Genetic Programming with Archive for Dynamic Flexible Job Shop Scheduling

Meng Xu , Fangfang Zhang✉ , Yi Mei , Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
{meng.xu, fangfang.zhang, yi.mei, mengjie.zhang}@ecs.vuw.ac.nz

2021 IEEE Congress on Evolutionary Computation (CEC) | 978-1-7281-8393-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/CEC45853.2021.9504752

***Abstract*—Genetic programming (GP) has achieved great success in evolving effective scheduling rules to make real-time decisions in dynamic flexible job shop scheduling (DFJSS). To improve generalization, a commonly used strategy is to change the training simulation(s) at each generation of the GP process. However, with such a simulation rotation, GP may lose potentially promising individuals that happen to perform poorly in one particular generation. To address this issue, this paper proposed a new multi-tree GP with archive (MTAGP) to evolve the routing and sequencing rules for DFJSS. The archive is used to store the potentially promising individuals of each generation during evolution of genetic programming. The individuals in the archive can then be fully utilized when the simulation is changed in subsequent generations. Through extensive experimental tests, the MTAGP algorithm proposed in this paper is more effective than the multi-tree GP without archive algorithm in a few scenarios. Further experiments were carried out to analyze the use of the archive and some possible guesses were ruled out. We argue that the use of archives does increase the diversity of the population. However, the number of individuals in the archive that ranked in the top five of the new population is small. Therefore, the archive may not be able to greatly improve the performance. In the future, we will investigate better ways to use the archive and better ways to update individuals in the archive.**

***Index Terms*—dynamic flexible job shop scheduling, genetic programming, archive**

I. Introduction

Job shop scheduling (JSS) [1] is an important combinatorial optimization problem that has the purpose of optimizing resource allocation. Flexible job shop scheduling (FJSS) [2] is an extension of JSS, and the problem is more complex. In FJSS, an operation can be processed on a set of optional machines. It is well known that FJSS is NP-hard [3]. In the real world, the scheduling process is carried out under dynamic events, such as the arrival of new jobs, order cancellations or modifications, processing time uncertainties, and machine breakdowns, which is known as dynamic flexible job shop scheduling (DFJSS) [4], [5]. DFJSS is a more complex problem than FJSS and it is an attractive aspect of manufacturing systems, which has been intensively studied in the previous decades.

In this work, we consider DFJSS with new job arrivals, that the information about the job is unknown until the job arrives.

Over the years, many approaches have been proposed to solve the DFJSS problem. Dispatching rules are the most widely used methods in the industry to deal with the DFJSS problem because of its simplicity of implementation and ability to react in real time. Many empirical studies [6]–[9] have shown the success of using dispatching rules for the DFJSS problem.

Although dispatching rules have the advantage of generating schedules instantaneously, their design involves demanding experimental studies and is very difficult. Hyper-heuristics [10] has been applied to generate high-level heuristics by selecting or combining a set of low-level heuristics. These new high-level heuristics are then applied to solve various types of NP-hard problems. Genetic programming hyper-heuristics (GPHH), as a hyper-heuristic method, has been widely used to automatically evolve dispatching rules that produce high quality schedules for DFJSS. Literature reviews [11]–[17] have been conducted to show the advantages of the dispatching rules evolved by genetic programming. To improve the generalization of the dispatching rules evolved by GP, most existing methods [11], [14], [15], [18] typically change the training DFJSS simuations during the GP training process. In other words, the random seeds of the DFJSS simulations are reset in each new generation. This strategy is similar to the mini-batch learning [19], which splits the training set into mutually exclusive subsets (i.e., batch), and uses a different subset in each training step. Although such simulation rotation has shown its effectiveness in previous studies, it still has limitations. One major limitation is that the GP does the individual selection purely based on the fitness at the current generation. As a result, some potentially promising individuals may be lost if they happen to have poor fitness for the simulation in the current generation.

To address the issue above, this paper proposes to use an archive in GPHH to store potential individuals of each generation. Archive is an important technique that has been used to assist in solving some scheduling problems. Existing works mainly use archive to store and maintain the solutions obtained during the optimization procedure for multi-objective problem [20]–[22]. The use of archive helps to find better solutions in the evolutionary process and to compare the distance of these solutions, i.e., to analyze whether the result is trapped in a local optimum. In this paper, the archive is used to store individuals during the evolution process.

The overall goal of this paper is to develop an effective GP method with archive to automatically evolve better dispatching rules for DFJSS. The main contributions of this paper are listed as follows:

- A multi-tree GP method with archive (MTAGP) to evolve dispatching rule is proposed. This method considers not only individuals in the current generation, but also individuals in the archive.
- Proper strategies are designed to update the archive and the archive is used during the GP process to store potential individuals (dispatching rules) acquired during the evolutionary process in each generation.
- The effectiveness of the proposed algorithm was verified and the experimental results are analyzed by comparing it with the multi-tree GP without archive (MTGP).

