97]; 2. Balas et al. proposed the evaluation method of generating neighborhood solutions by inserting an operation before or after other operations on the critical path block[60]; 3. Nowicki et al. improved the method of Taillard[61]; 4. Gui et al. improved Balas et al. ’s method[98].
Types of solutions	Non-delayed schedule	1. It was first proposed by Jackson, and it was called availability schedule at that time[4]; 2. Nugent renamed it as non-delayed scheduling[99].
	Active schedule	1. It was first introduced by Giffler and Thompson and proved that the optimal scheduling scheme must be in the set of active schedules of the problem[38].
	Semi-active schedule	1. It was first introduced by Nugent[99].
	Full-active schedule	1. It was first proposed by Zhang et al., and proved that the optimal scheduling scheme must be in the set of full-active schedule of the problem[69].

domain knowledge applied to them.

In the JSP, there are 8 types of benchmarks, 242 different instances for all, including FT[39], LA[105], ABZ[56], ORB[106], SWV[107], YN[108], TA[109], and DMU[110]. Due to the hardness of the JSP, there are still many open problems so far in which the upper and lower bounds of the instances are not equal. The website (http://optimizizer.com/jobshop.php) has

Table 2 Details of different neighborhood structures.

Reference	Perturbation operator	Quality	Feasibility	Number
Van Laarhoven et al.[57]	1. Exchange the position of any two adjacent operations.	(1)	(4)	k−1;
Nowicki and Smutnicki[90]	1. Exchange the position of the first or last two operations; 2. except for the position change by the first two operations in the first critical block or the last two operations in the last critical block.	(1), (2), (3)	(4)	1 (the first or last critical block); 2 (the other critical path blocks);
Balas and Vazacopoulos[60]	1. Move one operation to the position just before (after) the first (last) operation.	(1), (2)	(5)	No more than 2(k−1);
Zhang et al.[103]	1. Move one operation to the position just before (after) the first (last) operation; 2. move the first (last) operation to the position just after (before) other operations.	(1), (2)	(5)	No more than 4(k−1)-2;
Xie et al.[104]	1. Move one operation (except for the first one) to the position before (not only just before) the first operation; 2. move one operation (except for the last one) to the position after (not only just after) the last operation.	(1), (2)	(5)	More than the number generated by Zhang’s;

updated the upper and lower bounds of these open instances and also lists the literature that refreshed these instances. In this part, only the literature listed on the website is collated, and the domain knowledge used in these algorithms is summarized, as shown in Table 3. Since all meta-heuristics require the encoding and decoding of the problem, they are not enumerated in the table. Only the use of global characteristics (represented by GC) and local characteristics (represented by LC) is counted. The symbol *Y* is used to indicate the use of the corresponding domain knowledge.

Among the above articles, except for Brinkkötter and Brucker’s work[111], the algorithms used for refreshing the instances are all meta-heuristic algorithms, and only one of them[117] does not use the domain knowledge mentioned above. Of course, the optimal solutions of instances[111, 117] were also found in the later papers[103, 115, 116]. This shows that the meta-heuristic algorithm combined with problem domain knowledge can solve the job shop scheduling problem efficiently.

Table 3 Details of different neighborhood structures.

Reference	GC	LC	Instance	Reference	GC	LC	Instance
Taillard[97]	-	*Y*	TA 01, 35, 51, 52, 53, 54, 56, 57, 58, 59, 60, 71, 72, 73, 74, 75, 76, 77, 78, 79	Pardalos et al.[73]	*Y*	*Y*	TA 32; DMU 06,07,08
Balas and Vazacopoulos[60]	-	*Y*	TA 03, 18	Zhang et al.[112]	-	*Y*	DMU 13, 26
Nowicki and Smutnicki[90]	-	*Y*	TA 02, 14	Beck et al.[113]	-	*Y*	TA 24, 26
Demirkol et al.[110]	-	-	DMU 32, 33, 34, 35	Gonçalves and Resende[114]	-	*Y*	DMU18,46
Brinkkötter and Brucker[111]	-	-	TA 05, 06, 07, 08, 09, 10, 12; DMU 03, 04, 05	Peng et al.[115]	*Y*	*Y*	TA47,49,50; DMU 11, 41, 50; SWV 01, 15
Nowicki and Smutnicki[61]	*Y*	*Y*	TA 22, 23, 27, 30, 45	Shylo and Shams[101]	*Y*	*Y*	TA 48; DMU 12, 16, 17,42, 43, 44, 47, 48, 49, 51, 67, 72, 74, 75; ABZ 08, 09; SWV 02, 03, 07, 09, 10, 11, 13, 14 YN 01,02,03,04
Pardalos and Shylo[68]	-	*Y*	TA 11, 15, 19, 20; DMU 10	Constanino and Segura[116]	-	*Y*	TA 34, 40, 42, 44; DMU 19, 20, 45, 52, 53, 54, 55, 58, 59, 60, 61, 62, 63, 64, 65, 66, 68, 69, 70, 71, 73, 76, 77, 78, 79, 80; SWV 06, 12
Zhang et al.[103]	-	*Y*	TA 28, 33, 37	Xie et al.[104]	-	*Y*	DUM 56, 57

