Research Article

Multipopulation GA/IWO with Coupled Scale-Free Networks for Solving Flexible Job-Shop Scheduling Problems

Yuqiang Shi, 1,2 Dingshan Deng 3, Jiankang Yin, 4 Li Luo 1, and Xiaoqiu Shi 2

1 Business School, Sichuan University, Chengdu 610000, China
2 School of Manufacturing Science and Engineering, Southwest University of Science and Technology, Mianyang 621000, China
3 School of Automation, Chengdu University of Information Technology, Chengdu 610225, China
4 Sichuan Provincial Tobacco Companies Chengdu Company, Chengdu 610000, China

Correspondence should be addressed to Dingshan Deng; ddsszpy@163.com

Received 25 April 2022; Accepted 13 September 2022; Published 24 September 2022

Academic Editor: Hao Gao

Copyright © 2022 Yuqiang Shi et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

In recent years, an increasing number of population-based metaheuristic algorithms have been proposed to solve the flexible job-shop scheduling problem (FJSP) according to its practicality and complexity. Most of these algorithms are single-population-based and hence are very susceptible to becoming trapped in local optimal values. Thus, multipopulation methods are widely used to improve these algorithms, resulting in multipopulation algorithms. These multipopulation algorithms have been widely studied in the context of single-layer complex networks recently. However, coupled networks used to control two (or more) algorithms simultaneously to get a better algorithm are always ignored in literature. Therefore, in this paper, using coupled scale-free networks (with different scaling exponents) to control the genetic algorithm (GA) and the invasive weed optimization (IWO) simultaneously, a multipopulation GA/IWO with coupled scale-free networks (MPGAIWO-SF) is proposed to solve FJSP. Then, we study how some parameters (e.g., subpopulation size, subpopulation number, and scaling exponent) of MPGAIWO-SF affect its performance, which is also ignored in literature. The simulation results illustrate that (1) the performance of MPGAIWO-SF is significantly improved compared with that of GA and IWO; (2) as the subpopulation size increases, the performance of MPGAIWO-SF first becomes better and then remains almost unchanged; (3) as the subpopulation number increases, the performance of MPGAIWO-SF first becomes better and then decreases rapidly; and (4) the performance of MPGAIWO-SF becomes worse slightly as the scaling exponent decreases. Finally, solving more FJSP instances, the MPGAIWO-SF with optimized parameters is compared with other related algorithms to verify its effectiveness.

1. Introduction

Effective scheduling is very essential for many manufacturing firms to survive well in today’s fiercely competitive marketplace [1]. Therefore, numerous scheduling problems have been extensively studied in recent years, such as flexible job-shop scheduling problems [2–4], job-shop scheduling problems [5, 6], and flow-shop scheduling problems [7, 8]. Among them, the flexible job-shop scheduling problem (FJSP) was introduced for the first time by Brucker and Schlie [2], wherein each operation can be processed on more than one machine, i.e., machine flexible process or path flexible process [9]. This flexible process enables FJSP to be more adaptive and valuable to actual production, where the demand for personalized and customized products is increasing and thus flexible and individualized production is necessary [10, 11]. In addition, FJSP is more complex than the classical job-shop scheduling problem which is NP-hard [5], wherein each operation can just be processed on only one machine. Due to its practicality and complexity, FJSP is of great importance and has been solved by using different algorithms, including heuristic algorithms [12] and population-based metaheuristic algorithms [9, 13]. Recently, different population-based metaheuristic algorithms are more likely to be used to solve FJSP [14], such as the evolutionary grey wolf optimizer [15], African buffalo optimization algorithm [16], bacterial foraging optimization algorithm [17], Jaya algorithm [18], artificial bee colony...
algorithm [19, 20], interactive coevolutionary algorithm [21], fruit fly algorithm [22], virus optimization algorithm [23], particle swarm optimization [1, 9, 24], genetic algorithm [25], and invasive weed optimization [13]. For example, focusing on the innovation of information communication among particles, Ding and Gu [9] proposed an improved particle swarm optimization algorithm which exhibits remarkable ability to solve FJSP. Lu et al. [23] applied an improved virus optimization algorithm to solve FJSP, indicating that it can exhibit better performance than other related algorithms.

Among the aforementioned algorithms, both genetic algorithm (GA) and invasive weed optimization (IWO) are widely used population-based metaheuristic algorithms. To solve a problem by using these population-based metaheuristic algorithms, the feasible solutions of the problem to be solved are first encoded into individuals that can be manipulated easily by some operators (e.g., crossing operator, mutation operator, etc.); then, a number of individuals constitute a population; individuals in the population communicate with each other and hence coevolve under the influence of their fitness; finally, find the optimal or near-optimal solution to the problem [26, 27].

