Solving Flexible Job-shop Scheduling Problem Based on Improved Genetic Algorithm

Xinhui Zhou

Liaoning Technical University, Huludao, China

2406202220@qq.com

Abstract. In order to optimize the allocation of production factors and improve resource utilization, this paper constructs a scheduling model aiming at minimizing the maximum completion time, and proposes an improved genetic algorithm with self-adjusting search domain to solve flexible job-shop scheduling problems. The algorithm adopts a double-layer chromosome coding scheme based on sequence arrangement and machine selection. A new population initialization method was designed, and adaptive parameters were introduced to optimize the genetic operation flow in the crossover and mutation stages. The introduction of new populations in the later stages of evolution increases the diversity of chromosomes, and self-adjusts the algorithm search domain to help the population jump out of the local optimal to find a better global solution. The experimental results show that the proposed improved genetic algorithm performs well in terms of optimization accuracy and convergence ability.

Keywords: Job shop scheduling, Flexibility, Genetic algorithm, Self-tuning the search domain.

1 Introduction

1.1 A Subsection Sample

The flexible job shop scheduling problem is an extension of the job shop scheduling problem model, which fits the real production scene more closely. FJSP gives more freedom in machine selection, can deal with the complex scheduling needs involving many machines, multiple processes and tasks, so as to better adapt to the changing and complex production environment, and therefore has been concerned by the academia and the industry.

Through the in-depth exploration of domestic and foreign scholars, the intelligent optimization algorithm shows stronger adaptability and flexibility than the traditional method when dealing with FJSP. Especially in the face of large and complex FJSP, the intelligent optimization algorithm shows excellent performance, which can quickly lock the global optimal solution in the vast solution space and optimize the efficiency of resource use. Long et al.[1]enhanced the global search capability and convergence accuracy of the artificial bee colony algorithm by combining the reinforcement learning

S. M. Ghadiri et al. (eds.), *Proceedings of the 2025 4th International Conference on Engineering Management and Information Science (EMIS 2025)*, Advances in Computer Science Research 121,
https://doi.org/10.2991/978-94-6463-736-6_26

algorithm. Xin et al.[2]proposed an improved particle swarm optimization algorithm, which significantly improved search accuracy and convergence speed through adaptive parameter adjustment and chaotic local optimizer. Wang et al.[3]proposed an innovative hybrid algorithm, which combines two strategies of genetic algorithm and taboo search to solve the problem. While achieving global optimization, it can also make fine local adjustments. Tang[4] proposed a knowledge-based ant colony optimization solution for FJSP. This model learns from ant colony optimization and applies the acquired knowledge to the current search process. Wang[5] proposed a hybrid particle swarm and taboo search to solve FJSPS containing multiple conflicts and incommensurable factors, and the algorithm showed higher efficiency and accuracy when dealing with complex and multi-conflict FJSPS.

Genetic Algorithm (GA), as a representative of intelligent optimization algorithm, has shown excellent performance in solving FJSP. Based on the population of candidate solutions, GA continuously generates new solutions through genetic operations such as crossover and mutation. In addition, GA also has powerful parallel processing capabilities, which can simultaneously optimize multiple objectives, such as machine configuration, time allocation, etc., so as to achieve efficient use of resources and improve production efficiency[6].

This research takes genetic algorithm as the core and aims to minimize the maximum completion time as the main optimization goal. An improved genetic algorithm (SAGA) for Self-adjusting Search Domain GA is proposed. Through in-depth exploration of the theory and method of solving FJSP, it is expected to achieve efficiency improvement in the actual production environment, promote the application of intelligent optimization algorithm in the field of production scheduling, and provide strong support for resource allocation and efficiency improvement in the actual production process.

