Multitask Multiobjective Genetic Programming for Automated Scheduling Heuristic Learning in Dynamic Flexible Job-Shop Scheduling

Fangfang Zhang, *Member, IEEE*, Yi Mei, *Senior Member, IEEE*, Su Nguyen, *Member, IEEE*, and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Evolutionary multitask multiobjective learning has been widely used for handling more than one multiobjective task simultaneously. However, it is rarely used in dynamic combinatorial optimization problems, which have valuable practical applications such as dynamic flexible job-shop scheduling (DFJSS) in manufacturing. Genetic programming (GP), as a popular hyperheuristic approach, has been used to learn scheduling heuristics for generating schedules for multitask single-objective DFJSS only. Searching in the heuristic space with GP is more difficult than in the solution space, since a small change on heuristics can lead to ineffective or even infeasible solutions. Multiobjective DFJSS is more challenging than single DFJSS, since a scheduling heuristic needs to cope with multiple objectives. To tackle this challenge, we first propose a multipopulation-based multitask multiobjective GP algorithm to preserve the quality of the learned scheduling heuristics for each task. Furthermore, we develop a multitask multiobjective GP algorithm with a task-oriented knowledge-sharing strategy to further improve the effectiveness of learning scheduling heuristics for DFJSS. The results show that the designed multipopulation-based GP algorithms, especially the one with the task-oriented knowledge-sharing strategy, can achieve good performance for all the examined tasks by maintaining the quality and diversity of individuals for corresponding tasks well. The learned Pareto fronts also show that the GP algorithm with task-oriented knowledge-sharing strategy can learn competitive scheduling heuristics for DFJSS on both of the objectives.**

***Index Terms*—Dynamic job-shop scheduling, genetic programming (GP), hyperheuristics, multiobjective, multitask learning.**

I. Introduction

EVOLUTIONARY multitask learning/optimization is a learning paradigm that aims at handling multiple tasks simultaneously [1]. The key idea of multitask learning is the knowledge sharing between tasks [2]. A unique feature of multitask learning is the use of skill factor, assortative mating, and vertical cultural transmission to allocate individuals for optimizing tasks [3]. However, existing multitask studies mainly focus on single-objective optimization [4], [5], [6], [7], while the work on multiobjective optimization [8], [9] are still limited. In addition, the limited studies on multitask multiobjective optimization mainly work on continuous numeric optimization problems [10], [11], while the studies on discrete combinatorial optimization problems are rare [2]. To the best of our knowledge, there is no study of multitask multiobjective optimization on *dynamic* combinatorial optimization. Essentially, dynamic combinatorial optimization has a wide range of applications such as goods delivery in logistics [12]; designing water distribution networks [13]; and dynamic flexible job-shop scheduling (DFJSS) in manufacturing [14], [15]. A good schedule for DFJSS is the key for increasing the enterprise benefits by improving customer satisfaction and loyalty [16] and helping businesses react quickly and efficiently to disruptions (e.g., panic buying due to COVID-19). However, research on multitask multiobjective for DFJSS is still in its infancy.

In a real manufacturing, the objectives of production vary at different times. For example, minimizing the mean-flowtime (Fmean) of finishing all clothes in a clothing processing factory is preferred in the off-season to reduce the cost of the entire production line. However, minimizing the max-flowtime (Fmax) of all jobs is preferred to deliver clothes to all customers as soon as possible in busy seasons. Based on our preliminary work [17], we find that there are conflicting objectives in DFJSS, such as minimizing Fmax and Fmean, and minimizing max-tardiness (Tmax) and mean-tardiness (Tmean), which is a naturally multiobjective optimization problem. On the other hand, although the DFJSS tasks can be different, some tasks are related such as minimizing Fmax and Tmax [18], since both of them aim to finish jobs as soon as possible. Handling these tasks simultaneously in a multitask framework can utilize the learned knowledge from a task to another task to improve the quality of all learned scheduling heuristics across tasks.

Manuscript received 23 February 2022; revised 25 May 2022; accepted 31 July 2022. This work was supported in part by the Marsden Fund of New Zealand Government under Contract MFP-VUW1913 and Contract VUW1614; in part by the Science for Technological Innovation Challenge Fund under Grant 2019-S7-CRS; and in part by the MBIE SSIF Fund under Contract VUW RTVU1914. This article was recommended by Associate Editor J. Cao. *(Corresponding author: Fangfang Zhang.)*

Fangfang Zhang, Yi Mei, and Mengjie Zhang are with the Evolutionary Computation Research Group, School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: fangfang.zhang@ecs.vuw.ac.nz; yi.mei@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

Su Nguyen is with the Centre for Data Analytics and Cognition, La Trobe University, Melbourne, VIC 3086, Australia (e-mail: p.nguyen4@latrobe.edu.au).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCYB.2022.3196887.

Digital Object Identifier 10.1109/TCYB.2022.3196887

Genetic programming (GP) [19], as a hyperheuristic approach, has been widely used to learn scheduling heuristics for DFJSS due to its flexible representation [20], [21], [22]. Scheduling heuristics, that is, *a routing rule* and *a sequencing rule*, are applied as priority rules to prioritize candidate machines and operations in DFJSS to make routing and sequencing decisions, respectively [18], [21]. GP generates scheduling heuristics in the space of heuristics rather than solutions. The traditional multitask multiobjective algorithm [3] is an effective multiple-tasks solving framework for numeric continuous optimization on the basis of evolutionary algorithms. GP, as an evolutionary algorithm, can be adapted to a multitask multiobjective approach based on [3]. However, for GP in DFJSS, a small change on heuristics can lead to ineffective or even infeasible solutions. Therefore, the way that shares knowledge between tasks by managing skill factors to assign individuals for tasks as in [3] might not effective for GP [18]. In other words, the effectiveness of a heuristic can get worse if we change it from one task to another task. This problem is more serious on multiobjective GP, since a heuristic needs to cope with more than one objective. A better way to share knowledge among multiobjective DFJSS tasks with GP is worth studying.

The overall goal of this article is to develop an effective multitask multiobjective GP approach to learning scheduling heuristics in handling multiple multiobjective DFJSS tasks simultaneously. The proposed algorithms are expected to learn effective Pareto fronts for all tasks in a multitask scenario. The major contributions of this article are shown as follows.