Some single-population-based metaheuristic algorithms (e.g., a single-population-based GA) are very susceptible to becoming trapped in locally optimal solutions [13], and hence a multipopulation method is usually applied to improve these algorithms (algorithms represent metaheuristic algorithms from then on in this paper), resulting in multipopulation algorithms [28, 29]. To get a multipopulation algorithm, the population of a basic algorithm is divided into $N$ (subpopulation number) subpopulations with the same number of individuals (subpopulation size denoted by $S$); individuals in the same subpopulation communicate with each other and coevolve through various operators, while subpopulations communicate with each other through the migrations of elite individuals. If the subpopulations of a multipopulation algorithm are regarded as nodes and the migrations of elite individuals among different subpopulations are regarded as edges, the multipopulation algorithm can be regarded as a network [3]. Previous studies have shown that the structure of a network can seriously affect its behavior. For example, both the cooperative evolution [30, 31] and the risk propagation [32, 33] on networks have been proved to be affected seriously by their network structures. Similarly, the network structure abstracted from a multipopulation algorithm should also affect the behavior of this multipopulation algorithm [4]. In fact, numerous scholars have studied some algorithms in the context of networks. For example, using different crossing operators to different subpopulations, Herrera and Lozano [34] studied the performance of GA modified by regular networks. Cantú-Paz [35] also used some regular networks (e.g., fully connected topology, uni- and bidirectional rings, regular hypercube, etc.) to design GAs and then studied their performance. Giacobini et al. [36] utilized regular one-dimensional and two-dimensional lattices to design evolutionary algorithms and then studied their selection pressures. Bryden et al. [37] applied twenty-six types of networks (e.g., complete graph, complete bipartite graph, n-cycle, tree, etc.) to design evolutionary algorithms. It is worth noting that the networks used in the aforementioned studies are all regular networks. But, in reality, networks are mostly irregular networks, such as small-world networks [38] and scale-free networks (SFNs) with different scaling exponents [39–42]. For example, the real-life networks studied by Kuhnert et al. [40], Parhi [41], and Perera et al. [42] are all scale-free networks and their scaling exponents are 1.10, 2.52, and 2.31, respectively. Accordingly, Payne and Eppstein [43] studied the selection pressures of evolutionary algorithms designed by SFNs, where single-layer SFNs were used to control the communication relationships among individuals. However, only two fitness values (0 and 1) of individuals were analyzed in their study, which limited the application of their results in practical problems. Giacobini et al. [44] applied scale-free and small-world networks to limit possible crossover partners in a population, revealing that evolutionary algorithms designed by scale-free networks are not better than those designed by regular networks. Kirley and Stewart [45] used scale-free, small-world, and random networks to limit possible crossover partners in a population and then studied the performance of differently obtained algorithms on multiobjective optimization problems. Mateo and Alberto [46] used directed domination graphs to represent the domination relations between individuals in the population, obtaining a multiobjective evolutionary algorithm with an improved performance. The influence of network structures on the selection pressures of algorithms was also studied theoretically by Allen et al. [47].

In summary, most of the existing studies focused on utilizing networks to control the communication relationships among individuals and studying the selection pressures of single-population-based algorithms theoretically. And few existing studies (e.g., Herrera and Lozano [34], Cantú-Paz [35], and Giacobini et al. [36]) used networks to control the communication relationships among subpopulations and then studied multipopulation algorithms. However, the networks used in [34–36] are all regular networks. Therefore, the multipopulation algorithms designed by irregular networks are always ignored, let alone using them to solve FJSP. In fact, in our previous studies [3, 4], we have used different networks (e.g., SFNs, ER networks [48], small-world networks, etc.) to design multipopulation GAs and then studied how different network structures affect the performance of the corresponding multipopulation GAs in the context of FJSP. Unfortunately, how different SFNs with different scaling exponents affect the performance of multipopulation algorithms is still not clear. Furthermore, networks utilized in all of the aforementioned existing studies are single-layer networks, while networks in reality often depend on each other to form coupled networks [49, 50]. Compared with single-layer networks, coupled networks are more conducive to compound use of different algorithms.

Therefore, to overcome these drawbacks, we propose a multipopulation algorithm with coupled SFNs (with different scaling exponents) controlling two different algorithms in this paper. As mentioned in [13], GA and IWO are very suitable for compound use. Thus, we use coupled (two
layers) SFNs to design GA and IWO, resulting in a multipopulation GA/IWO with coupled scale-free networks (called MPGAIWO-SF). In MPGAIWO-SF, one layer of the corresponding coupled SFN is used to control the communication relationships of elite individuals among GA subpopulations and another layer is used to control that of IWO subpopulations. The communication relationships of elite individuals between GA and IWO subpopulations are controlled by the one-to-one interconnecting pattern of the coupled SFN. Then how the parameters \( (N, S, \text{scaling exponents}) \) affect its performance of MPGAIWO-SF is studied. Finally, the MPGAIWO-SF with optimized parameters is compared with other algorithms to verify its effectiveness.

In conclusion, this paper contributes to the existing literature by filling in the following research gaps:

1. This paper uses network to control the communication relationships among multiple populations, while previous studies mainly focused on the communication relationships among individuals in a single population. In fact, the network structure formed by the communication relationships among multiple populations may also affect the performance of a multipopulation algorithm.

2. This paper uses SFNs to design multipopulation algorithms, while most previous studies used regular networks. As aforementioned, most networks in reality are irregular networks, such as SFNs with different scaling exponents. Therefore, this paper studies how SFNs with different scaling exponents affect the performance of multipopulation algorithms.

3. This paper uses the interdependent coupling network to realize the combination of different algorithms, while the previous research used a single-layer network to control an algorithm. In reality, multilayer networks often depend on each other to form a coupled network. The coupled network is more conducive to the combination of different algorithms.

The outline of this paper is as follows. Section 2 first describes FJSP simply and then gives its mathematical model formally. In Section 3, MPGAIWO-SF is proposed and described carefully. We explain how to use MPGAIWO-SF to solve FJSP in Section 4. In Section 5, we first study how the parameters \( N, S, \) and scaling exponents of MPGAIWO-SF affect its performance and then the MPGAIWO-SF with optimized parameters is compared with other algorithms to verify its effectiveness. And we conclude this paper in Section 6.

2. Flexible Job-Shop Scheduling Problem

FJSP has been studied extensively due to its practicality and complexity [51–53]. Based on these studies, FJSP can be described as follows.