2 Problem Description

FJSP is defined to determine the optimal process sequence and the task of executing the machine in each process in a workshop with diverse machine clusters and different types of work pieces. The specific mathematical model is described as: a set of mutually independent number of n work pieces (J1, J2, ..., Jn), these work pieces need to be in m different machines (M1, M2, ..., Mm) on the processing. For each work piece Ji, it consists of u processing processes. Each processing step takes a certain amount of time to complete, the time is greater than 0, and each step can be completed on at least one machine. At any given moment, a machine can only handle one processing step and cannot handle multiple tasks simultaneously. In addition, once a processing step is started, it must be completed until the processing is complete, and there can be no interruption in the middle. At the same time, the transport time of the work piece between different machines is not taken into account. The core goal of this scheduling model is to minimize the completion time of all processing steps, which can be expressed by the following objective function:

$$f = \min\left(\sum_{i=1}^{n}\sum_{j=1}^{n_i} S_{ijk} + t_{ijk}\right)$$

Where Sijk represents the start time of process Oij on machine k. tijk represents the processing time required for operation Oij on machine k.

3 Algorithm Design

3.1 Chromosome Representation

The key of FJSP is to deal with two core sub-problems, namely Operation Sequence (OS) and Machine Selection (MS), which can be better solved by using double-layer chromosome representation. Using table 1 as a reference, in the OS section, the first number "2" refers to the first process of the second job, which is labeled O21. The following number "1" represents the first operation of the first artifact, denoted as O11. When the number "2" appears again, it represents the second step of the second job, referred to as O22. Subsequent numbers follow the same coding rules, and the length of the chromosomes matches the total number of operations. Therefore, the operation sequence "2-1-2-3-1" can be interpreted correspondingly: "O21-O11-O22-O31-O12".

Table 1. Example of two-layer coding.

OS				
2	1	2	3	1
2	4	1	3	2
MS				

J_2	J_1	J_2	J_3	J_1
2	1	2	3	1
O_{21}	O_{11}	O_{22}	O_{31}	O_{12}
O_{11}	O_{12}	O_{21}	O_{22}	O_{31}
2	4	1	3	2
M_2	M_4	M_1	M_3	M_2

Similar to the OS part, the chromosome length of the MS part is equal to the total number of operations. Taking the MS part in Figure 1 as an example, if an operation, such as O22, has a digit of 3, it means that a third machine is selected to perform the operation. The exact number of machines available depends on the actual situation of the particular problem. So in the MS section, each number is specifically set to represent the machine number that is performing the current operation.

3.2 Adaptive Parameter Adjustment Mechanism

In the SAGA algorithm, the crossover probability p_c and mutation probability p_m are key parameters that are dynamically adjusted. This dynamic adjustment mechanism can adaptively balance the global search ability and local search ability based on the fitness distribution of the population, thereby avoiding the algorithm from falling into local optima too early. The specific adjustment formula is as follows:

$$p_c = 1/1 + e^{-\alpha(F_{avg} - F_{min})}$$

$$p_m = 1/1 + e^{-\beta(F_{avg}-F_{max})}$$

Among them, F_{avg} represents the average fitness of the population, F_{min} and F_{max} represent the minimum and maximum fitness of individuals in the population, respectively, and α and β are regulatory parameters used to control the impact of fitness differences on probability.

Through this dynamic adjustment mechanism, when there is a significant difference in the fitness of the population, the crossover probability pc will increase, thereby promoting global search capability; When the fitness difference of the population is small, the mutation probability pm will increase, thereby enhancing the local search ability. This adaptive adjustment mechanism enables the SAGA algorithm to dynamically balance global search and local search during the evolution process, thereby improving the optimization performance of the algorithm.

3.3 Computational Complexity Analysis

The time complexity of SAGA algorithm mainly consists of population initialization, selection operation, crossover operation, and mutation operation. Assuming the population size is N and the problem size is m × n (where m is the number of machines and n is the number of work pieces), the time complexity analysis of each part is as follows:

(1) population initialization: The time complexity is O (N × m × n). This is because it is necessary to initialize the operation sequence and machine selection for each individual.

(2) Select operation: Adopting a three-way tournament selection with a time complexity of O (N × m × n). This is because each selection operation requires comparing the fitness of three individuals

(3) Cross operation: For double layered chromosomes, the time complexity of crossover operation is O (N × m × n). This is because the crossover operation requires traversing every gene on the chromosome and performing gene exchange based on the crossover probability.