1) We start with investigating the possible variations of multitask multiobjective GP algorithms for DFJSS by adapting ideas from classical multitask multiobjective algorithm [3] along with the characteristics of DFJSS. This is the first work that studies multitask multiobjective GP on DFJSS. This provides a fundamental study of automated scheduling heuristic learning for GP in DFJSS. It is noted that the adapted algorithms are under the framework of one population as in [3].
2) We have proposed a multipopulation-based multitask multiobjective GP algorithm to maintain the effectiveness of scheduling heuristics for tasks by separating them into different populations. The knowledge sharing between tasks are realized by the crossover operator. The results show that the proposed multitask GP can learn effective scheduling heuristics for all the multiobjective DFJSS tasks in the examined multitask scenarios. The comparison between learning frameworks shows that the multipopulation framework is better than one population for GP to learn scheduling heuristics for multitask multiobjective DFJSS. This also confirms that using skill factors to assign individuals for tasks is ineffective for GP in multitask multiobjective DFJSS.
3) Based on the findings, we have developed a task-oriented knowledge-sharing strategy for the multitask multiobjective GP algorithm to further maintain the individual quality for DFJSS tasks. The results show the superiority of the proposed algorithm among all compared algorithms. Further analyses on the effect of the proposed algorithm reveal that the effectiveness of the proposed algorithm is realized by the improvement of the diversity of individuals for tasks.

II. Literature Review

A. Evolutionary Multitask Multiobjective Optimization

Assuming there are k related multiobjective tasks, evolutionary multiobjective multifactorial optimization (MOMF) aims to handle k tasks simultaneously [3]. MOMF starts with a population that consists of a number of individuals (*initialization*). The same as the classical multitask evolutionary algorithms on a single objective, skill factor τ_i is used to allocate individuals to corresponding tasks, where an individual with τ_i will be evaluated to optimize task T_i (*evaluation*). The skill fitness of individuals is set as their ranks for optimizing the corresponding tasks (i.e., for minimization problem). For *evolution*, parents are selected based on their skill fitness to breed offspring with assortative mating (i.e., states that individuals prefer to produce offspring with the same skill factors) and vertical cultural transmission (i.e., is a mode that offspring is similar to its parents). Newly produced offspring are evaluated along with their skill factors. The parent population and offspring population are emerged and only population size individuals are selected. If the stopping criterion is not met, the algorithm will move to the next generation. Otherwise, the best learned solutions for all tasks will be reported as the outputs of the algorithm.

The main difference between multitask multiobjective optimization and multitask single-objective optimization is the calculation of skill fitness which indicates the quality of individuals. Skill fitness is the rank of individuals which can directly be obtained with the fitness of individuals for multitask single-objective optimization. However, for multiobjective optimization, the individuals on the same front are no-dominated, and assigning them with the same skill fitness is impossible to distinguish the quality between them. To handle this issue, both the front rank and crowding distance of individuals are considered. Assuming there are two individuals (i.e., ind_1 and ind_2), ind_2 is better than ind_1, if: 1) $\text{Rank}(NF_2) < \text{Rank}(NF_1)$ or 2) $\text{Rank}(NF_2) = \text{Rank}(NF_1)$ and $CD_2 > CD_1$, where $\text{Rank}(NF_1)$ and $\text{Rank}(NF_2)$ indicate the rank of the nondominated front of ind_1 and ind_2, and CD_1 and CD_2 represent the crowding distance of ind_1 and ind_2, respectively [3].

B. Multitask Multiobjective DFJSS

In DFJSS, m machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$ have to process n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$. Each job J_j has a sequence of operations $\mathcal{O}_j = \{O_{j1}, O_{j2}, \ldots, O_{jl_j}\}$ that need to be processed one by one, where l_j is the number of operations of job J_j. Each operation O_{ji} can be processed on more than one machine $M(O_{ji}) \subseteq \pi(O_{ji})$ [23]. In addition, the machine that processes an operation determines its processing time $\delta(O_{ji}, M(O_{ji}))$. This article focuses on the most common dynamic event in real life, that is, new jobs arrive both dynamically and stochastically [24], [25], [26]. A new job cannot be

known until it arrives on the shop floor. Below are the main constraints of the DFJSS problems.

1) A machine can only process one operation at a time.
2) At any given time, each operation can be handled by only one candidate machine.
3) An operation cannot be handled until all of its precedents have been processed.
4) Once an operation has been started, the processing cannot be paused or stopped until it has been completed.

1) Decision Making: There are two decisions, that is: 1) *machine assignment* for allocating operations to machines and 2) *operation sequencing* for selecting an operation to be processed next by an idle machine. When an operation becomes ready, a routing rule is triggered to calculate the priority values of its accessible machines, and the most prior machine is used to process this operation. It is noted that the ready operations contain the first operation of a job and the next operation of a just-finished operation. When a machine turns into idle and there are operations in its queue, we will use a sequencing rule to prioritize all the operations in the machine's queue, and pick out the most prior operation to be processed next.

2) Multitask Multiobjective Tasks: For a DFJSS task, different customers may have different requirements [27], and the production scheduling may also have conflicting objectives such as minimizing Fmax and Fmean [17]. In addition, the DFJSS tasks with different objectives might be related to each other such as Fmax and Tmax, since these two objectives aim to finish each job as soon as possible. This is naturally a multitask multiobjective scheduling problem. This article takes DFJSS as a case study of multitask multiobjective GP. This is the first work of multitask multiobjective on dynamic combinatorial optimization. The calculations of these objectives are shown as follows.

1) *Fmax:* $\max_{j=1}^{n}(C_j - r_j)$.
2) *Fmean:* $([\sum_{j=1}^{n}(C_j - r_j)]/n)$.
3) *Tmax:* $\max_{j=1}^{n}\max\{0, C_j - d_j\}$.
4) *Tmean:* $([\sum_{j=1}^{n}\max\{0, C_j - d_j\}]/n)$.

Here, C_j represents the completion time of a job J_j, r_j represents the release time of J_j, d_j represents the due date of J_j, and n represents the number of jobs.

C. Automated Scheduling Heuristics Learning for DFJSS

1) Heuristics Versus Hyperheuristics: Heuristics search methods such as genetic algorithms [28] have been successfully used to find good solutions for large-scale problems. However, it is hard for them to react to dynamic events in time since they face rescheduling problems. In addition, the solutions found by heuristics search methods are often restricted to specific contexts, tasks, and system configurations. *Scheduling heuristics*, such as dispatching rules, are the most popularly used methods for constructing schedules in dynamic scheduling. Scheduling heuristics can be regarded as priority functions for prioritizing machines or operations to make routing or sequencing decisions. Most scheduling heuristics are *manually* designed with domain expertise. However, manually designing effective scheduling heuristics is difficult, time consuming, or even impossible. This is because the experts cannot identify all the subtle and interrelated conditions between attributes to design such heuristics in complicated dynamic environments. *Hyperheuristic methods* aim to select or generate heuristics, that is, heuristic selection and heuristic generation, for handling complex computationally expensive problems. The output of heuristic selection methods is a sequence of heuristics and decide which heuristic to use in specific situations, while the output of heuristic generation methods is a comprehensive heuristic to cover different situations [29]. GP has been widely used as a heuristic generation approach to learning scheduling heuristics because of its flexible representation [18], [30], [31], [32].