There are \( n \) jobs \( \{J_1, J_2, \ldots, J_n\} \) that are processed on \( m \) machines \( \{M_1, M_2, \ldots, M_m\} \) in a job-shop. The \( i \)-th job (denoted by \( J_i \)) consists of \( n_i \) operations \( \{O_{i1}, O_{i2}, \ldots, O_{in_i}\} \). The \( j \)-th operation of \( J_i \) (denoted by \( O_{ij} \)) can be processed on more than one machine, resulting in a candidate machine set (denoted by \( S_{ij} \)). \( O_{ij} \) can be processed on the \( k \)-th machine (denoted by \( M_k \)) which belongs to \( S_{ij} \) and the processing time of \( O_{ij} \) on \( M_k \) is denoted by \( F_{ijk} \). It is worth noting that the processing time of an operation on different machines can be different. In addition, all machines and materials in the job-shop are well-prepared, regardless of machine failures; the loading time of an operation is included in its processing time; an operation can only be processed on one machine at a given time without interruption; one machine can only process one operation at a given time; and the operations of a job can just be processed one after another according to its processing constraints, while the processing order of the operations of different jobs assigned to a machine is not fixed. Usually, FJSP includes two subproblems: machine selection, that is, choosing a proper machine for each operation, and operation arrangement, that is, giving an appropriate processing order for all operations assigned to a machine on the premise of processing constraints. Therefore, FJSP is to determine the start and completion times of each operation on the selected machine on the premise of processing constraints so as to optimize one or more given objectives. The common objectives include minimizing the maximum completion time, minimizing the machine load, and so on. In this paper, we consider the objective of minimizing the maximum completion time, and the corresponding mathematical model is as follows [13, 54]:

\[
\min F_{\max} = \min \left( \max \left( F_{ij} \right) \right),
\]

\[
\text{s.t. } F_{ij} - F_{i(j-1)} - P_{ijk} \times X_{ijk} \geq 0, \forall i, j, k,
\]

\[
\sum_{k \in S_{ij}} X_{ijk} = 1 \land F_{ijk} - B_{ijk} = P_{ijk}, \forall i, j,
\]

\[
F_{i j k} \leq B_{ijk} \lor F_{i j k} \leq B_{i j k}, \forall i, j, k \neq i, j,
\]

\[
X_{ijk} \in \{0, 1\}, \forall i, j, k.
\]

Equation (1) is an objective function, where \( F_{\max} \) is the maximum completion time and \( F_{ij} \) is the completion time of \( O_{ij} \). Symbols “\( \min (\cdot) \)” and “\( \max (\cdot) \)” denote the functions that set the minimum and maximum values, respectively. Equation (2) denotes processing constraints, in which \( F_{ij(j-1)} \) denotes the completion time of the previous operation of \( O_{ij} \) and \( X_{ijk} = 1 \) when \( O_{ij} \) is processed on \( M_k \); otherwise \( X_{ijk} = 0 \). Symbol “\( \lor \)” denotes “any.” Equation (3) denotes that an operation can only be processed on one machine at a time without interruption, in which \( F_{ijk} \) and \( B_{ijk} \) denote the start and completion times of \( O_{ij} \) on \( M_k \), respectively, and symbol “\( \land \)” denotes a “Logical AND.” Equation (4) denotes that one machine can only process one operation at a time, in which symbol “\( \lor \)” denotes a “Logical OR.” Equation (5) denotes
variable constraints. Table 1 gives an example of FJSP including 2 jobs and 5 machines (2 × 5), wherein a zero indicates that the corresponding operation cannot be processed on the corresponding machine, while a positive integer denotes the processing time of the corresponding operation on that machine.

| Jobs | Operations | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$J_1$</td>
<td>$O_{11}$</td>
<td>8</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$O_{12}$</td>
<td>1</td>
<td>5</td>
<td>0</td>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>$J_2$</td>
<td>$O_{21}$</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>$O_{22}$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>$O_{23}$</td>
<td>10</td>
<td>0</td>
<td>7</td>
<td>2</td>
<td>0</td>
</tr>
</tbody>
</table>

3. Multipopulation GA/IWO with Coupled Scale-Free Networks

A network can be described as a graph $G = (V, E)$, $V = \{v_i | i = 1, 2, \ldots, N\}$ is the node set of $G$, where $v_i$ is the $i^{th}$ node of $G$ and $N$ is the number of nodes in $G$. $E = \{(v_i, v_j) | e_{ij} = e_{ji} = 1, i, j = 1, 2, \ldots, N\}$ is the edge set of $G$, where $e_{ij} = e_{ji} = 1$ means that the nodes $v_i$ and $v_j$ are connected with each other through this edge. A network is called a scale-free network if its degree distribution obeys power-law with a scaling exponent, i.e., $P(D) \sim D^{-\gamma}$, where $D$, $P(D)$, and $\gamma$ denote the degree, degree distribution, and scaling exponent, respectively [39]. BA model [39] is the most famous model for generating SFNs. However, the scaling exponent of an SFN generated by the BA model is a constant, 3. Therefore, we use the model proposed by Payne and Eppstein [43] and Krapivsky et al. [55] to generated SFNs with different scaling exponents. This model can be described as follows. Start with a fully connected network ($G_0$) with $m_0$ nodes. In each time step $(t)$, a newly arriving node $(j)$ connects to $m$ ($\leq m_0$) existing nodes. The probability ($P_{\text{new}}$) that an existing node $(i)$ is connected with $j$ is defined as follows:

$$P_{\text{new}}(j \rightarrow i) = \frac{A_i}{\sum_{k \neq i} A_k},$$

where $\forall k$ denotes that the summation “$\sum$” covers all existing nodes, and $A_i$ is defined as follows:

$$A_i = \begin{cases} m_0, & D_i \leq m_0, \\ \mu \times D_i, & D_i > m_0, \end{cases}$$

where $D_i$ is the degree of $v_i$, which is defined as the number of edges connecting to $v_i$, $\mu(>0)$ is a tunable parameter. The scaling exponent of an SFN generated by this model can be controlled by $\mu$ [55]. Thus, we study how the parameter $\mu$ affects the performance of MPGAIWO-SF in the following simulations.