II. Background

A. Dynamic Flexible Job Shop Scheduling

In the shop floor, there are a set of machines $M = \{M_1, M_2, ..., M_m\}$. In a DFJSS problem, the jobs $J = \{J_1, J_2, ..., J_n\}$ arrive at the shop floor over time. Each job J_i has a sequence of operations $O_j = \{O_{j1}, O_{j2}, ..., O_{jl_j}\}$. Each operation O_{ji} can only be processed by one of its optional machines $\pi(O_{ji}) \subseteq M$ and its processing time $\sigma(O_{ij})$ depends on the machine that processes it. Each operation O_{ji} cannot be processed until its preceding operation $O_{j,i-1}$ has been completed. DFJSS aims to determine which operation will be chosen to process next by a idle machine (i.e., a machine that is not processing any operation) and which machine will be chosen to process the ready operation (an operation with its preceding operation already completed). In DFJSS, the jobs arrival to the shop floor dynamically, and the information of a job (e.g., operations, processing time of each operation, due date) is unknown until the job is arrived.

There are a variety of objectives that can be considered in DFJSS, such as the total tardiness, makespan, mean flowtime, total flowtime , etc. In this paper, We consider max flowtime, mean flowtime, and mean weighted flowtime as the three objectives in the DFJSS problem. Here, we use $Fmax$, $Fmean$, and $WFmean$ to represent the max flowtime, mean flowtime, and mean weighted flowtime, respectively. The objective function is shown in the following.

- Minimisation $Fmax = \max\{C_1 - r_1, ..., C_n - r_n\}$
- Minimisation $Fmean = \frac{\sum_{j=1}^{n}(C_j - r_j)}{n}$
- Minimisation $WFmean = \frac{w_j \times \sum_{j=1}^{n}(C_j - r_j))}{n}$

where, r_j, w_j and C_j are the release time, weight and completion time of the job j.

In addition, some normal assumptions, such as no recirculation of jobs, no alternate routing and no machine breakdown are taken into considered in this work.

B. Dispatching Rules for Dynamic Flexible Job Shop Scheduling

A dispatching rule is a priority function used to calculate the priority of the candidate machines or operations based on the current system state and the state of the candidate machines and operations. In DFJSS, a dispatching rule consists of a routing rule and a sequencing rule which is used when the decision point is arrived. In DFJSS, there are two kinds of decision points. Once a machine becomes idle (sequencing decision point), preceding operation decided by the sequencing rule will be selected as the next task for this machine. Once an operation becomes ready (routing decision point), preceding machine determined by the routing rule will be selected to process this operation. The most commonly considered features by the dispatching rules include WIQ (the works in queue), PT (the processing time of operation), SL (the slack) and so on [23], [24]. Dispatching rule is an efficient method to make fast decisions at decision points and can cope well with dynamic events.

C. Genetic Programming Hyper-heuristics

Hyper-heuristic as a heuristic search method aims to select or generate heuristics to efficiently tackle hard computational search problems. Hyper-heuristics are often used in conjunction with machine learning techniques to meet their goals. GP, as a hyper-heuristic method, with the advantages of flexible representations, has been successfully applied to evolve dispatching rules for solving DFJSS problem [25], [26].

The whole process of GP can be divided into two phases, namely the training process and the test process. The output of the training process is a heuristic, which in DFJSS is the dispatching rule. Then, the evolved dispatching rule is used as the input to the test process, which allows measuring the performance of the dispatching rule.

The traditional evolutionary process of genetic programming includes three main processes which are population evaluation, selection, and evolution (reproduction, crossover, mutation). In particular, crossover and mutation are two extremely important processes that determine the evolutionary direction of populations and the merit of their solution. In this paper, archive is used to select dominant populations and thus further influence the direction of crossover and mutation.

D. Related Work

Existing methods [9], [11], [12], [14], [15], [24], [26] that use GP for solving DFJSS problem generate new training DFJSS simuation to evaluate individuals in each generation. The individuals of each generation are obtained based on the population breeding of the previous generation. However, previously eliminated individuals may perform well in subsequent simulations. Therefore, in order to take advantage of individuals already generated in the process of breeding, a multi-tree GP method with archive (MTAGP) is proposed. Archive techniques are mainly applied in multi-objective problems [27]–[30]. For example, Nguyen et al. employ an external archive to store the non-dominated scheduling policies for solving multi-objective DFJSS [31]. New individuals will be created from parents in the archive through crossover, which will help to focus the search for non-dominant scheduling policies in the early stages of the evolution. Zhang et al.

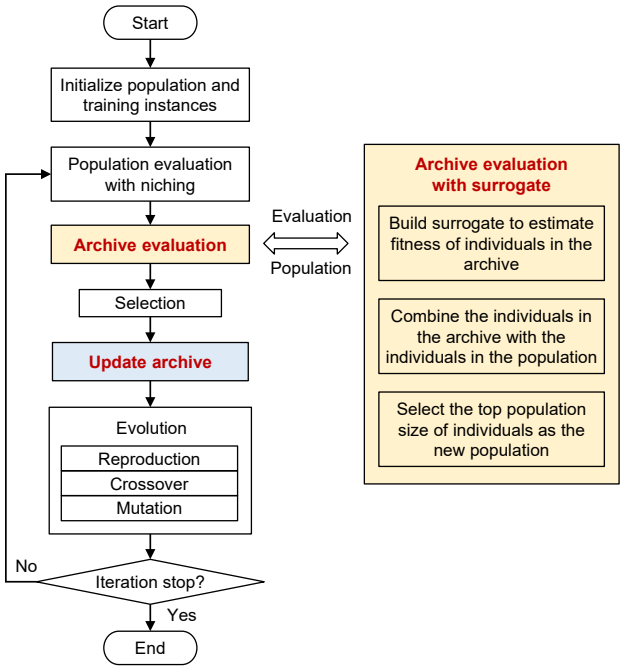

Fig. 1. The flowchart of the proposed MTAGP method.