Algorithm 1: Framework of GP for DFJSS

```
   Input  : A DFJSS task with jobs and machines
   Output: The best learned scheduling heuristic for the task
1  Initialisation: Randomly initialise GP population
2  set h* ← null, fitness_h* ← +∞, t ← 0
3  while t < maxGen do
4      // Evaluation: Assign fitness for all GP individuals
5      for i = 1 to popsize do
6          Run a DFJSS instance with h_i to get the schedule S_i
7          fitness_hi ← Obj(S_i)
8      end
9      for i = 1 to popsize do
10         if fitness_hi < fitness_h* then
11             h*_i ← h_i
12         end
13     end
14     if t < maxGen − 1 then
15         Parent selection: select individuals for producing offspring
16         Evolution: Produce offspring with genetic operators
17         Rotate training instance with a new random seed
18     end
19     t ← t + 1
20 end
21 return h*
```

2) Genetic Programming: GP mimics the evolutionary process in nature to improve the offspring generation by generation, and it has four main steps (i.e., initialization, evaluation, parent selection, and evolution) as shown in Algorithm 1. GP starts with a number of randomly initialized individuals, that is, *popsize* individuals, where h_i represents ith individual, and fitness_{h_i} indicates the fitness of h_i obtained from the objective value $\text{Obj}(S_i)$ of h_i on a simulation S_i. The quality of each GP individual is measured with DFJSS instances during the evaluation. If the stopping criterion is not met (i.e., line 3, the current generation number t is smaller than the maximal number of generations *maxGen*), parent selection is conducted to select individuals with good fitness to produce new offspring by the genetic operators (i.e., reproduction, crossover and mutation). Otherwise, the best-found scheduling heuristic h^* is reported as the output of the GP algorithm for the DFJSS problem to be handled.

Fig. 1 shows an example of a GP individual with a routing rule and a sequencing rule for DFJSS. The rules consist of terminals and functions. Terminals are extracted based on the features of jobs, machines, and the job-shop system. The functions are basic arithmetic operators. The rules can be regarded as priority functions to prioritize machines or operations. For example, the routing rule can be interpreted as

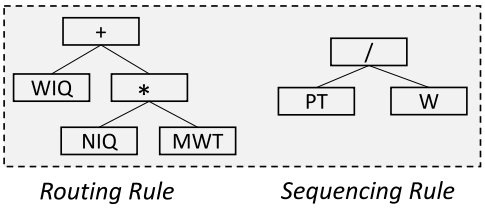


Fig. 1. Example of a GP individual with a routing and sequencing rule for DFJSS.

WIQ + NIQ*MWT, where WIQ is the workload of a machine, NIQ is the number of machines' operations, and MWT is the needed time for a machine to finish its processing operation.

D. Related Work

Evolutionary multitask multiobjective was first established in [3], and has been extended in a number of different aspects. Some multitask multiobjective related studies aim to improve positive knowledge transfer between tasks by selecting the most proper task to learn from [33] and [34]. The results show that selecting a proper assisted task can help improve the effectiveness of multitask learning. Dynamic resource allocation was investigated for giving different computational resources for tasks based on the characteristics of involved tasks [35]. The results show that this approach is suitable for multitask scenarios with different converged speeds. Last but not least, the adaptive transfer ratio was studied to control the knowledge sharing between tasks [36]. The results show that the adaptive transfer ratio is a good way to control when to learn from other tasks. However, these kinds of studies mainly work on numeric continuous optimization with target optimal solutions. The research of multitask multiobjective for discrete combinatorial optimization is still limited.

There are a few studies of multitask multiobjective on combinatorial optimization, such as routing [37], [38], [39], [40], [41] and exam timetabling [42]. A unified representation scheme was introduced to evolutionary multitask learning on combinatorial optimization, such as the travel salesman problem and job-shop scheduling problem [43]. An autoencoding-based multitask algorithm was developed to transfer knowledge across vehicle routing tasks explicitly [39]. However, all of them work on static combinatorial optimization, and the investigated problems are single-objective optimization. Among the limited multitask multiobjective methods on combinatorial optimization, multitask GP methods were first introduced to handle dynamic JSS problems in [18], [24], and [44]. However, the proposed algorithms are for single-objective optimization only rather than multiobjective optimization.

III. Automated Scheduling Heuristics Learning via Multitask Multiobjective GP

It is nontrivial to handle multitask multiobjective DFJSS with GP. This section starts with adapting algorithms based on traditional multitask multiobjective algorithms, followed by the proposed multitask multiobjective GP algorithms.

A. Adapted Multitask Multiobjective Algorithms

Technically, the classical multitask multiobjective algorithm MOMF can be applied to GP directly. However, the fitness of individuals across generations is incomparable due to the training instance rotation for GP in DFJSS, which makes a difference at the step of concatenating individuals between the population and newly generated offspring population. It is noted that rotating training instances and only using one instance at each generation is a typically strategy for GP in dynamic job-shop scheduling to learn scheduling heuristics with good generalization efficiently [45].

Fig. 2 shows the framework of adapted one population-based multitask multiobjective GP algorithms for DFJSS. The algorithms start with a randomly initialized population *Pop* which contains a number of individuals. Each individual is associated with a skill factor that indicates its corresponding task to optimize. The individuals are evaluated on their corresponding tasks only. If the stopping criterion is met, the best individuals, that is, scheduling heuristics, will be recorded for corresponding tasks. Otherwise, offspring are produced with the parent selection method via genetic operators (i.e., crossover, mutation, and reproduction) to generate a new population *newPop*, and their skill factors are determined by assortative mating and vertical cultural transmission strategies. The individuals from the previous generation *Pop* and newly generated individuals *newPop* are emerged, and then ranked based on their front ranks and crowding distance on their corresponding tasks separately. Their skill fitness are set as their rank value. The top-ranked *popsize* individuals based on scalar fitness are moved to the next generation.

We develop three multitask multiobjective GP algorithms based on three possible ways to handle the incomparable fitness issue of the combination of the population and newly generated population.

1) A possible way is to use the same training instance at each generation. In this way, the fitness across different generations are comparable. The traditional multitask multiobjective algorithm [3] can be adapted to GP for DFJSS directly. However, it may lose the effectiveness of learned scheduling heuristics.
2) A second way is to rotate the training instance at every generation, and reevaluate the individuals from the population with the new training instance at the current generation. The goal is to make the individuals from the previous generation comparable to the individuals in the current (offspring) population. This is unique for this way only, which is marked by a dotted rectangle in Fig. 2.
3) Except for these two ways, we also rotate the training instance but do not reevaluate the individuals from the parent population, and the individuals are compared based on incomparable fitness. Note that we do not expect this algorithm to obtain good performance, and this is just to verify the previous two adapted ideas.