(4) Mutation: The time complexity of mutation operation is O (N × m × n). This is because mutation operations require traversing every gene on the chromosome and mutating genes based on mutation probability.

Therefore, the total time complexity of the SAGA algorithm is O (N × m × n). Compared with traditional genetic algorithms, SAGA reduces invalid searches by adjusting its search domain strategy, thus demonstrating better scalability in large-scale problem instances. In addition, the adaptive parameter adjustment mechanism further optimizes the running efficiency of the algorithm, achieving a good balance between optimization accuracy and computational complexity.

3.4 Improve the framework of GA

In order to more intuitively demonstrate the overall design and operation process of the SAGA algorithm, we propose a complete improvement framework and present it in the

form of a flowchart. This framework combines adaptive parameter adjustment mechanism and search domain self adjustment strategy, aiming to improve the global search ability and convergence speed of the algorithm while avoiding premature convergence.

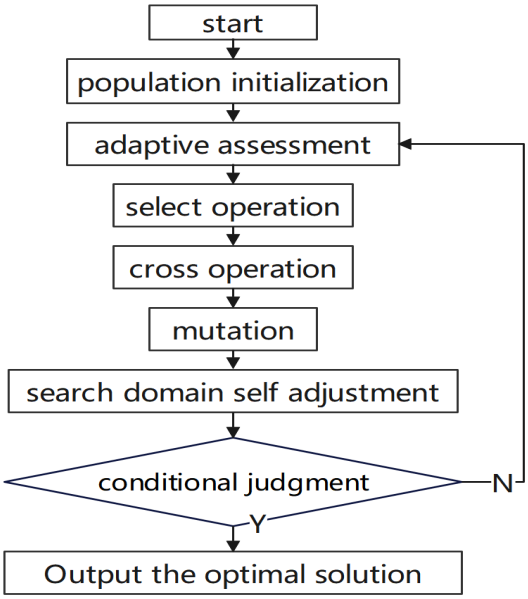

Fig. 1. Algorithm frame diagram

4 Simulation Experiment and Analysis

The experiment uses two widely recognized FJSP benchmark datasets: the Kacem dataset, which includes 5 test instances and is relatively small in scale, suitable for verifying the performance of the algorithm on small-scale problems. Brandimarte dataset: contains 10 test instances, with a large scale, suitable for verifying the performance and scalability of algorithms on large-scale problems.

In order to comprehensively evaluate the performance of SAGA algorithm, the following representative algorithms were selected for comparison: Standard Genetic Algorithm (SLGA[7]): a traditional genetic algorithm used to verify the effectiveness of the improved strategy. Enhanced Particle Swarm Optimization (edPSO [8]): An improved particle swarm optimization algorithm with good convergence speed and accuracy. Grey Wolf Optimization Algorithm (GWO[9]): A swarm intelligence based optimization algorithm suitable for complex optimization problems. DL Scheduling: An emerging intelligent optimization algorithm used to validate the competitiveness of SAGA in the latest algorithms.

Table 2 shows the performance comparison between SAGA algorithm and other algorithms on the Kacem dataset. Among them, LB represents the known optimal lower bound.

Table 2. Optimal completion time for Kacem dataset

question	LB	edPSO	GWO	SLGA	DL Scheduling	SAGA
Kcaem01	11	11	11	13	11	11
Kcaem02	14	17	14	16	14	14
Kcaem03	11	-	11	12	11	11
Kcaem04	7	8	7	9	7	7

From Table 2, it can be seen that SAGA achieved the optimal lower bound (LB) in all four instances of the Kacem dataset, demonstrating excellent optimization accuracy. Compared with SLGA, SAGA outperforms or equals SLGA in all instances; Compared with dPSO and GWO, SAGA performs better in the Kcaem04 instance, indicating that SAGA has higher optimization ability and stability in small-scale problems.