4.3 Relationship among optimization problems, domain knowledge and optimization methods

From the above analysis, it is obvious that the meta-heuristic algorithms with the domain knowledge in the JSP have an effective improvement for the meta-heuristic algorithms. In fact, this situation does not only exist in the JSP. When the domain knowledge of other shop scheduling problems, or even other types of optimization problems, is used in the design of meta-heuristic algorithms, the optimization effect of the algorithm can be effectively improved. Similarly, even if the optimization algorithm is not a meta-heuristic algorithm, the corresponding domain knowledge of the problem can also improve the optimization effect of the algorithm. Therefore, it is necessary to discuss the relationship among optimization problems, domain knowledge and optimization algorithms, and a triangular ring graph is used to represent the relationship between them, as shown in Fig. 12.

In the figure, there are two different correspondences between these three: 1) The clockwise loop in the inner circle represents the thinking process before solving the problem. Specifically, the optimization method should be selected according to the characteristics of the optimization problem (e.g., the JSP is an NP-hard problem, and the meta-heuristic algorithm is a good choice). Secondly, it is necessary to map the domain knowledge to be used according to the type of optimization method (different types of algorithms require different domain knowledge, for example, mixed-integer programming model uses more convex optimization and branch and bound domain knowledge, while the meta-heuristic algorithm uses domain knowledge such as globe information and local information of the search space). Finally, according to the required domain knowledge, the acquisition method in the problem is analyzed (for example, the domain knowledge of the fitness landscape of the solution space needs to use the means of statistics, etc.); 2) The counterclockwise loop of the outer circle represents the application process of the algorithm design. Firstly, it is necessary to obtain the required domain knowledge in the optimization problem (through theoretical analysis, machine learning, etc.). Secondly, specific domain knowledge should be used in the design of optimization methods (such as global characteristics or local characteristics for the design of algorithm optimization operators). Finally, the designed optimization algorithm is used to solve the optimization problem.

In fact, the pairwise correspondence between these three is also the direction to be studied in the optimization problem. The problems that how to select the optimization method through the optimization problem, how to know the type of domain knowledge needed by the type of optimization method, and how to analyze the optimization problem through the type of domain knowledge, similarly, the problems that how to mine domain knowledge from optimization problems, how to use domain knowledge to design optimization algorithms, and how to use optimization algorithms to solve problems should be further studied. Although this paper does not give the specific research methods for these problems, through the analysis of the relationship between the above problems, the research directions are more clear for the study of optimization problems, and we believe that it could play a certain role in promoting the research of optimization problems. It is hoped that the discussion can provide some reference for the solving of optimization problems or the design of optimization methods.

5 Discussion and Research Direction

In Sections 3 and 4, we summarized the domain knowledge in the JSP for the design of meta-heuristic algorithms. It is easy to know that domain knowledge plays an important role in meta-heuristic algorithms. In this section, the shortcomings of the existing research and the research directions are proposed for the JSP.

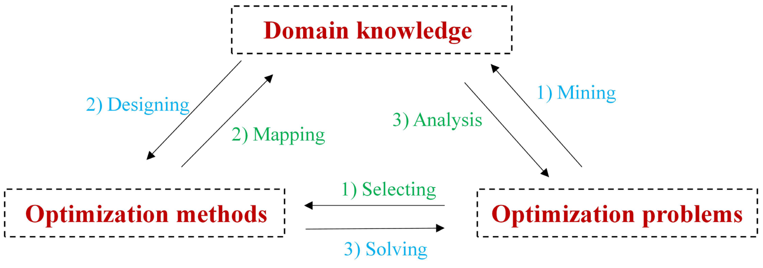

Fig. 12 Relationship between different types of solutions.

5.1 Paying more attention to the fitness landscape analysis for the JSP

In continuous optimization problems, global characteristics can be used to determine the difficulty of the optimization problem and can also greatly improve the optimization performance of the algorithm. It shows the importance of global characteristics for solving optimization problems. In the JSP, fitness landscape analysis is generally used to describe the global characteristics of this problem. However, the study of fitness landscape analysis in the JSP is insufficient, so more attention should be paid to it in future research. Research from three aspects can be considered: 1) To construct an appropriate metric space to describe the geometric relationship between different solutions in the solution space; 2) Study the sampling methods (such as random sampling or uniform sampling, etc.) for the JSP to obtain data for analysis; 3) Choose suitable statistical parameters to describe the characteristics of the fitness landscape.

5.2 Domain knowledge of the relationship between different objectives

In actual production, there may be many different optimization objectives. The existing research usually directly uses multi-objective optimization algorithm to solve the problem by showing the conflict relationship between different objectives through examples. However, the relationship between different optimization objectives is complex, and there may be different combinations of conflict and non-conflict among them. If the multi-objective optimization algorithm is simply used, it may lead to low efficiency and poor results. Therefore, the domain knowledge of the relationship between different objectives should be analyzed.