In this paper, we regard the subpopulations of MPGAIWO-SF as nodes, so the concepts of nodes and subpopulations can be used alternatively from then on. To get an MPGAIWO-SF, GA and IWO are first modified according to the methods proposed by Shi et al. [13]. Second, the population of GA is divided into $N$ subpopulations and each includes $S$ individuals. The migrations of elite individuals among subpopulations of GA are controlled by a single-layer SFN. The population of IWO is also divided into $N$ subpopulations and each includes $S$ individuals. The migrations of elite individuals among subpopulations of IWO are controlled by another single-layer SFN. Finally, the two single-layer SFNs are interconnected with each other through the one-to-one interconnecting pattern [49], obtaining a coupled SFN. Figure 1 shows an MPGAIWO-SF with five subpopulations in each layer.

In Figure 1, the black dots and the circles represent the individuals and the subpopulations of this MPGAIWO-SF, respectively. Solid lines between individuals represent the operators of GA or IWO, and solid lines between subpopulations represent the migrations of elite individuals among subpopulations. Meanwhile, dotted lines between different layers represent the migrations of elite individuals between subpopulations of different algorithms. MPGAIWO-SF can be described as in Algorithm 1.

Nowadays, researchers have done a lot of research to overcome the premature convergence of algorithms [56–58]. Multipopulation method is one of the common methods [3]. As described in Algorithm 1, we also use multipopulation method to maintain the population diversity of the algorithm to avoid premature convergence. However, different from previous studies, we use a certain network structure to control the communication of elite individuals between populations. In our previous research [3], we found that the network structure will affect the performance of multipopulation GA. If the propagation rate of dominant genes between populations is too fast (e.g., centralized networks) or too slow (e.g., local networks), better algorithm performance cannot be obtained. In a SFN, the propagation rate of dominant genes is between the centralized network and the local network, and the SFN is one of the most common networks in reality. Therefore, this paper uses a SFN to control the propagation of dominant genes between populations. In addition, this paper uses the coupled SFN to realize the combination of GA and IWO. In existing research, the mixing of different algorithms is also an effective strategy to improve algorithm performance [59, 60]. According to Shi et al. [13], GA can find the optimal solution quickly but is easy to fall into the local optimum. In contrast, IWO is not easy to fall into the local optimum, but it is slower to find the optimal solution. IWO is a simulation algorithm proposed by Mehrabian and Lucas [61] to simulate the process of weed invasion in natural ecosystems. It operates through seed growth and reproduction, spatial expansion, and competitive extinction. Therefore, this paper combines GA and IWO and uses a coupled SFN to control
the communication of elite individuals among populations, in order to get a better algorithm.

4. MPGAIWO-SF for Solving FJSP

When using MPGAIWO-SF to solve FJSP, it mainly involves the processes of encoding, decoding, crossing, mutation, and spatial expansion. They are explained one by one as follows.

4.1. Encoding. We use an integer encoding method [13, 54] to encode the feasible solutions of FJSP into individuals. The integer encoding method can be divided into two stages: machine encoding and operation encoding.

Machine encoding is represented as a string of integers (called machine code), where the number of integers is equal to the total number of all jobs’ operations (denoted by $J_t$). The position of an integer of a machine code represents an operation and the value of an integer represents the ordinal number of the machine in the candidate machine set of the corresponding operation. For example, a machine code of the FJSP instance described in Table 1 is [3 2 1 1 1] which contains five integers ($J_t = 5$) exactly. The first integer 3 represents $O_{11}$ is processed on $M_4$ (rather than $M_3$), the fourth integer 1 represents $O_{22}$ is processed on $M_4$, and so on.

Operation encoding is also represented as a string of integers (called operation code) whose length is also $J_t$, where the position of an integer denotes the processing order. The value of an integer denotes the job number of a job and this integer will appear exactly $n_i$ (suppose the job number is $i$) times. For example, an operation code of the FJSP instance described in Table 1 is [2 1 2 2 1], where the integer 2 appears three times exactly because $J_2$ involves three operations. The first integer 2 indicates that $O_{21}$ is processed at this time; the second integer 1 indicates that $O_{11}$ is processed at this time; the third integer 2 indicates that $O_{22}$ is processed at this time, and so on. Finally, the string [3 2 1 1 1 2 1 2 2 1] is an individual.

4.2. Decoding. In order to calculate the fitness of an individual, this individual needs to be decoded into an actual scheduling scheme. In this paper, we use the decoding algorithm described by Shi et al. [13] to decode an individual into an actual scheduling scheme. This decoding algorithm will not be described in this paper repeatedly, and we recommend the readers to read one of our previous papers [3, 4, 13, 54] for a detailed description.

4.3. Crossing. Since the integer encoding method is divided into two stages, the crossing process used in this paper is also divided into two segments: machine crossing (corresponding to machine code) and operation crossing (corresponding to operation code). While the standard two-point crossing operator of GA is adopted to conduct machine crossing, the crossing operator described by Shi et al. [13] is adopted to conduct operation crossing. The operation crossing can be described as follows. There are two operation codes (parent 1 and parent 2) that are crossed, obtaining two offspring (offspring 1 and offspring 2). All jobs (i.e., integers) are randomly divided into two groups (group 1 and group 2). The offspring 1 (offspring 2) inherits the integers which belong to group 1 (group 2) of parent 1 (parent 2), while keeping the positions of these integers unchanged. The offspring 1 (offspring 2) inherits the integers which belong to group 2 (group 1) of parent 2 (parent 1), while keeping the sequence of these integers unchanged.