Algorithm 1: Update the archive for each generation G.

Input: The archive: arc, the archive update rate: R_{arc}, the archive size: S_{arc}, the population: pop
Output: The archive: arc

```
int num = 0;
while num < R_arc × population size do
    for i = 0; i < pop.size(); i++ do
        if arc.size() < S_arc then
            Check if pop_i is repeat with the individuals in arc;
            if Is NOT repeated then
                arc.add(pop_i)
            end
        else
            For each individual in arc, normalise rank and
              generation;
            arc.remove(max(rank + 1/(generation + 1)));
            Check if pop_i is repeat with the individuals in arc;
            if Is NOT repeated then
                arc.add(pop_i)
            end
        end
    end
    num++;
end
return arc
```

incorporate strength Pareto evolutionary algorithm 2 into the GPHH method to solve the multi-objective DFJSS problem [32] with archive assisted. In a word, the archive plays an important role in solving multi-objective problems.

However, to the best of our knowledge, there is currently no method to store promising individuals during GP evolution using archive.

III. The New Approach

In this section, the framework of the proposed MTAGP is illustrated first. Then, the mechanism of archive update is described. Finally, the niching and surrogate assisted mechanism is introduced.

A. The Main Framework of the Proposed Method

The multi-tree GP proposed in [23] is used as the basic framework. Then, this paper improves the multi-tree GP with the help of niching, named MTGP which is used as the baseline method. Further, improve the MTGP with archive, which is named as MTAGP. The flowchart of the MTAGP is shown in Figure 1.

The main processes of GP, including selection, replication, and mutation are adopted from the classical GP method. The main differences between the proposed MTAGP and MTGP are shown as follows.

1. The individuals evaluated in the population evaluation process include individuals from the current population as well as individuals in the archive. When the population evaluation is completed, the individuals in the archive are evaluated based on the surrogate model.

2. The individuals in the population are mixed with the individuals in the archive, and the good individuals with the population size are selected based on the rank to continue the following process. Therefore, the parent individuals in the crossover and mutation process include individuals in the current population as well as individuals in the archive.

In this work, the archive plays an important role in new population selection as well as breeding process, contributing to the evolution of quality individuals.

B. Archive Update Strategies

We need to update the archive when the selection process is finished for each generation. First, we define the archive update rate, i.e., how many individuals need to be updated each time the archive is updated. Also, we define the archive size, i.e., the maximum number of individuals the archive can have. We denote the archive update rate and archive size by R_{arc} and S_{arc}, respectively. Then, the following update strategies will be applied to update the archive.

- For each generation, the number of individuals we update is decided by $R_{arc} \times population\ size$. When the archive is not full, we select high quality individuals to be added to the archive. At the same time, we record which generation this individual come from and its rank in that generation.
- When the archive is full, we delete the worst individual from the archive based on rank and generation recorded. Then add a new high quality individual.
- Each time we update an individual in the archive, we need to first check if the individual is a duplicate of an individual in the archive, thus ensuring the diversity of individuals in the archive.

The details of the procedure to update the archive is given in Algorithm 1.

C. How to Use the Archive?

The archive is used after the population evaluation process and will affect the crossover and mutation process.

First, when the population evaluation process is finished, we need to evaluate the archive. We evaluate the archive using a surrogate model. That is, we build a surrogate model at each generation based on the current simulation and then evaluate the fitness of each individual in the archive. Then, we combine the individuals in the population and the individuals in the archive. Then, we mix the individuals in the population and the individuals in the archive, and rank the individuals from good to poor based on fitness. Finally, we select the former population size individuals from the mixed individuals as the new population for subsequent breeding process. In this way, we can expand the diversity of the population.

Second, since the new population contains individuals from both the original population and the archive, the parents of the crossover and mutation processes also contain individuals from both the original population and the archive. Thus, crossover and mutation processes are also controlled by the individuals in the archive. In this way, we expect higher quality individuals to be evolved.

D. Niching and Surrogate-Assisted Strategies

In addition to the use of archive, niching and surrogate methods are also used in this paper. Niching is used to avoid the evolution from falling into a local optimum. Surrogate model is used to evaluate the individuals in the archive, thus speeding up the evaluation process.

1) Niching: Based on the clearing method in [33], in this paper, we use niching technique by adding a clearing process with a radius $\delta = 0$ and a capacity $k = 1$ after traditional fitness evaluation. Specifically, $\delta = 0$ means that for each considered individual, all the remaining individuals with zero distance (i.e., clones) are cleared. In this paper, phenotypic characterization (PC) [34] is used to calculate the distance between different individuals. The representation of PC is a vector of numerical values, which represents the behavior of individual (GP tree). $k = 1$ means that for each individual in the diverse set, no other individual is within the radius around it. In this way, we can reduce the number of poor individuals in the crowded region of the search space and increase the diversity of the population.