5.3 Extending the domain knowledge to other problems

Domain knowledge is very important to the optimization algorithm design, which can improve the optimization performance of the algorithm. However, it is very difficult to extract the desired domain knowledge from the problem. Therefore, transferring the known domain knowledge to other similar problems by analyzing the similarities and differences between these problems is a promising research direction. For example, the domain knowledge related to the feasibility of neighborhood solutions in the classical JSP can also be applied to other JSP, such as flexible JSP, JSP considering transportation time, etc. Of course, some domain knowledge needs to be adjusted before it can be applied to other problems, such as the evaluation methods for neighborhood solutions.

5.4 Using data-driven approaches to obtain domain knowledge of the problem

As mentioned above, domain knowledge mining is a difficult task, especially for complex combinatorial optimization problems. In this case, we can utilize some data-driven approaches such as deep learning, reinforcement learning, etc. By using these methods, we can find some rules in a large number of data, and can apply these rules to the solution of the problem. For example, some rules or phenomena have been found in the optimization process, but it may be difficult to draw clear conclusions through theoretical analysis. At this time, a large number of samples can be analyzed using machine learning methods, and finally a conclusion can be given to improve the optimization performance of the meta-heuristic algorithm.

5.5 Selecting a meta-heuristic algorithm framework according to the domain knowledge

In addition to the strategies designed by domain knowledge, there are also many optimization strategies or frameworks unrelated to specific problems in meta-heuristic algorithms, such as Metropolis criterion in simulated annealing algorithm, tabu criterion in tabu search algorithm, and optimization frameworks (crossover, mutation, selection) in genetic algorithm, which can also directly affect the optimization performance of the algorithm. These optimization strategies or frameworks have certain generalization properties, which can be applied to different types of optimization problems. However, how to choose these optimization strategies and frameworks for a specific problem is unknown. In fact, although these optimization strategies and frameworks are not designed by the domain knowledge of the problem, they must be compatible with the deeper domain knowledge of the problem if they have excellent performance on a specific problem. Therefore, how to use the domain knowledge of the problem to select optimization strategies and frameworks of meta-heuristics is a promising research direction.

In addition to the research directions mentioned above, there are many directions or methods for domain knowledge in the JSP that deserve to be studied. In fact, the process of algorithm design is to constantly discover and use the domain knowledge of a specific problem, and it is this process that promotes the development of the field of shop scheduling, even the whole field of operations research.

6 Conclusion

The main work of this paper is to survey the domain knowledge used in the design of meta-heuristic algorithms for the JSP. Firstly, the importance of domain knowledge for the design of optimization algorithms is highlighted. After that, this paper reviews the development of optimization algorithms for the JSP. It shows that meta-heuristic algorithms are the mainstream methods for this problem and points out that different types of optimization algorithms need different domain knowledge. Then, the domain knowledge in the JSP is summarized, classified and analyzed according to the optimization steps of meta-heuristics, and the application of these domains knowledge in the design of meta-heuristics is presented. By analyzing the literatures which refreshed datasets, this paper illustrates that domain knowledge is the guarantee to ensure the excellent performance of meta-heuristic algorithms. At the same time, the relationship between optimization problems, optimization methods and domain knowledge is discussed. Finally, this paper lists the shortcomings of the existing research, and puts forward the need to strengthen the research on fitness landscape and objective analysis, expand the application of domain knowledge, use new technology to mine domain knowledge, and analyze the role of meta-heuristic algorithms in optimization. Although the main research object of this paper is JSP and meta-heuristic algorithms, the discussion in this paper can also provide reference for other optimization problems or optimization algorithm design.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Nos. U21B2029 and 51825502).

References

[1] D. H. Wolpert and W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.

[2] Y. An, X. Chen, K. Gao, Y. Li, and L. Zhang, Multiobjective flexible job-shop rescheduling with new job insertion and machine preventive maintenance, *IEEE Trans. Cybern.*, vol. 53, no. 5, pp. 3101–3113, 2023.

[3] Y. Fu, Y. Hou, Z. Wang, X. Wu, K. Gao, and L. Wang, Distributed scheduling problems in intelligent manufacturing systems, *Tsing Science and Technology*, vol. 26, no. 5, pp. 625–645, 2021.

[4] J. R. Jackson, Notes on some sheduling problems, research report no. 35, Research Report, Management Sciences Research Project, UCLA, 1954.

[5] Z. Hu and D. Li, Improved heuristic job scheduling method to enhance throughput for big data analytics, *Tsing Science and Technology*, vol. 27, no. 2, pp. 344–357, 2022.

[6] E. Balas, Machine sequencing via disjunctive graphs: An implicit enumeration algorithm, *Oper. Res.*, vol. 17, no. 6, pp. 941–957, 1969.