4.4. Mutation. As described by Shi et al. [13], the mutation process used in this paper also involves two stages: machine mutation (corresponding to machine code) and operation
FOR \( i = 1: N \)

**INITIALIZATION**

**GA INITIALIZATION** According to an encoding method which will be explained later, a number of \( S \) individuals are randomly initialized. These \( S \) individuals are included in a subpopulation.

**IWO INITIALIZATION** This step is the same as GA INITIALIZATION.

**END FOR**

WHILE \( k \leq I_{\text{max}} \) (the number of total iterations)

FOR \( i = 1: N \) (for GA)

**DECODING** According to a decoding algorithm which will be explained later, the fitness of each individual in the current subpopulation is obtained.

**SELECTING** According to the obtained fitness, a standard competition selection of GA is used to generate the next subpopulation, keeping the subpopulation size unchanged.

**ELITE** The elite individual of the subpopulation in question is selected and stored.

**CROSSING** We use a crossing operator which will be described later to get the next subpopulation.

**MUTATION** We use a mutation operator that will be described later to get the next subpopulation.

**END FOR**

FOR \( i = 1: N \) (for IWO)

**DECODING** According to a decoding algorithm which will be explained later, the fitness of each individual (weed) in the current subpopulation is obtained.

**COMPUTING SEED NUMBER** According to the obtained fitness, the seed number (denoted by \( N_{\text{ind}} \)), which is defined as the number of seeds that each individual can produce, is calculated as follows [13]:

\[
N_{\text{ind}} = [(f_{\text{max}} - f_{\text{now}}) \times s_{\text{max}} - s_{\text{min}}]/[f_{\text{max}} - f_{\text{min}} + s_{\text{min}}]
\]

where \( f_{\text{max}}, f_{\text{min}} \) and \( f_{\text{now}} \) represent the maximum, minimum, and current fitness values of the subpopulation in question, respectively. \( s_{\text{max}} \) and \( s_{\text{min}} \) denote, respectively, the maximum and minimum number of seeds that can be generated by a weed. Symbol \( "[.\]" denotes rounding function.

**SPATIAL EXPANSION** A number of integers (denoted by \( D_{\text{mut}} \)) which need to be mutated in a seed is obtained as follows [13]:

\[
D_{\text{mut}} = (I_{\text{max}} - I_{\text{now}})^{3} \times D_{\text{max}}/D_{\text{max}} + D_{\text{min}}
\]

where \( I_{\text{max}} \) and \( I_{\text{now}} \) represent, respectively, the number of total iterations and the current number of iterations, while \( D_{\text{max}} \) and \( D_{\text{min}} \) denote the maximum and minimum number of integers which need to be mutated, respectively. Then, using the obtained \( D_{\text{mut}} \), the spatial expansion which will be described later is implemented.

**SELECTING** Individuals with better fitness are selected and the subpopulation size is maintained.

**ELITE** The elite individual of the subpopulation in question is selected and stored.

**END FOR**

**COMMUNICATION** A subpopulation (denoted by \( v_{i}^{\text{GA}} \)) is randomly selected from the SFN that controls GA and the elite individual of \( v_{i}^{\text{GA}} \) is found. According to this SFN, all neighboring nodes (i.e., subpopulations) of \( v_{i}^{\text{GA}} \) and their corresponding elite individuals can be found. According to the obtained fitness, these seed numbers (denoted by \( N_{\text{ind}} \)) are defined as the number of seeds that each individual can produce, and is calculated as follows [13]:

\[
N_{\text{ind}} = [(f_{\text{max}} - f_{\text{now}}) \times s_{\text{max}} - s_{\text{min}}]/[f_{\text{max}} - f_{\text{min}} + s_{\text{min}}]
\]

where \( f_{\text{max}}, f_{\text{min}} \) and \( f_{\text{now}} \) represent the maximum, minimum, and current fitness values of the subpopulation in question, respectively. \( s_{\text{max}} \) and \( s_{\text{min}} \) denote, respectively, the maximum and minimum number of seeds that can be generated by a weed. Symbol \( "[.\]" denotes rounding function.

**END WHILE**

**OUTPUT** The best solution of all subpopulations (including GA and IWO) is our final solution.

**ALGORITHM 1:** The flowchart of MPGAIWO-SF. Multipopulation GA/IWO with coupled scale-free networks.

---

4.5. Spatial Expansion. As described in Algorithm 1, each individual in a subpopulation of IWO (i.e., a component of MPGAIWO-SF) will produce several seeds (\( N_{\text{ind}} \)) according to its fitness. Each such seed will develop into a weed (i.e., a new individual) which is a candidate individual of the next subpopulation of IWO, and this process is called spatial expansion [13]. The spatial expansion can be described as follows. A seed produced by an individual will develop into a
new individual. As known from Section 4.1, this individual consists of a machine code and an operation code. For the machine code of this individual, a number of $D_{\text{mut}}$ positions are selected randomly; for each position (representing an operation), an integer smaller than the number of candidate machines of this operation is generated randomly and then the original integer in this position is replaced with the generated integer. For the operation code of this individual, two different positions are selected randomly and the integers in the two selected positions are exchanged; this process is repeated $D_{\text{mut}}$ times. When all seeds produced by all individuals have developed into new individuals, we accomplish this spatial expansion.