2) Surrogate: The algorithm proposed in this paper is computationally time-consuming due to the use of the archive, which indirectly increases the population size. Therefore, in order to accelerate the evolution of GP with archive, this paper use surrogate model in [34] to estimate the fitness of each individual in the archive. The surrogate model helps to compute approximations of the fitness function efficiently. Specifically, how should we build and use the surrogate model? We build the surrogate model at each generation. In the surrogate model, the number of machines is 10 and the number of jobs is 500. For each generation, we first use the fixed rule (WSPT for sequencing rule, WIQ for routing rule) to generate 20 decision situations, where WSPT means the shortest processing time of the operation with considering the job weights and the meaning of WIQ can be seen in Table I. Then we calculate the PC of each individual in population, respectively, and save them as the nearest neighbor surrogate model. When we evaluate the fitness of each individual in the archive, we calculate the PC of each individual first. We then use the PC as the input of the surrogate model, and the distance to all individuals in the population can be calculated using the euclidean distance between PC. The fitness of the most similar individual was returned as the estimated fitness of the individual in the archive. In this way, we expect the efficiency of archive evaluation to be greatly improved.

IV. Experimental Design

To investigate the effectiveness (the objective values on test instances) of the proposed MTAGP algorithm in different scenarios, a set of experiments was conducted.

First, this section describes parameter settings of GP and the simulation model used in this paper. Then, the test performance is analyzed by comparing the results obtained by MTAGP and MTGP.

A. Parameter Settings for Genetic Programming

In our experiments, Table I shows the terminal set of GP. The terminals are set to represent the features associated with the job, operation and machine. The function set is $\{+, -, \times, \div, max, min\}$ [35]. The arithmetic operators have two arguments. The "$\div$" operator is protected division and returns 1 if it is divided by zero. The max and min functions take two arguments and return the maximum and minimum values of their arguments, respectively. The other parameters of MTAGP are set as shown in Table II.

In this paper, the archive size is set as 512, which is half of the population size (1024). That is, it is expected that at most half of the individuals in the new population will come from the archive, i.e., the archive can play an important role and also the original population will be able to continue to evolve. At the same time, the archive update rate is set as 0.015, which means that there are $1024 \times 0.015 \approx 15$ individuals will be added to the archive or updated in the archive. The reason why the archive update rate is set as 0.015 is that, in this way, we can ensure that the potential individuals at each generation can be retained until the last generation, i.e., we can take advantage of the potential individuals of the first generation in the last generation. At the same time, when the archive is full, the archive can also be updated to a certain extent, so that the individuals that have existed in the archive for a long time can be replaced with different individuals.

B. Simulation Model

In this paper, we use the simulation model as the experimental model to study the impacts of DFJSS. This paper assumes that in our simulation, there are 5000 jobs to be processed by 10 machines. A new job will arrive over time for the DFJSS simulation according to a Poisson process with rate λ. Each job has a different number of operations that are generated

TABLE I
THE TERMINA SET.

Notation	Description
NIQ	the number of operations in the queue
WIQ	the work in the queue
MWT	the waiting time of the machine = t - MRT
PT	the processing time of the operation
NPT	the median processing time for next operation
OWT	the waiting time of the operation = t - ORT
WKR	the work remaining
NOR	the number of operations remaining
W	the job weight
TIS	time in system = t - releaseTime

TABLE II
THE PARAMETER SETTINGS OF GP.

Parameter	Value
Population size	1024
Number of generations	51
Method for initialising population	ramped-half-and-half
Initial minimum/maximum depth	2 / 6
Elitism	5
Maximal depth	8
Crossover rate	0.80
Mutation rate	0.15
Reproduction rate	0.05
Parent selection	Tournament selection with size 7
Terminal/non-terminal selection rate	10% / 90%
The archive size	512
The archive update rate	0.015

randomly from a uniform discrete distribution between 2 and 10. Additionally, the importance of the jobs may vary, expressed by different weights. The weights for 20%, 60% and 20% of the jobs are set to 1, 2 and 4, respectively. The processing time for each operation is assigned according to a uniform discrete distribution in the range of $[1, 99]$. The due date factor is set to 1.5.

The utilization level is an important factor in simulating different scenarios. Utilisation level refers to how busy the machine is. The larger the utilization, the busier the machine and the harder the corresponding problem. In this paper, we consider six different scenarios. These scenarios include three objectives (i.e., max flowtime, mean flowtime, and mean weighted flowtime) and two utilization levels (e.g., 0.85 and 0.95).

To ensure the accuracy of the data collected, warm-up jobs (the first 1000 jobs) were used to obtain typical situations occurring in long-term simulations of the DFJSS system, with jobs arriving in a continuous arrival way. In this paper, data were collected for the next 5000 jobs. When the 6000th job is completed, the simulation stops. Also, we generate only one replication throughout the evolution, but change the random seed during each generation of GP to improve the generalizability of the evolved dispatching rules.

V. RESULTS

A. Test Performance

To verify the effectiveness of the proposed algorithm (MTAGP, multi-tree GP with archive) in this paper, it is compared with the MTGP algorithm (multi-tree GP without archive). Also, in order to make the results realistic and credible, for each scenario, 30 times of training is performed, and for each training, 30 times of testing is performed. Then the Wilcoxon rank sum test with a significance level of 0.05 was used to verify the performance of the proposed method. "−/+" indicates that the corresponding results are significantly better than, or worse than the results of the comparison algorithm. In addition, to accurately verify the usefulness of the archive, we also conducted experiments on the MTAGP without surrogate assisted methods, that is, to exclude the impact of the potentially inaccurate evaluation of the archive due to the use of the surrogate model. The test results are shown in Table III and Figure 2 shows the convergence curves of the three algorithms in six different scenarios.