[7] P. Brucker, An efficient algorithm for the job-shop problem with two jobs, *Computing*, vol. 40, no. 4, pp. 353–359, 1988.

[8] H. Matsuo, C. Juck SUH, and R. S. Sullivan, A controlled search simulated annealing method for the single machine weighted tardiness problem, *Ann. Oper. Res.*, vol. 21, no. 1, pp. 85–108, 1989.

[9] S. M. Alexander, An expert system for the selection of scheduling rules in a job shop, *Comput. Ind. Eng.*, vol. 12, no. 3, pp. 167–171, 1987.

[10] Z. Liu, Y. Wang, X. Liang, Y. Ma, Y. Feng, G. Cheng, and Z. Liu, A graph neural networks-based deep Q-learning approach for job shop scheduling problems in traffic management, *Inf. Sci.*, vol. 607, pp. 1211–1223, 2022.

[11] L. Wang, Z. Pan, and J. Wang, A review of reinforcement learning based intelligent optimization for manufacturing scheduling, *Complex System Modeling and Simulation*, vol. 1, no. 4, pp. 257–270, 2021.

[12] Z. Chen, L. Zhang, X. Wang, and P. Gu, Optimal design of flexible job shop scheduling under resource preemption based on deep reinforcement learning, *Complex System Modeling and Simulation*, vol. 2, no. 2, pp. 174–185, 2022.

[13] B. Xi and D. Lei, Q-learning-based teaching-learning optimization for distributed two-stage hybrid flow shop scheduling with fuzzy processing time, *Complex Syst. Model. Simul.*, vol. 2, no. 2, pp. 113–129, 2022.

[14] M. Gendreau and J. Y. Potvin, *Handbook of metaheuristics*, Vol.2 New York: Springer, 2010, p. 9.

[15] P. P. Bonissone, R. Subbu, N. Eklund, and T. R. Kiehl, Evolutionary algorithms+ domain knowledge= real-world evolutionary computation, *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 256–280, 2006.

[16] S. B. Akers Jr and J. Friedman, A non-numerical approach to production scheduling problems, *J. Oper. Res. Soc.*, vol. 3, no. 4, pp. 429–442, 1955.

[17] R. L. Sisson, Methods of sequencing in job shops—A review, *Oper. Res.*, vol. 7, no. 1, pp. 10–29, 1959.

[18] H. E. Nouri, O. B. Driss, and K. Ghédira, A classification schema for the job shop scheduling problem with transportation resources: state-of-the-art review, in Artificial Intelligence Perspectives in Intelligent Systems: Proceedings of the 5th Computer Science On-line Conference 2016 (CSOC2016), Springer International Publishing, vol. 1, pp. 1−11, 2016.

[19] Q. Li, J. Li, Q. Zhang, P. Duan, and T. Meng, An improved whale optimisation algorithm for distributed assembly flow shop with crane transportation, *Int. J. Autom. Control*, vol. 15, no. 6, pp. 710–743, 2021.

[20] S. C. Kim and P. M. Bobrowski, Impact of sequence-dependent setup time on job shop scheduling performance, *Int. J. Prod. Res.*, vol. 32, no. 7, pp. 1503–1520, 1994.

[21] X. Han, Y. Han, Q. Chen, J. Li, H. Sang, Y. Liu, Q. Pan, and Y. Nojima, Distributed flow shop scheduling with sequence-dependent setup times using an improved iterated greedy algorithm, *Complex System Modeling and Simulation*, vol. 1, no. 3, pp. 198–217, 2021.

[22] K. Gao, F. Yang, M. Zhou, Q. Pan, and P. N. Suganthan, Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm, *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, 2019.

[23] X. Wu, X. Xiao, and Q. Cui, Multi-objective flexible flow shop batch scheduling problem with renewable energy, *Int. J. Autom. Control*, vol. 14,nos.5-6, pp. 519–553, 2020.

[24] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, and Q. K. Pan, A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion, *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7652–7663, 2015.

[25] Q. Liu, C. Wang, X. Li, and L. Gao, Mathematical modeling and a multiswarm collaborative optimization algorithm for fuzzy integrated process planning and scheduling problem, *Tsing Science and Technology*, vol. 29, no. 2, pp. 285–304, 2024.

[26] L. Sun, T. Lu, Z. Li, Y. Li, Y. Yu, and J. Liu, Research on steelmaking-continuous casting production scheduling problem with uncertain processing time based on Lagrangian relaxation framework, *Int. J. Autom. Control*, vol. 16, no. 1, pp. 87–107, 2022.

[27] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, T. X. Cai, and C. S. Chong, Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling, *Inf. Sci*, vol. 289, pp. 76–90, 2014.

[28] W. Zhang, W. Hou, C. Li, W. Yang, and M. Gen, Multidirection update-based multiobjective particle swarm optimization for mixed no-idle flow-shop scheduling problem, *Complex System Modeling and Simulation*, vol. 1, no. 3, pp. 176–197, 2021.