B. Proposed Multitask Multiobjective Algorithms

Searching in heuristic space is more challenging than searching in solution space, since a small change on heuristics

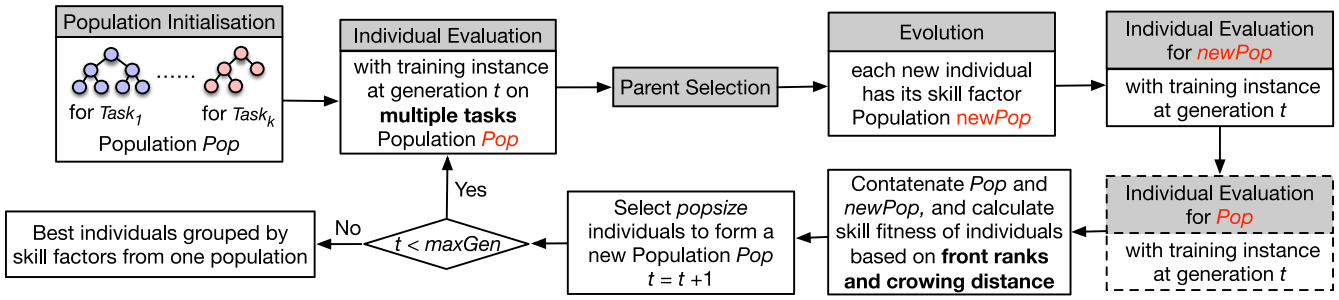


Fig. 2. Flowchart of the adapted multitask multiobjective GP algorithms under the one population framework, where *Pop* and *newPop* are populations, *t* is the current generation, and *maxGen* is the maximal number of generations.

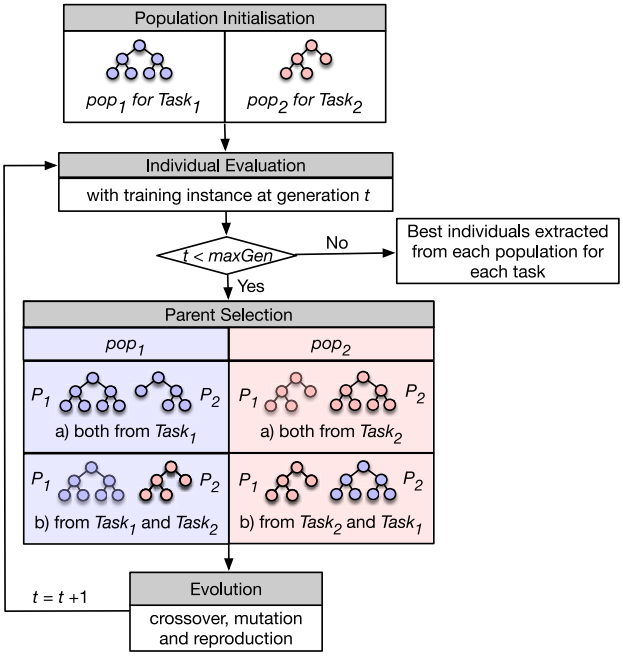


Fig. 3. Flowchart of the proposed multipopulation-based multitask multiobjective GP (e.g., taking two tasks as example), where pop_1 and pop_2 are populations, *t* is the current generation, *maxGen* is the maximal number of generations, and P_1 and P_2 are selected parents.

can make a big difference on solutions. Intuitively, allocating heuristics to different tasks via a skill factor is too stochastic for heuristics to obtain good performance on allocated multiobjective DFJSS tasks. To handle this issue, this article proposes to use multipopulation-framework-based multitask multiobjective GP algorithms for handling multiple multiobjective DFJSS simultaneously. The individuals in each population are always used for optimizing one task to maintain the quality of scheduling heuristics for the corresponding DFJSS task. The knowledge sharing between tasks is realized implicitly by crossover along with the individuals from different tasks. The details of the proposed algorithms are shown as follows.

1) Multipopulation-Based Multitask Multiobjective GP: Assuming there are two tasks, Fig. 3 shows the flowchart of the proposed multipopulation-based multitask multiobjective GP. The proposed algorithm starts with initializing two populations (i.e., pop_1 and pop_2, each for a task) with a number of individuals (i.e., scheduling heuristics) at the initialization stage. All individuals will be evaluated with a DFJSS instance to obtain their fitness at the individual evaluation stage. It is noted that in traditional GP, all the individuals in a population are normally evaluated with the same instance, while individuals in different subpopulations are evaluated with different instances representing different tasks. After we know the quality of individuals, we will select parents to produce offspring along with GP genetic operators (i.e., crossover, mutation, and reproduction). This article chooses to share knowledge between tasks implicitly via the crossover operator. Fig. 3 only shows the parent selection for crossover, because GP relies heavily on crossover as its primary genetic operator. The produced two offspring by a crossover operator are either with the parents from the same population or one of the parents is from another population. For mutation and reproduction, only one parent is needed, and the parent is from the corresponding population. If the stopping criterion is not met, the proposed GP algorithm will go to the next generation. Otherwise, we will return the best learned Pareto front from each population for each task.

2) Task-Oriented Multitask Multiobjective GP: The classical crossover operator produces two offspring by swapping two subtrees between parents. One of the two offspring derives the main genetic materials from one parent, and the other inherits more important genetic materials from the other parent [20]. Intuitively, the offspring that contains the main generic materials for one task is more likely to be good for that task. Based on the proposed multipopulation-based multitask multiobjective GP algorithm, we develop a task-oriented multitask multiobjective GP algorithm that only keeps the offspring that contains the main materials for that task. This design is based on our preliminary finding [20] that the subtrees which are closer to root are more important for a GP individual. This is expected to improve the effectiveness of scheduling heuristics on tasks by controlling a proper learning degree. Fig. 4 shows an example of how to produce offspring from two parents from different populations which are distinguished with different colors. Assuming we generate offspring for Task_1, a subtree of each parent (i.e., Parent_1 and Parent_2) is selected and swapped to generate two offspring. Only the Offspring_1 that contain the main information (e.g., the subtree in the dashed circle) for Task_1 is kept for optimizing $Task_1$.

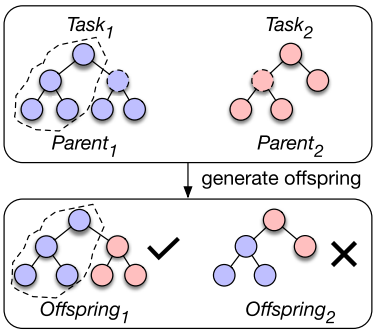


Fig. 4. Example of producing scheduling heuristic based on the proposed task-oriented knowledge sharing via crossover for multitask multiobjective GP on DFJSS.

Different from the traditional GP that selects parents from individuals for the same task, the proposed algorithm can also select parents from individuals for different tasks. Furthermore, the proposed task-oriented knowledge sharing via crossover expects to not only help share knowledge between tasks but also maintain the main genetic materials for specific tasks. Thus, it expects to benefit the performance improvement.