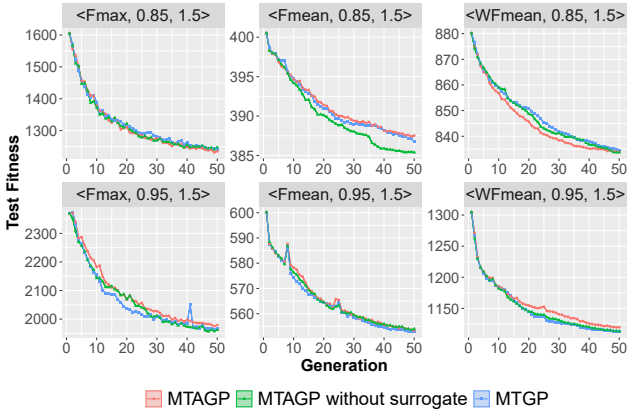

Fig. 2. Convergence curves of test fitness on six scenarios.

Based on the comparison results, we can see that the MTAGP algorithm with surrogate assisted performs basically the same as the MTGP algorithm in most scenarios (i.e., <Fmax, 0.85>, <Fmean, 0.85>, <Fmax, 0.95>, <Fmean, 0.95>, <WFmean, 0.95>). However, in one scenario (i.e., <WFmean, 0.85>), the MTAGP algorithm with surrogate assisted shows an advantage and achieves a better performance.

Also, we can see that the MTAGP algorithm without surrogate performs also basically the same as the MTGP algorithm in most scenarios (i.e., <Fmax, 0.85>, <WFmean, 0.85>, <Fmax, 0.95>, <Fmean, 0.95>, <WFmean,0. 95>). However, in one scenario (i.e., <Fmean, 0.85>), the MTAGP algorithm without surrogate achieved a better performance. Also, according to the convergence curves in Figure 2, we can see that the MTAGP algorithm without surrogate shows a remarkable convergence trend in Scenario <Fmean, 0.85> and achieves better objective function values.

B. Archive Analysis

The main innovation of this paper is the use of archive. Therefore, this section analyzes the individuals in the archive in the MTAGP algorithm. There are totally 512 individuals in archive ($S_{arc} = 512$) after the whole training process. Figure 3 shows the generation that each individual in the archive is from

TABLE III
THE BEST AND MEAN (STANDARD DEVIATION) RESULTS OF 30 INDEPENDENT RUNS OF MTAGP AND MTGP FOR SIX SCENARIOS.

Scenario	MTGP		MTAGP		MTAGP without surrogate	
	Best	Mean(std)	Best	Mean(std)	Best	Mean(std)
<Fmax, 0.85>	1173.14	1241.49(41.28)	1173.32	1235.13(44.73)	1173.95	1247.54(39.93)
<Fmax, 0.95>	1896.30	1966.72(72.47)	1892.24	1978.92(73.63)	1885.85	1962.15(36.86)
<Fmean, 0.85>	383.12	386.78(3.44)	383.30	387.52(4.24)	382.54	**385.39(2.66)(-)**
<Fmean, 0.95>	545.24	552.93(6.34)	545.71	553.35(6.24)	547.69	554.01(7.13)
<WFmean, 0.85>	826.54	834.65(7.79)	826.31	**833.94(9.70)(-)**	826.03	833.73(9.35)
<WFmean, 0.95>	1095.54	1113.18(15.73)	1097.37	1119.64(20.31)	1100.14	1113.40(10.37)

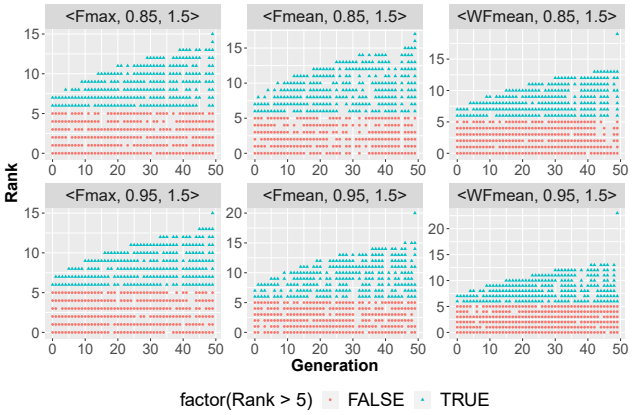

Fig. 3. The Rank versus from which generation of individuals in archive on six test scenarios.

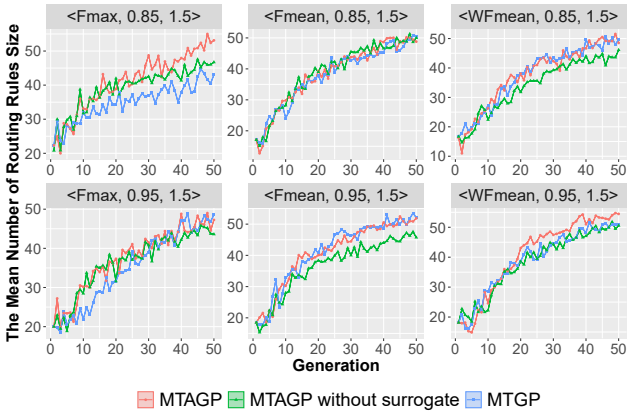

Fig. 4. The mean number of unique features in routing rules per tree.