[29] E. Jiang, L. Wang, and J. Wang, Decomposition-based multi-objective optimization for energy-aware distributed hybrid flow shop scheduling with multiprocessor tasks, *Tsing Science and Technology*, vol. 26, no. 5, pp. 646–663, 2021.

[30] K. Z. Gao, P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, and A. Sadollah, Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion, *Knowl. Based. Syst.*, vol. 109, pp. 1–16, 2016.

[31] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, C. S. Chong, and T. X. Cai, An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time, *Expert Syst. Appl.*, vol. 65, pp. 52–67, 2016.

[32] H. Xiong, S. Shi, D. Ren, and J. Hu, A survey of job shop scheduling problem: The types and models, *Comput. Oper. Res.*, vol. 142, p. 105731, 2022.

[33] J. Błażewicz, W. Domschke, and E. Pesch, The job shop scheduling problem: Conventional and new solution techniques, *Eur. J. Oper. Res.*, vol. 93, no. 1, pp. 1–33, 1996.

[34] M. Dhiflaoui, H. E. Nouri, and O. B. Driss, Dual-resource constraints in classical and flexible job shop problems: A state-of-the-art review, *Procedia Comput. Sci.*, vol. 126, pp. 1507–1515, 2018.

[35] R. G. Parker, *Deterministic Scheduling Theory*, CRC Press, 1996.

[36] S. M. Johnson, An extension of johnson's results on job lot scheduling, *Nav. Res. Logist.*, vol. 3, pp. 201–203, 1956.

[37] W. E. Smith, Various optimizers for single-stage production, *Nav. Res. Logist.*, vol. 3, pp. 59–66, 1956.

[38] B. Giffler and G. L. Thompson, Algorithms for solving production-scheduling problems, *Oper. Res.*, vol. 8, no. 4, pp. 487–503, 1960.

[39] H. Fisher and G. L. Thompson, Probabilistic learning combinations of local job-shop scheduling rules, *In*: *Muth JF and Thompson GL* (*eds*). *Industrial Scheduling, Prentice-Hall*: *Englewood Cliffs, NJ*, pp. 225– 251, 1963.

[40] W. B. Crowston, F. Glover, and J. D. Trawick, Probabilistic and parametric learning combinations of local job shop scheduling rules, Research Report Carnegie Inst of Tech Pittsburgh Pa Graduate School of Industrial Administration, 1963.

[41] B. Jeremiah, A. Lalchandani, and L. Schrage, Heuristics Rules Toward Optimal Scheduling, Department of Industrial Engineering, Research Report, Cornell University, New York, USA, 1964.

[42] G. H. Brooks, An algorithm for finding optimal or near optimal solutions to the production scheduling problem, *J. Ind. Eng.*, vol. 16, no. 1, pp. 34–40, 1969.

[43] M. R. Garey and D. S. Johnson, C*omputers and Intertract-ability*: *A Guide to the Theory of NP-Completeness*, Freeman, San Francisco, CA, 1979.

[44] E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, Recent developments in deterministic sequencing and scheduling: A survey, In Deterministic and Stochastic Scheduling: Proceedings of the NATO Advanced Study and Research Institute on Theoretical Approaches to Scheduling Problems held in Durham, England, pp. 35−73, 1981.

[45] M. Hefetz and I. Adiri, An efficient optimal algorithm for the two-machines unit-time job shop schedule-length problem, *Math. Oper. Res.*, vol. 7, no. 3, pp. 354–360,