C. Summary

To automatically learn scheduling heuristics for multiple DFJSS tasks simultaneously, different variations of adapted multitask multiobjective GP algorithms are carefully designed. This aims to investigate the effectiveness of the traditional multitask multiobjective approach on GP for DFJSS. Except for designing multitask multiobjective GP based on a single population framework, this article also develops multipopulation-based multitask multiobjective GP algorithms by considering the difference between heuristic search space and solution search space. Although previous studies normally treat one population framework and multipopulation framework as the same for multitask learning [46], the effectiveness difference between these two frameworks on heuristic search space may differ. The effectiveness of adapted algorithms based on one population and proposed algorithms based on multipopulation will be compared comprehensively.

IV. Experiment Design

A. Simulation Model

Refer to widely used DFJSS instances [21], [47], [48], 5000 jobs are assumed to be processed by ten machines. According to a Poisson process, new jobs will arrive over time at the rate of λ. Each job consists of a random number of operations from a uniform discrete distribution between 1 and 10. An operation has a random number of candidate machines following a uniform discrete distribution between 1 and 10. A uniform discrete distribution with the range [1, 99] is used to sample the processing time of operations. 1.5 times of its processing time is the main factor for assigning due date for a job. The importance (represented by weights) of jobs are set as 1 (20%), 2 (60%), and 4 (20%), respectively [49]. The training instance at each generation is unique by assigning a new random seed for the DFJSS simulation [50].

This article uses utilization-level parameter (p) to simulate job-shop scenarios with different characteristics [51]. It is defined as the proportion of time that a machine is expected to be busy. The utilization level is adjusted by the λ in Poisson process. Equation (1) shows its calculation, where μ denotes the machines' average processing time, and P_M represents the probability that a machine is visited by a job. For instance, P_M will be 2/10, if each job has two operations. A larger utilization level value indicates a busier job shop

$$\lambda = \mu * P_M/p. \tag{1}$$

The first 1000 jobs are considered warm-up jobs and are excluded from the objective calculations in order to estimate steady-state performance. The next 5000 jobs are collected to calculate the fitness values. The simulation ends when the 6000th job is completed.

B. Design of Comparisons

For the adapted algorithms with one population, the multitask multiobjective MOMF with fixed training instances is called MOMFGP$^{\text{fixed}}$. The one with training instance rotation and individual reevaluation is called MOMFGP$^{\text{r}+}$, while the one with training instance rotation but no individual reevaluation is called MOMFGP$^{\text{r}-}$. For the proposed algorithms with multipopulation, the algorithm that handles multiple multiobjective tasks separately is called GP. The proposed multitask *multiobjective GP* based on *multipopulation* is called MOP_GP, while the MOP_GP with task-oriented knowledge-sharing strategy is called MOPO_GP.

First, MOMFGP$^{\text{fixed}}$, MOMFGP$^{\text{r}+}$, and MOMFGP$^{\text{r}-}$ with random selection or tournament selection are compared separately to investigate which parent selection method is good for MOMFGP on DFJSS. Second, MOMFGP$^{\text{fixed}}$, MOMFGP$^{\text{r}+}$, and MOMFGP$^{\text{r}-}$ will be compared to find the best adapted MOMFGP algorithm for DFJSS. Third, we compare the best adapted MOMFGP with GP to investigate which evolutionary framework (i.e., one population or multipopulation) is good for multitask multiobjective GP on DFJSS. Then, we compare the multipopulation-based algorithms, that is, GP, MOP_GP, and MOPO_GP, to verify the effectiveness of the proposed multipopulation-based multitask multiobjective GP algorithms. Furthermore, we compare MOP_GP and MOPO_GP to verify the effectiveness of proposed task-oriented strategy.

Two pairs of conflicting objectives (Fmax and Fmean, and Tmax and Tmean) [17], [18] are formed as multitask multiobjective scenarios along with the utilization level (i.e., 0.75, 0.85, and 0.95) to verify the effectiveness of the algorithms.

C. Parameter Settings

The terminals of GP [44] are extracted from the features of the job shop. The terminals are either related to the features of jobs (i.e., WKR, NOR, W, and TIS), operations (i.e., PT, NPT, and OWT), and machines (i.e., NIQ, WIQ, and MWT). Table I shows the details of the terminals. It is noted that the relative

TABLE I
TERMINAL SET OF GP IN DFJSS

Notation	Description
WKR	Median amount of work remaining for a job
NOR	The number of operations remaining for a job
W	Weight (importance) of a job
TIS	Time that a job has been in the job shop
PT	Operation processing time on a machine
NPT	Median processing time for the next operation
OWT	The waiting time of an operation
NIQ	The number of operations in a machine's queue
WIQ	The workload of a machine, i.e., total needed processing time
MWT	The needed time to finish the current operation of a machine

TABLE II
PARAMETER SETTINGS IN GP

Ref.	Parameter	Value
[18]	*Population size	800
	*Number of elites	10
	**Population size	800
	***Number of populations	2
	***Population size	400
	***Number of elites in each population	5
	***Transfer ratio	0.6
	Parent selection method	Tournament selection (size 5)
[19]	Crossover / Mutation / Reproduction rate	80% / 15% / 5%
	Population initialisation method	ramped-half-and-half
	Initial minimum / maximum depth	2 / 6
	Maximal program depth	8
	Terminal / non-terminal selection rate	10% / 90%
	The maximal number of generations	51

* is for MOMFGP^{fixed} and MOMFGP^{r-} (one population)
** is for MOMFGP^{r+} (one population)
*** is for GP, MOP_GP and MOPO_GP (multi-population)

due-date (RDD) feature has been considered implicitly for tardiness related tasks in this article, since due-date information of jobs can be reflected by TIS and WKR, that is, RDD = (TIS + WKR) * DDFactor − TIS, where DDFactor is a constant. Following the suggestion in [44], the function set consists of {+, −, ∗, /, max, min}, and each function has two arguments. The protected division "/" is used, which returns one if the denominator equals zero. Table II shows other parameter settings of GP algorithms as suggested in [18] and [19]. For example, following the suggestions in [18], we set the population size for each task to 400, and the transfer ratio to 0.6, which have shown their effectiveness in multitask GP for DFJSS. Other parameters, such as the population initialization method and the number of generations, are commonly used parameter settings in GP [19].

V. RESULTS AND DISCUSSION

Along with 30 independent runs, Friedman's test and Wilcoxon rank-sum test with a significance level of 0.05 are used to examine the performance of algorithms. In the following results, "↑," "↓," and "≈" indicate the results are statistically significantly better than, worse than, or similar to its counterpart. "Win, Draw, Lose" is the number of tasks that the proposed algorithm is statistically better, similar, or worse than a compared algorithm. We compare each algorithm with its predecessor(s) one at a time. We use hyper volume (HV) [52] and inverted generational distance (IGD) [53] to examine the proposed algorithms. A larger (smaller) HV (IGD) value signifies better performance.

A. Tournament Selection Versus Random Selection

Random selection and tournament selection are suggested for multitask learning [1], [3] without providing justifications. Thus, it is not clear which one is good for multitask multiobjective DFJSS with GP. This will be first investigated here.