Table 3 shows the performance comparison between SAGA algorithm and other algorithms on the Brandimarte dataset. Among them, LB represents the known optimal lower bound.

From Table 3, it can be seen that among the 10 instances of SAGA in the Brandimarte dataset, 6 instances reached the optimal lower bound (LB), and the remaining 4 instances were also close to the optimal solution. Compared with SLGA, edPSO, and GWO, SAGA performs better in instances MK02, MK04, MK05, MK06, and MK07, indicating that SAGA has strong optimization capabilities and scalability in large-scale problems. Compared with DL Scheduling, SAGA performs better in the MK04 and MK06 instances, indicating that SAGA has higher optimization accuracy in complex problems.

Table 3. Optimal completion time for BR data set

	LB	edPSO	GWO	SLGA	DL Scheduling	SAGA
MK01	36	40	41	42	40	38
MK02	24	25	29	28	26	24
MK03	204	208	205	210	204	204
MK04	48	66	63	60	60	52
MK05	167	172	174	175	170	167
MK06	34	60	69	65	60	34
MK07	133	173	147	145	143	133
MK08	523	523	523	523	523	523
MK09	299	307	320	320	312	302
MK10	165	312	249	254	209	175

The experimental results from the Kacem dataset and Brandimarte dataset show that SAGA exhibits high optimization accuracy on both small-scale and large-scale problems. Compared with SLGA, edPSO, and GWO, SAGA achieved the optimal lower bound (LB) in multiple instances, indicating that SAGA is superior or equal to the comparison algorithm in both global and local search capabilities. In addition, SAGA performs better than DL Scheduling on complex problems, indicating that

SAGA has higher adaptability and robustness in dealing with complex scheduling problems.

5 Conclusion

This article proposes an improved genetic algorithm, the Self Adaptive Search Domain Genetic Algorithm (SAGA), for solving the Flexible Job Shop Scheduling Problem (FJSP). SAGA independently processes operation sequences and machine selection through dual layer chromosome encoding, and introduces adaptive parameter adjustment mechanism and search domain self adjustment strategy, effectively improving the global search ability and convergence speed of the algorithm, avoiding premature convergence.

In the future, the SAGA algorithm can be further optimized to reduce computational complexity, and real-time scheduling strategies in dynamic production environments can be explored to verify the application effectiveness of SAGA in actual production systems. In summary, SAGA provides an efficient and robust solution for solving FJSP, which has important theoretical and practical application value.

References

1. LONG X J, ZHANG J T, QI X, et al, F.: A self-learning artificial bee colony algorithm based on reinforcement learning for a flexible job-shop scheduling problem. Journal 34(4):e6658 (2023).
2. LIN X, ZHU Y X, F.: Exploration of Workshop Scheduling System Based on Particle Swarm Optimization Algorithm. Journal 20(17):104-106 (2024).
3. WANG S T, LIN R H, F.: Hybrid Taboo Search Genetic Optimization Algorithm for Asynchronous Mixed Flow Workshop Scheduling. Journal 1-15 (2024).
4. TANG S M, F.: Optimization Design of Data Analysis System Based on Ant Colony Intelligent Algorithm. Journal 32(11):37-41 (2024).
5. WANG Y Q, F.: Flexible Workshop Scheduling Algorithm Based on Particle Swarm Optimization and Ant Colony Hybrid Algorithm. Journal 31(17):65-69(2023).
6. ZHANG G H, GAO L, SHI Y, F.: An effective genetic algorithm for the flexible job-shop scheduling problem. Journal 38(4):3563-3573(2011).
7. CHEN R H, YANG B, LI S, et al, F,: A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem. Journal 149:106778(2020).
8. NOURI H E, BELKAHLA DRISS O, GHÉDIRA K, F.: Solving the flexible job shop problem by hybrid metaheuristics-based multiagent model. Journal14(1):1-14(2018)
9. JIANG T H, ZHANG C, F.: Application of grey wolf optimization for solving combinatorial problems: job shop and flexible job shop scheduling cases. Journal 6:26231-26240(2018)