1982.
[46] Y. N. Sotskov, Optimal scheduling two jobs with regular criterion, *Design Processes Automating*, pp. 86–95, 1985.
[47] P. Brucker, An efficient algorithm for the job-shop problem with two jobs, *Computing*, vol. 40, no. 4, pp. 353–359, 1988.
[48] N. M. Sadeh, Look-ahead techniques for micro-opportunistic job shop scheduling, Research Report, Carnegie Mellon University, 1991.
[49] M. Charalambous and K. S. Hindi, A review of artificial intelligence-based job-shop scheduling systems, *Information and Decision Technologies*, vol. 17, no. 3, pp. 189–202, 1991.
[50] D. N. Zhou, V. Cherkassky, T. R. Baldwin, and D. W. Hong, Scaling neural network for job-shop scheduling, In 1990 IJCNN International Joint Conference on Neural Networks, San Diego, CA, USA, 1990, vol.3, pp. 889–894.
[51] D. N. Zhou, V. Cherkassky, T. R. Baldwin and D. E. Olson, A neural network approach to job-shop scheduling, *IEEE Trans. Neural Netw.*, vol. 2, no. 1, pp. 175–179, 1991.
[52] T. A. J. Nicholson, A sequential method for discrete optimization problems and its application to the assignment, travelling salesman, and three machine scheduling problems, *IMA J. Appl. Math.*, vol. 3, no. 4, pp. 362–375, 1967.
[53] S. Kirkpatrick, C. D. Gelatt Jr, and M. P.Vecchi, Optimization by simulated annealing, *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[54] F. Glover, Tabu search—part I, *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989.
[55] F. Glover, Tabu search—part Ⅱ, *ORSA J. Comput.*, vol. 2, no. 1, pp. 4–32, 1990.
[56] J. Adams, E. Balas, and D. Zawack, The shifting bottleneck procedure for job shop scheduling, *Manage. Sci.*, vol. 34, no. 3, pp. 391–401, 1988.
[57] N. J. Van Laarhoven, E. H. Aarts, and J. K. Lenstra, Job shop scheduling by simulated annealing, *Oper. Res.*, vol. 40, no. 1, pp. 113–125, 1992.
[58] E. H. L. Aarts, P. J. M. Van Laarhooven, and N. L. J. Ulder, Local search based algorithms for job-shop scheduling, Working Paper, Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands, 1991.
[59] E. Taillard, Parallel taboo search technique for the job-shop scheduling problem, Internal Research Report ORWP89/11, Department de Mathematiques (DMA), Ecole Polytechnique Federale de Lausanne, 1015 Lausanne Switzerland, 1989.
[60] E. Balas and A. Vazacopoulos, Guided local search with shifting bottleneck for job shop scheduling, *Manage. Sci.*, vol. 44, no. 2, pp. 262–275, 1998.
[61] E. Nowicki and C. Smutnicki, An advanced tabu search algorithm for the job shop problem, *J. Scheduling*, vol. 8, no. 2, pp. 145–159, 2005.
[62] M. L. Fisher and A. H. Rinnooy Kan, The design, analysis and implementation of heuristics, *Manage. Sci.*, vol. 34, no. 3, pp. 263–265, 1988.
[63] F. Della Croce, R. Tadei, and G. Volta, A genetic algorithm for the job shop problem, *Comput. Oper. Res.*, vol. 22, no. 1, pp. 15–24, 1995.
[64] H. L. Fang, P. Ross, and D. Corne, A promising genetic algorithm approach to job-shop scheduling, rescheduling, and open-shop scheduling problems, Berlin, Heidelberg: University of Edinburgh, Department of Artificial Intelligence, pp. 375–382, 1993.
[65] A. Colorni, M. Dorigo, V. Maniezzo, and M. Trubian, Ant system for job-shop scheduling, *JORBEL-Belgian Journal of Operations Research, Statistics, and Computer Science*, vol. 34, no. 1, pp. 39–53, 1994.
[66] Z. Lian, B. Jiao, and X. Gu, A similar particle swarm optimization algorithm for job-shop scheduling to minimize makespan, *Appl. Math. Comput.*, vol. 183, no. 2, pp. 1008–1017, 2006.
[67] O. Niu, B. Jiao, and X. Gu, Particle swarm optimization combined with genetic operators for job shop scheduling problem with fuzzy processing time, *Appl. Math. Comput.*, vol. 205, no. 1, pp. 148–158, 2008.
[68] P. M. Pardalos and O. V. Shylo, An algorithm for the job shop scheduling problem based on global equilibrium search techniques, *Comput. Manag. Sci.*, vol. 3, no. 4, pp. 331–348, 2006.
[69] G. Zhang, X. Shao, P. Li, and L. Gao, An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem, *Comput. Ind. Eng.*, vol. 56, no. 4, pp. 1309–1318, 2009.
[70] D. C. Mattfeld, C. Bierwirth, and H. Kopfer, A search space analysis of the job shop scheduling problem, *Ann. Oper. Res.*, vol. 86, pp. 441–453, 1999.
[71] C. Bierwirth, D. C. Mattfeld, and J. P. Watson, Landscape regularity and random walks for the job-shop scheduling problem, In European Conference on Evolutionary Computation in Combinatorial Optimization (pp. 21-30). Springer, Berlin, Heidelberg, 2004.
[72] M. J. Streeter and S. F. Smith, How the landscape of random job shop scheduling instances depends on the ratio of jobs to machines, *J. Artif. Intell. Res.*, vol. 26, pp. 247–287, 2006.
[73] P. M. Pardalos, O. V. Shylo, and A. Vazacopoulos, Solving job shop scheduling problems utilizing the properties of backbone and “big valley”, *Comput. Optim. Appl.*, vol. 47, no. 1, pp. 61–76, 2010.
[74] J. Li, X. Gu, Y. Zhang, and X. Zhou, Distributed flexible job-shop scheduling problem based on hybrid chemical reaction optimization algorithm, *Complex System Modeling and Simulation*, vol. 2, no. 2, pp. 156–173, 2022.
[75] S. H. Niu, S. K. Ong, and A. Y. C. Nee, An improved intelligent water drops algorithm for achieving optimal job-shop scheduling solutions, *Int. J. Prod. Res.*, vol. 50, no. 15, pp. 4192–4205, 2012.
[76] B. Wu, J. Cheng, and M. Dong, Hybrid fruit fly optimisation algorithm for field service scheduling problem, *Int. J. Autom. Control*, vol. 14, no. 5-6, pp. 554–570, 2020.