Table III shows the mean and deviations of the HV and IGD values on test instances of $\mathrm{MOMFGP}^{\mathrm{fixed}}$, $\mathrm{MOMFGP}^{\mathrm{r}+}$, and $\mathrm{MOMFGP}^{\mathrm{r}-}$ over 30 independent runs in six tasks. In terms of HV, $\mathrm{MOMFGP}^{\mathrm{fixed}}$ and $\mathrm{MOMFGP}^{\mathrm{r}+}$ with tournament selection achieve a similar performance in four out of six tasks, and obtain significantly better performance than the ones with random selection in two out of six tasks. In addition, $\mathrm{MOMFGP}^{\mathrm{r}-}$ with tournament selection shows its superiority in most of the tasks. This indicates that using the tournament selection is a good choice for multitask multiobjective GP algorithms. The superiority of tournament selection is more obvious on $\mathrm{MOMFGP}^{\mathrm{r}-}$ compared with $\mathrm{MOMFGP}^{\mathrm{fixed}}$ and $\mathrm{MOMFGP}^{\mathrm{r}+}$. We know that the evolutionary process of $\mathrm{MOMFGP}^{\mathrm{r}-}$ is quite random, since the individual fitness of $\mathrm{MOMFGP}^{\mathrm{r}-}$ are not presentable for the quality of individuals due to the incomparable fitness across generations. Although not all the individuals have comparable fitness, we can see that using tournament selection is better than random selection (at least) to identify good individuals as parents for producing offspring. We have the same observation on the IGD values of the algorithms with random and tournament selection. Therefore, we choose to use tournament selection in multitask multiobjective GP algorithms. In the following sections, tournament selection is used for all algorithms.

B. Comparison Among Variations of MOMFGP

In this section, we investigate the effectiveness of the three multitask multiobjective GP algorithms designed by adapting the classical multitask multiobjective algorithm. It is noted that the training instance at the last generation of $\mathrm{MOMFGP}^{\mathrm{fixed}}$ is different from $\mathrm{MOMFGP}^{\mathrm{r}+}$ and $\mathrm{MOMFGP}^{\mathrm{r}-}$. Therefore, the HV and IGD of the algorithms on training instances are not comparable. We will use the results on test instances for this purpose. Table IV shows the mean and standard deviations of the HV and IGD on test instances of $\mathrm{MOMFGP}^{\mathrm{r}-}$, $\mathrm{MOMFGP}^{\mathrm{fixed}}$, and $\mathrm{MOMFGP}^{\mathrm{r}+}$ according to 30 independent runs in six tasks. We can see that $\mathrm{MOMFGP}^{\mathrm{r}+}$ performs the best among them. Therefore, $\mathrm{MOMFGP}^{\mathrm{r}+}$ will be used as the baseline of the multitask multiobjective GP algorithm. This is different from the finding in [18] which is a work on multitask GP on single-objective optimization, where $\mathrm{MOMFGP}^{\mathrm{fixed}}$ is slightly better than $\mathrm{MOMFGP}^{\mathrm{r}+}$. One possible reason is that multiobjective optimization has a higher requirement on the generalization of the learned scheduling heuristics than single-objective optimization because of the reflection of multiple objectives, which requires to use of training instance rotation. As we can see, it is necessary to consider the different

TABLE III
MEAN (STANDARD DEVIATION) OF THE HV AND IGD ON *Test Instances* OF MOMFGP$^{\text{fixed}}$, MOMFGP$^{\text{r}+}$, AND MOMFGP$^{\text{r}-}$ OVER 30 INDEPENDENT RUNS IN SIX TASKS WHICH ARE REPRESENTED BY THE OBJECTIVE AND UTILIZATION LEVEL

		HV					
		MOMFGP^{fixed}		MOMFGP^{r+}		MOMFGP^{r-}	
Scenario	Tasks	Random	Tournament	Random	Tournament	Random	Tournament
1	<Fmax-Fmean, 0.75>	0.25(0.14)	0.32(0.22)(≈)	0.37(0.18)	0.51(0.22)(↑)	0.15(0.09)	0.19(0.10)(≈)
	<Tmax-Tmean, 0.75>	0.25(0.13)	0.36(0.23)(≈)	0.16(0.19)	0.30(0.24)(↑)	0.23(0.18)	0.31(0.20)(≈)
2	<Fmax-Fmean, 0.85>	0.31(0.15)	0.42(0.19)(↑)	0.36(0.18)	0.34(0.14)(≈)	0.23(0.14)	0.39(0.16)(↑)
	<Tmax-Tmean, 0.85>	0.30(0.14)	0.28(0.19)(≈)	0.31(0.22)	0.28(0.19)(≈)	0.21(0.14)	0.31(0.16)(↑)
3	<Fmax-Fmean, 0.95>	0.32(0.16)	0.39(0.20)(≈)	0.31(0.17)	0.35(0.16)(≈)	0.12(0.09)	0.34(0.23)(↑)
	<Tmax-Tmean, 0.95>	0.27(0.16)	0.40(0.23)(↑)	0.32(0.19)	0.39(0.21)(≈)	0.20(0.18)	0.32(0.17)(↑)
		IGD					
1	<Fmax-Fmean, 0.75>	0.45(0.12)	0.43(0.14)(≈)	0.41(0.09)	0.39(0.13)(≈)	0.45(0.11)	0.43(0.11)(≈)
	<Tmax-Tmean, 0.75>	0.54(0.14)	0.48(0.21)(≈)	0.88(0.25)	0.70(0.29)(↑)	0.58(0.21)	0.49(0.19)(≈)
2	<Fmax-Fmean, 0.85>	0.50(0.16)	0.50(0.09)(≈)	0.45(0.13)	0.47(0.12)(≈)	0.43(0.18)	0.33(0.11)(↑)
	<Tmax-Tmean, 0.85>	0.53(0.14)	0.57(0.14)(≈)	0.51(0.22)	0.54(0.21)(≈)	0.55(0.18)	0.42(0.16)(↑)
3	<Fmax-Fmean, 0.95>	0.53(0.17)	0.45(0.14)(≈)	0.52(0.17)	0.51(0.16)(≈)	0.67(0.16)	0.51(0.21)(↑)
	<Tmax-Tmean, 0.95>	0.54(0.21)	0.41(0.22)(↑)	0.52(0.22)	0.46(0.24)(≈)	0.58(0.18)	0.45(0.22)(↑)

TABLE IV
MEAN (STANDARD DEVIATION) OF THE HV AND IGD ON *Test Instances* OF MOMFGP$^{\text{r}-}$, MOMFGP$^{\text{fixed}}$, AND MOMFGP$^{\text{r}+}$ ACCORDING TO 30 INDEPENDENT RUNS IN SIX TASKS WHICH ARE REPRESENTED BY THE OBJECTIVE AND UTILIZATION LEVEL