5. Numerical Simulation and Discussion

5.1. Evaluation Index. GA or IWO is actually a stochastic search algorithm with some control strategies [13]. When the total individual number (TIN, defined as the total number of individuals used by an algorithm in the whole searching process) of an algorithm is larger, it should find a better solution, because the algorithm can search more times. In other words, if two algorithms find the same solution, the algorithm with a smaller TIN performs better relatively. Therefore, when we study the influence of different parameters on the performance of an algorithm, TIN should be considered as a constant. Thus, an average optimal value (AOV) and a success rate (SR) based on TIN are used to measure the performance of MPGAIWO-SF in this paper. For a given TIN, each algorithm runs several times (denoted by $N_{\text{tol}}$) independently. Then, AOV can be defined as follows [3]:

$$AOV = \frac{1}{N_{\text{tol}}} \sum_{i=1}^{N_{\text{tol}}} AOV_i,$$

where $AOV_i$ refers to the best solution found by the algorithm in a single run. $N_{\text{tol}}$ is set to be 50 in this paper. SR can be defined as follows [3]:

$$SR = \frac{N_{\text{auc}}}{N_{\text{tol}}} \times 100\%,$$

where $N_{\text{auc}}$ denotes the number of times MPGAIWO-SF can find the optimal value (the optimal value herein refers to the best solution that could be found in the literature to date) among $N_{\text{tol}}$ independent runs. The TIN of MPGAIWO-SF can be approximately calculated as follows:

$$\text{TIN} = I_{\text{max}} \times N \times \left( \frac{S + S \times \frac{s_{\text{min}} + s_{\text{max}}}{2}}{S + S \times \frac{s_{\text{min}} + s_{\text{max}}}{2}} \right).$$

It is worth noting that $I_{\text{max}}$ can be calculated approximately by using equation (10) when TIN is known in advance.

5.2. Simulation Setup. All the parameters of MPGAIWO-SF are as follows: $S$, $N$, $P_r$, $m_0$, $m$, $\mu$, $s_{\text{max}}$, $s_{\text{min}}$, $D_{\text{max}}$, $D_{\text{min}}$, and TIN. $P_r$ is the mutation probability of GA. It is used to increase population diversity but cannot be set too large. In this paper, we mainly use multipopulation to increase population diversity. Thus, $P_r$ is set to be a small constant value (0.01) randomly in this paper. $m$ determines the average number of edges in a network. As discussed in [3], to be fair, we keep the average numbers of edges of different networks (e.g., SFNs with different scaling exponents) unchanged. Thus, $m$ should be a constant. In addition, a real-life network is always a sparse network [62]; hence, we set $m$ to be 2, leading to sparse networks. Because $m$ is less than or equal to $m_0$, we set $m_0$ to be 2 for simplicity. Because the parameters $s_{\text{max}}$, $s_{\text{min}}$, $D_{\text{max}}$, and $D_{\text{min}}$ are studied extensively in our previous studies [13, 54], they are ignored in this paper. In [13], we found that TIN can hardly affect the performance of algorithms (GA and IWO) when it is very large. Thus, it is ignored here and is set to be a large constant number. To provide more chances for communication among subpopulations, we set TIN to be 4,000,000 in this study. In summary, the parameters $P_r$, $m_0$, $m$, $s_{\text{max}}$, $s_{\text{min}}$, $D_{\text{max}}$, $D_{\text{min}}$, and TIN are not addressed here. And they are set to be constants in all the following simulations as follows: $P_r = 0.01$, $m_0 = 2$, $m = 2$, $s_{\text{max}} = 5$, $s_{\text{min}} = 1$, $D_{\text{max}} = 20$, $D_{\text{min}} = 4$, and TIN = 4,000,000. We mainly study how the parameters $S$, $N$, and $\mu$ affect the performance of MPGAIWO-SF.

5.3. How the Parameter $S$ Affects the Performance of MPGAIWO-SF. In this section, how the parameter $S$ affects the performance of MPGAIWO-SF is studied by using the 8×8 (the best solution to date is 14) and 10×10 (the best solution to date is 7) FJSP instances proposed in [63], where $S$ changes from 10 to 200 with a step size of 10 while $N$ and $\mu$ equal 100 and 1, respectively. And each situation runs 50 times independently (i.e., $N_{\text{tol}}$ is 50). Figure 2 shows the simulation results.

In Figure 2, the X-axis, Y-axis (left), and Y-axis (right) denote the values of $S$, AOV, and SR, respectively. For the 8×8 FJSP instance (Figure 2(a)), the performance of MPGAIWO-SF is very poor when $S$ is relatively small. For example, the values of AOV (SR) are 15.22 (16%), 15.5 (8%), and 15.24 (18%) when the values of $S$ are 10, 20, and 30, respectively. As $S$ increases, the performance of MPGAIWO-SF becomes better rapidly. When $S$ is 100 or 150, the performance of MPGAIWO-SF is the best, where the values of AOV and SR are 14.06 and 94%, respectively. When $S$ is larger than 100, the performance of MPGAIWO-SF remains almost unchanged. For the 10×10 FJSP instance (Figure 2(b)), the performance of MPGAIWO-SF also becomes better as $S$ increases. When $S$ is 120 or 140, the performance of MPGAIWO-SF is the best, where the values of AOV and SR are 7.58 and 42%, respectively. Therefore, to obtain a better MPGAIWO-SF, $S$ should not be too small (larger than or equal to 100 is reasonable). This is mainly because when the subpopulation contains a small number of individuals, the diversity of this subpopulation is naturally poor. With the increase of $S$, there are enough individuals in a single subpopulation to maintain its diversity, so the performance of the corresponding MPGAIWO-SF becomes better. Nevertheless, when $S$ increases to a certain extent, it has little influence on MPGAIWO-SF.
5.4. How the Parameter $N$ Affects the Performance of MPGAIWO-SF. In this section, the $8 \times 8$ and $10 \times 10$ FJSP instances are also used to study the influence of $N$ on the performance of MPGAIWO-SF, wherein $N$ changes from 10 to 200 with a step size of 10 while $S$ and $\mu$ equal 100 and 1, respectively. Figure 3 gives the simulation results.