[77] S. Wang, C. Liu, D. Pei, and J. Wang, A novel hybrid election campaign optimisation algorithm for multi-objective flexible job-shop scheduling problem, *Int. J. Struct. Integr.*, vol. 7, no. 3, pp. 160–170, 2013.

[78] S. Wang, G. Liu, and S. Gao, A hybrid discrete imperialist competition algorithm for fuzzy job-shop scheduling problems, *IEEE Access*, vol. 4, pp. 9320–9331, 2017.

[79] C. Aranha, C. L. Camacho Villaló n, F. Campelo, M. Dorigo, R. Ruiz, M. Sevaux, K. Sörensen, and T. Stü tzle, Metaphor-based metaheuristics, a call for action: the elephant in the room, *Swarm Intell.*, vol. 16, no. 1, pp. 1–6, 2022.

[80] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem, *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 621–639, 2012.

[81] S. Mirshekarian and D. N. Šormaz,Correlation of job-shop scheduling problem features with scheduling efficiency, *Expert Syst. Appl.*, vol. 62, pp. 131–147, 2016.

[82] Z. C. Li, B. Qian, R. Hu, L. L. Chang, and J. B. Yang, An elitist nondominated sorting hybrid algorithm for multi-objective flexible job-shop scheduling problem with sequence-dependent setups, *Knowl. Based Syst.*, vol. 173, pp. 83–112, 2019.

[83] E. R. Marsh, The harmonogram: An overlooked method of scheduling work, *Proj. Manage. Q.*, vol. 7, no. 1, pp. 21–25, 1976.

[84] H. L. Gantt, *Work, Wages and Profits*, Engineering Magazine Co., New York, 1916.

[85] W. Clark, *The Gantt chart*: *A working tool of management*, Ronald Press Company, 1922.

[86] B. Roy and B. Sussman, *Les problem`es d' ordonnancement avec contraintes disjonctives* (*in French*). Note DS No. 9 bis, SEMA, Montrouge, 1964.

[87] K. P. White and R. V. Rogers, Job-shop scheduling: Limits of the binary disjunctive formulation, *Int. J. Prod. Res.*, vol. 28, no. 12, pp. 2187–2200, 1990.

[88] L. Gui, L. Fu, X. Li, W. Zhou, L. Gao, Z. Xiang, and W. Zhu, Optimisation framework and method for solving the serial dual-shop collaborative scheduling problem, *Int. J. Prod. Res.*, vol. 61, pp. 4341–4357, 2022.

[89] J. Bazewicz, W. Domschke, and E. Pesch, The job shop scheduling problem: Conventional and new solution techniques, *Eur. J. Oper. Res.*, vol. 93, no. 1, pp. 1–33, 1996.

[90] E. Nowicki and C. Smutnicki, A fast taboo search algorithm for the job shop problem, *Manage. Sci.*, vol. 42, no. 6, pp. 797–813, 1996.

[91] S. Wright, The roles of mutation, inbreeding, crossbreeding, and selection in evolution, *Proceedings of the Sixth international Congress of Genetics*, vol. 1, pp. 356–366, 1932.

[92] J. E. Kelley Jr and M. R. Walker, Critical-path planning and scheduling. In Papers presented at the December, Eastern Joint IRE-AIEE-ACM Computer Conference pp. 160−173, 1959.

[93] P. W. Conway, W. L. Maxwell and L. W. Miller, *Theory of Scheduling, Addison-Wesley*: *Reading*, MA, 1967.

[94] C. N. Potts, Analysis of a heuristic for one machine sequencing with release dates and delivery times, *Oper. Res.*, vol. 28, no. 6, pp. 1436–1441, 1980.

[95] J. Grabowski, E. Nowicki, and C. Smutnicki, Block algorithm for scheduling of operations in job-shop system (in Polish), Przeglad Statystyczny, vol. 35, pp. 67–80, 1988.

[96] L. Gui, X. Li, L. Gao, and C. Wang, Necessary and sufficient conditions for feasible neighbourhood solutions in the local search of the job-shop scheduling problem, *Chin. J. Mech. Eng.*, vol. 36, no. 1, pp. 1–16, 2023.

[97] E. Taillard, Parallel taboo search techniques for the job shop scheduling problem, *ORSA J. Comput.*, vol. 6, no. 2, pp. 108–117, 1994.

[98] L. Gui, X. Li, L. Gao, and J. Xie, An approximate evaluation method for neighbourhood solutions in job shop scheduling problem, *IET CIM*, vol. 4, no. 3, pp. 157–165, 2022.

[99] C. E. Nugent, On Sampling Approaches to the Solution of the n-by-m Static Sequencing Problem, Ph.D. dissertation, Cornell University, USA, 1964.

[100] C. Zhang, Y. Rao, and P. Li, An effective hybrid genetic algorithm for the job shop scheduling problem, *Int. J. Adv. Manuf. Technol.*, vol. 39, no. 9, pp. 965–974, 2008.