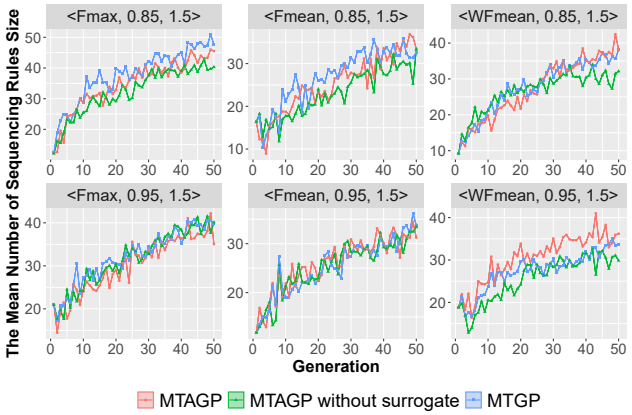

Fig. 5. The mean number of unique features in sequencing rules per tree.

(x-axis) and its rank in that generation (y-axis). According to Figure 3, it can be seen that, at the beginning of GP evolution, the number of individuals in the archive does not reach the set value of the archive size. Therefore, the unduplicated top 15 individuals (no duplication with individuals in the archive, $R_{arc} \times population\ size = 15$) in each generation are added directly to the archive. When the number of individuals in the archive reaches the set size, the unduplicated top 15 individuals in each generation are updated to the archive, while avoiding duplication of individuals in the archive.

In this paper, the breed elite size is 5. Based on Figure 3, the potential individuals (top 5 individuals) in archive from each generation is shown in red circle. We can see that, for each generation, basiclly the top 5 individuals are add to the archive and not be deleted during the whole process of GP evolution. These potential individuals may contribute in the breeding process. In the six scenarios, for each generation, basically top 5 individuals were all retained. However, according to the comparison results in Section V-A, it can be seen that the potential individuals retained in the archive did not play a significant role.

C. The Sizes of Evolved Rules

According to the investigation, the smaller the size of the evolved dispatching rule, the better the interpretability of the dispatching rule [36]. Therefore, this subsection analyzes the size of the dispatching rules evolved by the three algorithms. The mean (standard deviation) results of routing rule size and sequencing rule size of 30 independent runs of MTAGP (with and without surrogate assisted) and MTGP for six scenarios is shown at Table IV. Figure 4 and Figure 5 show the convergence curves of routing rule size and sequencing rule size during the evolution, respectively. Based on Table IV, Figure 4 and Figure 5, we can see that except <Fmax, 0.85>, in all other scenarios, the MTAGP algorithm and the MTGP algorithm have essentially the same sequencing rule size and routing rule size. That is, the use of archive does not affect the size of the dispatching rules evolved by GP, i.e., it does not affect the interpretability of the dispatching rules evolved by GP.

TABLE IV
THE MEAN (STANDARD DEVIATION) RESULTS OF ROUTING RULE SIZE AND SEQUENCING RULE SIZE OF 30 INDEPENDENT RUNS OF MTAGP AND MTGP FOR SIX SCENARIOS.

Scenario	Routing Rule Size			Sequencing Rule Size		
	MTGP	MTAGP	MTAGP without surrogate	MTGP	MTAGP	MTAGP without surrogate
<Fmax, 0.85>	43.00(20.86)	53.13(15.29)(+)	46.72(17.29)	47.60(19.11)	45.53(15.26)	40.31(17.78)
<Fmax, 0.95>	48.53(15.33)	47.27(17.33)	43.60(19.06)	39.93(18.37)	35.07(12.47)	40.13(18.02)
<Fmean, 0.85>	50.60(17.87)	48.80(16.94)	50.00(14.96)	32.60(17.70)	33.47(16.14)	33.33(17.39)
<Fmean, 0.95>	52.07(17.42)	51.87(15.46)	45.73(13.02)	33.80(17.17)	31.27(13.41)	33.40(15.20)
<WFmean, 0.85>	49.73(17.02)	48.52(19.45)	46.13(15.51)	38.13(18.47)	38.24(21.18)	32.13(16.41)
<WFmean, 0.95>	50.87(18.78)	54.47(13.47)	50.53(14.70)	33.60(13.49)	36.20(15.35)	29.87(14.06)

TABLE V
THE GENERATIONS WITH DUPLICATE INDIVIDUALS IN ARCHIVE AND POPULATION, AND THE NUMBER OF DUPLICATE INDIVIDUALS AND ARCHIVE SIZE.

Gen [1]	ArcSize [2]	NumRepeat [3]	NumFromArc [4]	Rank <5 [5]
6	96	1	96	1
9	144	1	136	1
36	512	0	394	4
44	512	0	370	2
49	512	0	341	1

[1] Generation in the evolutionary process.
[2] The number of individuals in the archive in current generation.
[3] The number of repeat individuals between archive and population.
[4] The number of individuals from the archive in the new population.
[5] The number of individuals from the archive and ranked in the top 5 in the new population.

D. Further Analysis

Based on the above analysis, the use of archive does not lead to a significant performance improvement of the evolved dispatching rules. This section presents analysis of possible reasons as to why performance was not being significantly improved. A total of four possible reasons were proposed, and two of them was disproved by experiment.