		HV			IGD		
Scenario	Tasks	MOMFGP^{r-}	MOMFGP^{fixed}	MOMFGP^{r+}	MOMFGP^{r-}	MOMFGP^{fixed}	MOMFGP^{r+}
1	<Fmax-Fmean, 0.75>	0.33(0.18)	0.40(0.20)(≈)	0.60(0.19)(↑)(↑)	0.42(0.16)	0.38(0.16)(≈)	0.30(0.14)(↑)(↑)
	<Tmax-Tmean, 0.75>	0.40(0.10)	0.35(0.15)(≈)	0.49(0.17)(↑)(↑)	0.46(0.11)	0.53(0.18)(≈)	0.39(0.16)(↑)(↑)
2	<Fmax-Fmean, 0.85>	0.35(0.15)	0.41(0.17)(≈)	0.35(0.15)(≈)(≈)	0.47(0.11)	0.45(0.11)(≈)	0.43(0.11)(≈)(↑)
	<Tmax-Tmean, 0.85>	0.46(0.21)	0.49(0.21)(≈)	0.50(0.16)(≈)(≈)	0.39(0.17)	0.36(0.15)(≈)	0.34(0.12)(≈)(≈)
3	<Fmax-Fmean, 0.95>	0.48(0.24)	0.64(0.13)(↑)	0.59(0.16)(≈) (≈)	0.36(0.24)	0.22(0.10)(↑)	0.27(0.13)(≈)(≈)
	<Tmax-Tmean, 0.95>	0.72(0.18)	0.81(0.14)(↑)	0.80(0.12)(≈)(≈)	0.20(0.18)	0.13(0.13)(↑)	0.15(0.11)(≈)(≈)
	Win / Draw / Lose	0 / 4 / 2	0 / 4 / 2	N/A	0 / 4 / 2	1 / 3 / 2	N/A
	Average Rank	2.28	1.94	**1.78**	2.22	1.93	**1.86**

characteristics of problems when designing algorithms. For convenience, MOMFGP is used to indicate the best adapted multitask multiobjective GP algorithm, which will also serve as a baseline algorithm for comparison in the following sections.

C. GP Versus MOMFGP

In this section, we will investigate the effectiveness of one population and multipopulation-based multitask multiobjective GP algorithms by comparing MOMFGP with GP. Figs. 5 and 6 show the violin plot of the HV and IGD of GP and MOMFGP over 30 independent runs in three multitask multiobjective scenarios. Fig. 5 shows that GP is significantly better than MOMFGP in all tasks with larger HV values. All the IGD values achieved by GP are smaller than MOMFGP, which indicates that GP performs better than MOMFGP. This finding is also different from what we can see from [18], where the adaption of classical multitask algorithm can achieve similar performance as GP. This indicates that using the skill factor to assign individuals to tasks is ineffective for multiobjective GP in DFJSS, and using multipopulation is a good choice in this case. One possible reason is that changing an individual

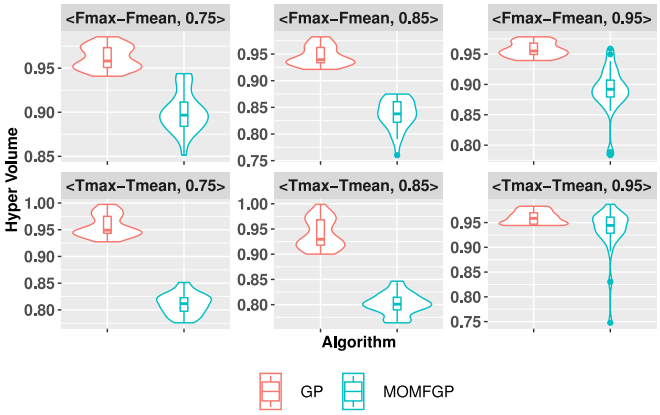


Fig. 5. Violin plots of the **HV** on training instances of GP and MOMFGP according to 30 independent runs in three multiobjective multitask scenarios (each column corresponds to a multitask scenario).

to optimize another multiobjective DFJSS task can disrupt its performance significantly. This also shows the challenge of learning scheduling heuristics by multitask multiobjective GP for DFJSS in heuristic search spaces.

TABLE V
MEAN (STANDARD DEVIATION) OF THE HV AND IGD ON *Training Instances and Test Instances* OF GP, MOP_GP, AND MOPO_GP BASED ON 30 INDEPENDENT RUNS IN SIX TASKS WHICH ARE REPRESENTED BY THE OBJECTIVE AND UTILIZATION LEVEL

		HV			**IGD**		
		Training					
Scenario	Tasks	GP	MOP_GP	MOPO_GP	GP	MOP_GP	MOPO_GP
1	<Fmax-Fmean, 0.75>	0.74(0.07)	0.79(0.07)(↑)	0.83(0.06)(↑)(↑)	0.12(0.04)	0.09(0.04)(↑)	0.07(0.03)(↑)(↑)
	<Tmax-Tmean, 0.75>	0.68(0.18)	0.79(0.13)(↑)	0.83(0.09)(↑)(≈)	0.24(0.13)	0.16(0.09)(↑)	0.13(0.06)(↑)(≈)
2	<Fmax-Fmean, 0.85>	0.87(0.06)	0.87(0.05)(≈)	0.89(0.05)(≈)(≈)	0.06(0.03)	0.06(0.03)(≈)	0.05(0.02)(≈)(≈)
	<Tmax-Tmean, 0.85>	0.86(0.09)	0.85(0.08)(≈)	0.87(0.08)(≈)(≈)	0.08(0.05)	0.09(0.04)(≈)	0.08(0.04)(≈)(↑)
3	<Fmax-Fmean, 0.95>	0.87(0.03)	0.89(0.02)(↑)	0.89(0.02)(↑)(≈)	0.06(0.01)	0.04(0.01)(↑)	0.05(0.01)(↑)(≈)
	<Tmax-Tmean, 0.95>	0.85(0.04)	0.88(0.02)(↑)	0.88(0.02)(↑)(≈)	0.06(0.02)	0.04(0.01)(↑)	0.05(0.01)(↑)(≈)
	Win / Draw / Lose	0 / 2 / 4	0 / 5 / 1	N/A	0 / 2 / 4	0 / 4 / 2	N/A
	Average Rank	2.32	1.93	**1.76**	2.41	1.87	**1.72**
		Test					
1	<Fmax-Fmean, 0.75>	0.76(0.08)	0.81(0.08)(↑)	0.86(0.07)(↑)(↑)	0.13(0.04)	0.10(0.04)(↑)	0.08(0.03)(↑)(↑)
	<Tmax-Tmean, 0.75>	0.61(0.17)	0.72(0.15)(↑)	0.78(0.09)(↑)(≈)	0.26(0.16)	0.16(0.13)(↑)	0.12(0.06)(↑)(≈)
2	<Fmax-Fmean, 0.85>	0.83(0.07)	0.84(0.06)(≈)	0.86(0.05)(≈)(≈)	0.08(0.04)	0.07(0.04)(≈)	0.06(0.03)(≈)(≈)
	<Tmax-Tmean, 0.85>	0.80(0.13)	0.78(0.12)(≈)	0.82(0.11)(≈)(↑)	0.13(0.09)	0.13(0.09)(≈)	0.11(0.08)(≈)(≈)
3	<Fmax-Fmean, 0.95>	0.86(0.04)	0.89(0.03)(↑)	0.88(0.03)(↑)(≈)	0.06(0.02)	0.05(0.01)(≈)	0.05(0.02)(≈)(≈)
	<Tmax-Tmean, 0.95>	0.83(0.06)	0.86(0.04)(↑)	0.86(0.04)(↑)(≈)	0.08(0.03)	0.06(0.02)(↑)	0.06(0.02)(↑)(≈)
	Win / Draw / Lose	0 / 2 / 4	0 / 4 / 2	N/A	0 / 3 / 3	0 / 5 / 1	N/A
	Average Rank	2.27	1.97	**1.76**	2.31	1.93	**1.77**