In Figure 3, the $X$-axis, $Y$-axis (left), and $Y$-axis (right) denote the values of $N$, AOV, and SR, respectively. For the $8 \times 8$ FJSP instance (Figure 3(a)), the performance of MPGAIWO-SF first becomes better and then becomes worse as $N$ increases. For example, as $N = 10$, the performance of MPGAIWO-SF is relatively poor, where the values of AOV and SR are 14.56 and 44%, respectively. When the value of $N$ is 50 or 100, the performance of MPGAIWO-SF is the best, where the values of AOV and SR are 14.1 and 90%, respectively. As $N$ increases continuously, the performance of MPGAIWO-SF becomes very poor. For example, the values of AOV and SR are just 15.46 and 6%, respectively, when $N = 200$. The $10 \times 10$ FJSP instance (Figure 3(b)) shows almost the same regulation as Figure 3(a) does, except that the performance of MPGAIWO-SF is the best when $N$ is 80 (the values of AOV and SR are 7.6 and 40%, respectively) and the corresponding MPGAIWO-SF cannot find the optimal value $7$ exactly when $N$ is relatively large (e.g., 160, 190, and 200). Therefore, to get a better MPGAIWO-SF, $N$ should not be too small or too large, and it is appropriate that $N$ is between 50 and 110. This is mainly because it is not conducive to maintaining the diversity of the population as $N$ is very small. However, at this point, the number of individuals in a single subpopulation is moderate ($S = 100$), which is conducive to maintaining the diversity, so the performance of MPGAIWO-SF is not very poor and the value of $N$ has little impact on the algorithm. However, as $N$ increases continuously with a fixed value of TIN, the total number of iterations becomes very small, which makes individuals have no time to accumulate advantageous genes, so the performance of MPGAIWO-SF becomes poor or even cannot find the optimal value at all (Figure 3(b)). In a nutshell, with a fixed TIN, to get a better MPGAIWO-SF, $N$ should not be too small or too large in order to ensure enough diversity and iterations.

5.5. How the Parameter $\mu$ Affects the Performance of MPGAIWO-SF. In this section, the $8 \times 8$ and $10 \times 10$ FJSP instances are also used to study the influence of $\mu$ on the performance of MPGAIWO-SF, where the values of $N$ and $S$ are both 100. And $\mu$ is set to be 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 3, 5, 7, 9, 11, 13, 15, 17, and 19. Figure 4 gives the simulation results.

In Figure 4, the $X$-axis, $Y$-axis (left), and $Y$-axis (right) denote the values of $\mu$, AOV, and SR, respectively. From both the $8 \times 8$ (Figures 4(a) and 4(b)) and $10 \times 10$ (Figures 4(c) and 4(d)) FJSP instances, we can find almost the same regulation that the performance of MPGAIWO-SF becomes worse slightly as $\mu$ increases. Thus, to get a better MPGAIWO-SF, $\mu$ should not be too large.

5.6. Effectiveness of MPGAIWO-SF. In this section, we also use the $8 \times 8$ and $10 \times 10$ FJSP instances to verify the effectiveness of MPGAIWO-SF by comparing it with GA and IWO. The parameters of MPGAIWO-SF are set as follows: $S = 100$, $N = 100$, $\mu = 1$, and TIN = 4,000,000. Meanwhile, by setting the one-to-one interconnecting pattern and the networks of MPGAIWO-SF as null and keeping the other parameters unchanged, the corresponding IWO and GA are obtained. We find that neither the GA nor IWO can find the optimal value $14$ (7), but only the local optimal value $16$ (8). Figure 5 shows the search processes of MPGAIWO-SF, IWO, and GA in a single run.

In Figure 5, the $X$-axis and $Y$-axis denote the makespan and the number of iterations, respectively. For the $8 \times 8$ FJSP instance (Figure 5(a)), GA and IWO find the local optimal value $16$ at the 36th and 90th iterations, respectively, while MPGAIWO-SF can find the optimal value $14$ at the 70th iteration. For the $10 \times 10$ FJSP instance...
(Figure 5(b)), GA and IWO find the local optimal value 8 at the 58th and 60th iterations, respectively, while MPGAIWO-SF can find the optimal value 7 at the 87th iteration. This verifies the effectiveness of MPGAIWO-SF. According to the convergence speed, for the 8 \times 8 FJSP instance (Figure 5(a)), the convergence speed of MAGAIWO-SF is slower than GA but faster than IWO. For the 10 \times 10 FJSP instance (Figure 5(b)), the convergence speed of MPGAIWO-SF is slower than GA and IWO. Although the convergence speeds of GA and IWO are sometimes faster, GA and IWO cannot obtain high performance in global search. According to the above results, MAGAIWO-SF can find a better optimal value.