[101] O. V. Shylo and H. Shams, Boosting binary optimization via binary classification: A case study of job shop scheduling. arXiv preprint arXiv: 1808.10813, 2018. M.

[102] M. Nasiri and F. Kianfar, A GES/TS algorithm for the job shop scheduling, *Comput. Ind. Eng.*, vol. 62, no. 4, pp. 946–952, 2012.

[103] C. Zhang, P. Li, Z. Guan, and Y. Rao, A tabu search algorithm with a new neighborhood structure for the job shop scheduling problem, *Comput. Oper. Res.*, vol. 34, no. 11, pp. 3229–3242, 2007.

[104] J. Xie, X. Li, L. Gao, and L. Gui, A hybrid algorithm with a new neighborhood structure for job shop scheduling problems, *Comput. Ind. Eng.*, vol. 169, pp. 108205, 2022.

[105] Lawrence, Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques (Relatório técnico), dissertation, Carnegie Mellon University, Pittsburgh, USA, 1984.

[106] D. Applegate and W. Cook, A computational study of the job-shop scheduling problem, *ORSA J. Comput.*, vol. 3, no. 2, pp. 149–156, 1991.

[107] R. H. Storer, S. D. Wu, and R. Vaccari, New search spaces for sequencing problems with application to job shop scheduling, *Manage. Sci.*, vol. 38, no. 10, pp. 1495–1509, 1992.

[108] T. Yamada and R. Nakano, A genetic algorithm applicable to large-scale job-shop problems, In PPSN vol. 2, pp. 281−290, 1992.

[109] E. Taillard, Benchmarks for basic scheduling problems, *Eur. J. Oper. Res.*, vol. 64, no. 2, pp. 278–285, 1993.

[110] E. Demirkol, S. Mehta, and R. Uzsoy, Benchmarks for shop scheduling problems, *Eur. J. Oper. Res.*, vol. 109, no. 1, pp. 137–141, 1998.

[111] W. Brinkkötter and P. Brucker, Solving open benchmark instances for the job﹣shop problem by parallel head–tail adjustments, *J. Scheduling*, vol. 4, no. 1, pp. 53–64, 2001.
[112] C. Zhang, P. Li, Y. Rao, and Z. Guan, A very fast TS/SA algorithm for the job shop scheduling problem, *Comput. Oper. Res.*, vol. 35, no. 1, pp. 282–294, 2008.
[113] J. C. Beck, T. K. Feng, and J. P. Watson, Combining constraint programming and local search for job-shop scheduling, *INFORMS J. Comput.*, vol. 23, no. 1, pp. 1–14, 2011.
[114] J. F. Gonçalves and M. G. Resende, A biased random-key genetic algorithm for job-shop scheduling, AT&T Labs Research Technical Report, vol. 46, pp. 253–271, 2011.
[115] B. Peng, Z. Lü, and T. C. E. Cheng, A tabu search/path relinking algorithm to solve the job shop scheduling problem, *Comput. Oper. Res.*, vol. 53, pp. 154–164, 2015.
[116] O. H. Constanino and C. Segura, A parallel memetic algorithm with explicit management of diversity for the job shop scheduling problem, *Appl. Intell.*, vol. 52, no. 1, pp. 141–153, 2022.
[117] E. Demirkol, S. Mehta, and R. Uzsoy, A computational study of shifting bottleneck procedures for shop scheduling problems, *J. Heuristics*, vol. 3, no. 2, pp. 111–137, 1997.

Xinyu Li received the PhD degree from Huazhong University of Science and Technology, China in 2009. He is a professor of Huazhong University of Science and Technology. He has published more than 90 peer reviewed papers. He serves as an associate editor of *IET Collaborative Intelligent Manufacturing*, and the editorial board member of *Sensors*. His research interests include intelligent algorithms, big data, and machine learning, among others.

Qingfu Zhang received the PhD degree from Xidian University, China in 1994. He is a Chair Professor of City University of Hong Kong. He serves as an associate editor of *IEEE Transactions on Evolutionary Computation*, *IEEE Transactions on Cybernetics* and *International Journal of Swarm Intelligence Research*. His research interests include evolutionary computation, multiobjective optimization, and machine learning.

Lin Gui received the BS degree in mechanical engineering from Shandong University, Jinan, China, 2018. He is currently pursuing the PhD degree in industrial engineering with the Huazhong University of Science and Technology, Wuhan, China. His research interests are shop scheduling problems and optimisation algorithms.

Liang Gao received the PhD degree from Huazhong University of Science and Technology, China in 2002. He is a professor of Huazhong University of Science and Technology. He serves as editor-in-chief of *IET Collaborative Intelligent Manufacturing*, and associate editor of *Swarm and Evolutionary Computation* and *Journal of Industrial and Production Engineering*. His research interests are the research on the application of intelligent optimization and machine learning methods in design and manufacturing.