1) Reason 1: The individuals in archive are chosen from the population of each generation which represents the archive update process. During the archive update process, uniqueness check is done to avoid adding duplicate individual into the archive. However, we select the new population based on the old population and archive before this process. Also, the old population is generated based on the breeding process. Therefore, there may be duplicate individuals between old population and archive when we select the new population based on the fitness. If the number of duplicate individuals is big, then the archive will not improve the diversity, thus will not have the potential to improve the performance.

To verify this, we counted the number of duplicate individuals in the archive and the population before evaluation. Take an example of MTAGP method with surrogate, the results of one of the 30 runs is shown as Table V. As we can see, there are only a few generations in 51 generations, where the archive contains duplicated individuals in the population, and the number of duplicated individuals is very small compared to the archive size, and can be ignored. This means that the use of archive does increase the diversity of the population.

2) Reason 2: It is possible that the number of individuals from the archive in the new population is small, so the individuals in the archive will not have a large effect on the subsequent crossover and mutation process.

Based on this concern, we counted the number of individuals from the archive in the new population, and the number of individuals from the archive ranked in the top five of the new population, as shown in Table V. Based on the table, we can see that there are many individuals in the new population which are from the archive. However, there are a small number of individuals from the archive ranked in the top five of the new population. Therefore, individuals from the archive will play a role in the subsequent crossover and mutation. However, individuals from the archive do not outperform most of the individuals in the population. Therefore, the poor performance is not due to a small number of individuals from the archive.

3) Reason 3: We have seen that the use of archive does increase the diversity of the population. In this case, another possible reason that performance not being improved may be because the archive plays a small role. Specifically, the use of archive is like expanding the population size. Expanding population size theoretically has the potential to yield better solutions. However, with the archive based method, the diversity of individuals retained in the archive actually varies less per generation than with the method of expanding the population size (for example expanding population size from 1024 to 1024 + 512).

4) Reason 4: The strategies to update the archive may not be suitable. Specifically, the way to delete the poor individuals in archive based on the function $max(Rank + 1/(Generation + 1))$, where $Rank$ and $Generation$ have been normalized, may not be suitable. We also tried a different way to delete the poor individuals in archive based on the function $max(0.2 \times Rank + 0.8/(Generation + 1))$, but the performance still was not improved.

VI. CONCLUSIONS

In this paper, a MTGP method with archive for DFJSS was proposed. In the process of GP evolution, an archive is used to store potential individuals from each generation that will be used to influence the breeding process together with individuals in the population. This way, we can fully utilize the potential individuals from each generation without losing them, especially when the training simulations are changed in each generation. The experimental results show that the

proposed MTAGP algorithm outperforms the baseline MTGP algorithm in a few scenarios. Furthermore, we present several analysis on possible reasons that may have led to the lack of significant improvement in the performance of the MTAGP algorithm sometimes. We argue that it may be that the archive is not being used to their full potential, or not being used in a good way. In future work, we will investigate better ways to use archive and conduct further experimental validations.

REFERENCES

[1] C. A. Brizuela and N. Sannomiya. "A diversity study in genetic algorithms for job shop scheduling problems," in Proceedings of the Genetic and Evolutionary Computation Conference, pp. 75–82, 1999.

[2] J. Xie, L. Gao, K. Peng, et al. "Review on flexible job shop scheduling," IET Collaborative Intelligent Manufacturing, vol. 1, no. 3, pp. 67–77, 2019.

[3] I. A. Chaudhry, A. A. Khan. "A research survey: review of flexible job shop scheduling techniques," International Transactions in Operational Research, vol. 23, no. 3, pp. 551–591, 2016.

[4] D. Karunakaran, Y. Mei, G. Chen, et al. "Toward evolving dispatching rules for dynamic job shop scheduling under uncertainty," in Proceedings of the Genetic and Evolutionary Computation Conference, ACM, pp. 282–289, 2017.

[5] F. Zhang, Y. Mei, S. Nguyen, M. Zhang and K.C. Tan. "Surrogate-Assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling," IEEE Transactions on Evolutionary Computation, 2021. Doi: 10.1109/TEVC.2021.3065707

[6] K. C. Liu. "Dispatching rules for stochastic finite capacity scheduling," Computers & Industrial Engineering, vol. 35, no. 1–2, pp. 113–116, 1998.

[7] S. Nguyen, M. Zhang, M. Johnston, et al. "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," IEEE Transactions on Evolutionary Computation, vol. 17, no. 5, pp. 621–639, 2012.

[8] O. Holthaus, C. Rajendran. "Efficient dispatching rules for scheduling in a job shop," International Journal of Production Economics, vol. 48, no. 1, pp. 87–105, 1997.

[9] Y. Mei, M. Zhang. "A comprehensive analysis on reusability of gp-evolved job shop dispatching rules," in Proceedings of the IEEE Congress on Evolutionary Computation, pp. 3590–3597, 2016.

[10] P. Ross. "Hyper-heuristics," Search methodologies. Springer, Boston, MA, pp. 529-556, 2005.

[11] F. Zhang, Y. Mei, M. Zhang. "A new representation in genetic programming for evolving dispatching rules for dynamic flexible job shop scheduling," in Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization. Springer, pp. 33–49, 2019.

[12] S. Nguyen, Y. Mei, B. Xue, et al. "A hybrid genetic programming algorithm for automated design of dispatching rules," Evolutionary computation, vol. 27, no. 3, pp. 467–496, 2019.

[13] D. Karunakaran, Y. Mei, G. Chen, et al. "Evolving dispatching rules for dynamic Job shop scheduling with uncertain processing times," in Proceedings of the IEEE Congress on Evolutionary Computation. IEEE, pp. 364–371, 2017.