A standard GA is very susceptible to becoming trapped in local optimal values due to its low population diversity. MPGAIWO-SF algorithm divides the population of GA and IWO into \( N \) subpopulations with the same individuals, respectively, which breaks the state in GA population and overcomes the local optimization in GA algorithm in theory, evolves through the population in GA (IWO), and exchanges the individuals in the population at the same time. A standard IWO is relatively poor due to its low convergence rate. According to the flowchart of MPGAIWO-SF described in Algorithm 1, we can find that, in the early stage of the search process of MPGAIWO-SF, the search process of IWO (a component of MPGAIWO-SF) is almost random due to its large value of \( D_{\text{mut}} \) (too large number of mutation positions), which is not conducive to retaining better individuals. As the search process evolves, the value of \( D_{\text{mut}} \) becomes smaller, resulting in a local search. The \( D_{\text{mut}} \) of IWO is spatially expanded, is continuously searched, is developed into an appropriate value, and reduced the number of mutations, which can theoretically increase the efficiency of the algorithm and retain better individuals. Meanwhile, the individual of IWO will communicate with the individual of GA, which is analogy to adding a local search to GA. In other words, the elite individuals in the GA subpopulation will communicate with the elite individuals in the IWO subpopulation through the coupled network, which can theoretically enrich the search process. MPGAIWO-SF not only uses a multipopulation method to maintain population diversity, but also uses a coupled SPN to combine GA and IWO. Therefore, these improvements make the MPGAIWO-SF better than GA and IWO.

For further verifying the effectiveness of MPGAIWO-SF, we use it to solve more FJSP instances mentioned in [64] and compare it with other algorithms [52, 65–70]. In this situation, the total number of iterations is 300, and the other parameters of MPGAIWO-SF remain unchanged. Tables 2 and 3 gives the results. The two algorithms mentioned in [65] are AIA and HHS. The symbol “/” indicates that the original paper does not give the corresponding data. It is worth noting that the HHS proposed by Yuan et al. [65] and the HA proposed by Li and Gao [52] can find all the optimal values of the mentioned FJSP instances so far; that is why we chose them as benchmark algorithms.

From Table 2, we can find that the optimal value of MPGAIWO-SF for MFJS02 is 446, which is better than 448 found by AIA. When it comes to MFJS05, the optimal value of MPGAIWO-SF is 514, which is better than 527 found by AIA. For MFJS06, the optimal value 634 of MPGAIWO-SF is better than 635 found by AIA. For MFJS04 and MFJS07, the optimal values of MPGAIWO-SF are 554 and 879 which are better than 564 and 928 mentioned in [66], respectively. Figure 6 gives the Gantt chart of MFJS08. From Table 3, we can find that the optimal value of MK02 is 27, which is better than 28 and 29 of HO and LEGA. For MK07 and MK09, the optimal values of MPGAIWO-SF are 142 and 307, which are better than 144 of TS and 312 of AIA, respectively. This means that the performance of MPGAIWO-SF is relatively better than AIA and the algorithm mentioned in [66], indicating the effectiveness of MPGAIWO-SF.
Figure 4: How the parameter $\mu$ affects the performance of MPGAIWO-SF. (a) $0.1 \leq \mu \leq 1$ for the $8 \times 8$ FJSP instance. (b) $1 \leq \mu \leq 19$ for the $8 \times 8$ FJSP instance. (c) $0.1 \leq \mu \leq 1$ for the $10 \times 10$ FJSP instance. (d) $1 \leq \mu \leq 19$ for the $10 \times 10$ FJSP instance.

Figure 5: The search processes of MPGAIWO-SF, IWO, and GA. (a) The $8 \times 8$ FJSP instance. (b) The $10 \times 10$ FJSP instance.
6. Conclusions and Future Work

Recently, in order to avoid premature convergence, more and more population-based metaheuristic algorithms were improved by using multipopulation methods and then were studied in the context of single-layer complex networks. However, how to use a coupled scale-free network to control two or more algorithms for getting a better algorithm is not reported in literature. Moreover, how different coupled scale-free network structures with different scaling exponents affect the performance of the corresponding algorithms is also ignored. Therefore, in this paper, we applied two-layer SFNs to control GA and IWO simultaneously, obtaining an MPGAIWO-SF with a better
most unchanged. To obtain a better MPGAIWO-SF, MPGAIWO-SF first becomes better and then remains almost unchanged. To obtain a better MPGAIWO-SF, S should not be too small (larger than or equal to 100 is reasonable). As N increases continuously, the performance of MPGAIWO-SF first becomes better and then decreases rapidly. To get a better MPGAIWO-SF, N should not be too small or too large, and it is appropriate that N is between 50 and 110. The performance of MPGAIWO-SF becomes worse slightly as μ increases continuously. Therefore, to get a better MPGAIWO-SF, μ should not be too large (smaller than or equal to 1 is reasonable).

In this paper, we use coupled SFNs to design GA and IWO and get the MPGAIWO-SF. And taking FJSP as an example, the influence of different network structure parameters on the performance of the algorithm is studied. FJSP is a typical combinatorial optimization problem. Therefore, the MPGAIWO-SF in this paper can also be used to solve more similar problems. We can also combine more other algorithms through the coupling network to get a better algorithm. Moreover, the performance of the algorithm can be adjusted by changing the network structure.

Unfortunately, in this paper, we just used the multipopulation method (with two-layer SFNs) to improve IWO and GA. The degree of improvement of the performance of the resulting algorithm is limited, which means that, to get a better algorithm, just using the multipopulation method to improve an algorithm is not enough. While using the multipopulation method to improve an algorithm, some other methods (e.g., individual refinement, domain-based knowledge, etc.) should also be used simultaneously in order to get a better algorithm, which is our future work.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest regarding this work.

Acknowledgments

This research was partially supported by the Natural Science Foundation of Southwest University of Science and Technology (20zx7117).

References