[14] Y. Mei, S. Nguyen, M. Zhang. "Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling," in Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning. Springer, pp. 435–447, 2017.

[15] Y. Mei, M. Zhang, S. Nyugen. "Feature selection in evolving job shop dispatching rules with genetic programming," in Proceedings of the Genetic and Evolutionary Computation Conference, pp. 365–372, 2016.

[16] F. Zhang, Y. Mei, S. Nguyen, K.C. Tan and M. Zhang. "Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling," IEEE Transactions on Cybernetics, 2021. Doi: 10.1109/TCYB.2021.3065340

[17] M.A. Ardeh, Y. Mei, and M. Zhang. "Transfer learning in genetic programming hyper-heuristic for solving uncertain capacitated arc routing problem,". in Proceedings of the IEEE Congress on Evolutionary Computation, pp. 49-56, 2019.

[18] X. Chen, X. Hao, H.W. Lin, T. Murata. "Rule driven multi objective dynamic scheduling by data envelopment analysis and reinforcement learning," in Proceedings of the International Conference on Automation and Logistics, IEEE, pp. 396–401, 2010.

[19] H. D. Nguyen, F. Forbes, G. J. McLachlan. "Mini-batch learning of exponential family finite mixture models," Statistics and Computing, pp. 1–18, 2020.

[20] D. Lei. "A Pareto archive particle swarm optimization for multi-objective job shop scheduling," Computers & Industrial Engineering, vol. 54, no. 4, pp. 960–971, 2008.

[21] X. Wen, et al. "Improved genetic algorithm with external archive maintenance for multi-objective integrated process planning and scheduling," in Proceedings of the International Conference on Computer Supported Cooperative Work in Design. IEEE, pp. 385–390, 2013.

[22] L. Wang, C. Fang, C. D. Mu, et al. "A Pareto-archived estimation-of-distribution algorithm for multiobjective resource-constrained project scheduling problem," IEEE Transactions on Engineering Management, vol. 60, no. 3, pp. 617–626, 2013.

[23] F. Zhang, Y. Mei, and M. Zhang. "Genetic programming with multi-tree representation for dynamic flexible job shop scheduling," in Proceedings of Australasian Joint Conference on Artificial Intelligence. Springer, pp. 472–484, 2018.

[24] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang. "Correlation Coefficient Based Recombinative Guidance for Genetic Programming Hyperheuristics in Dynamic Flexible Job Shop Scheduling," IEEE Transactions on Evolutionary Computation, 2021. Doi: 10.1109/TEVC.2021.3056143

[25] H. Al-Sahaf, Y. Bi, Q. Chen, A. Lensen, Y. Mei, et al. "A survey on evolutionary machine learning," Journal of the Royal Society of New Zealand, vol. 49, no. 2, pp. 205–228, 2019.

[26] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang. "Collaborative Multi-fidelity Based Surrogate Models for Genetic Programming in Dynamic Flexible Job Shop Scheduling," IEEE Transactions on Cybernetics, 2021. Doi: 10.1109/TCYB.2021.3050141

[27] M. Masood, M. M. Fouad, R. Kamal, I. Glesk, I. U. Khan. "An improved particle swarm algorithm for multi-objectives based optimization in MPLS/GMPLS networks,". IEEE Access, vol. 7, pp. 137147–137162, 2019.

[28] G. Luo, X. Wen, H. Li, W. Ming, G. Xie. "An effective multi-objective genetic algorithm based on immune principle and external archive for multi-objective integrated process planning and scheduling," The International Journal of Advanced Manufacturing Technology, vol. 91, no. 9, pp. 3145–3158, 2017.

[29] N. Chen, W. N. Chen, Y. J. Gong, Z. H. Zhan, J. Zhang, Y. Li, Y. S. Tan. "An evolutionary algorithm with double-level archives for multiobjective optimization," IEEE Transactions on Cybernetics, vol. 45, no. 9, pp. 1851–1863, 2014.

[30] P. C. Chang, S. H. Chen. "The development of a sub-population genetic algorithm II (SPGA II) for multi-objective combinatorial problems," Applied Soft Computing, vol. 9, no. 1, pp. 173–181, 2009.

[31] S. Nguyen, M. Zhang, M. Johnston, K. C. Tan. "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," IEEE Transactions on Evolutionary Computation, vol. 18, no. 2, pp. 193–208, 2013.

[32] F. Zhang, Y. Mei, M. Zhang. "Evolving dispatching rules for multi-objective dynamic flexible job shop scheduling via genetic programming hyper-heuristics,". in Proceedings of the IEEE Congress on Evolutionary Computation, pp. 1366–1373, 2019.

[33] F. Zhang, Y. Mei, and M. Zhang. "A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling," in Proceedings of the Genetic and Evolutionary Computation Conference. ACM, pp. 347–355, 2019.

[34] T. Hildebrandt and J. Branke. "On using surrogates with genetic programming," Evolutionary Computation, vol. 23, no. 3, pp. 343–367, 2015.

[35] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang. "Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling,". IEEE Transactions on Cybernetics, vol. 51, no. 4, pp. 1797–1811, 2021.

[36] Y. Mei, S. Nguyen, and M. Zhang. "Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling," in Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning. Springer, pp. 435–447, 2017.