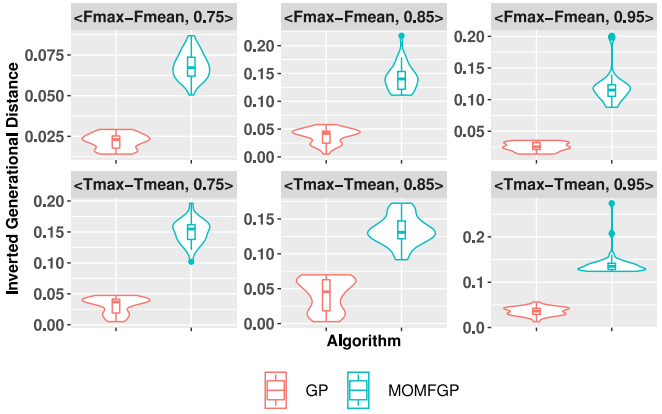


Fig. 6. Violin plots of the **IGD** on training instances of GP and MOMFGP based on 30 independent runs in three multiobjective multitask scenarios (each column corresponds to a multitask scenario).

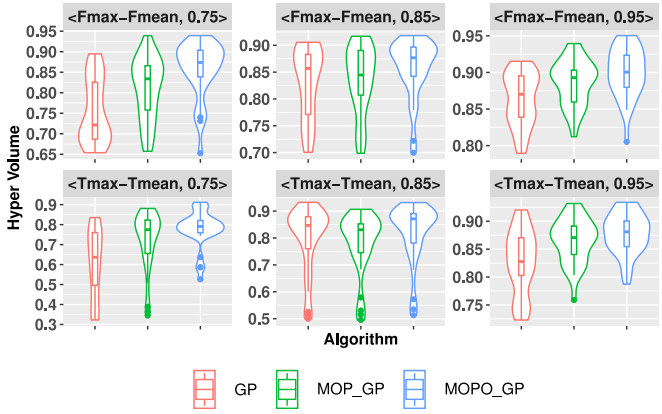


Fig. 7. Violin plots of the HV of GP, MOP_GP, and MOPO_GP on *test instances* of six tasks (each column corresponds to a multitask scenario).

D. Comparison Among Proposed Algorithms

To compare the effectiveness of GP, MOP_GP, and MOPO_GP, Table V shows the mean and standard deviation of the HV and IGD on training and test instances based on 30 independent runs of six tasks. For training, the results show that MOP_GP performs better than GP in terms of HV and IGD for four out of six tasks. This verifies the effectiveness of the proposed multitask multiobjective GP algorithm based on a multipopulation framework and the knowledge sharing via crossover. In addition, MOPO_GP is the most effective algorithm which has the smallest average rank value. This verifies the effectiveness of the proposed multitask multiobjective GP algorithm with the task-oriented knowledge sharing.

Fig. 7 shows the violin plots of the HV values of GP, MOP_GP, and MOPO_GP of six tasks on test instances. We can see that MOP_GP and MOPO_GP obtain larger HV values than GP for all six tasks. In addition, overall, MOPO_GP obtains larger HV values compared with MOP_GP based on the HV distributions. Fig. 8 shows the violin plots of the IGD values of GP, MOP_GP, and MOPO_GP of six tasks on training instances. The results show that the proposed algorithms MOP_GP and MOPO_GP achieve a better IGD value, as the IGD values of MOP_GP and MOPO_GP are smaller than GP based on the IGD distributions. In addition, MOPO_GP is the best among them. This shows the effectiveness of the proposed multipopulation-based multitask multiobjective GP algorithms.

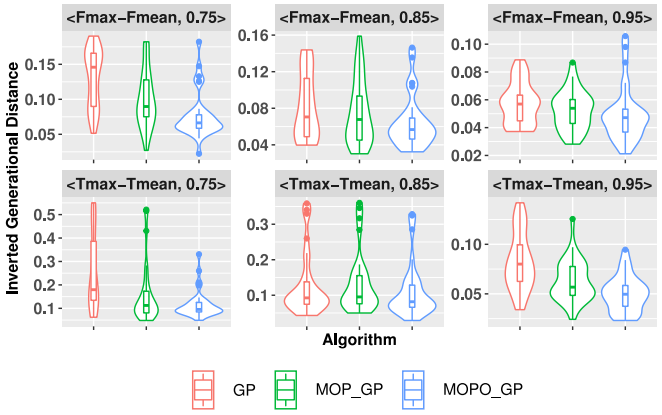


Fig. 8. Violin plots of the IGD of GP, MOP_GP, and MOPO_GP on *test instances* of six tasks (each column corresponds to a multitask scenario).

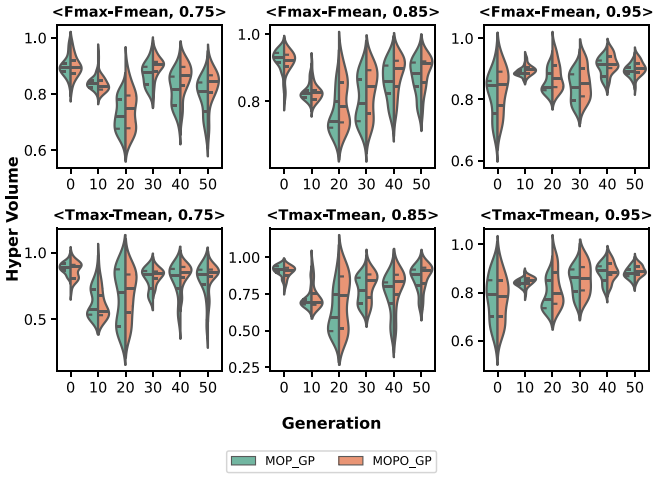


Fig. 9. Violin plots of the HV of MOP_GP and MOPO_GP on the training instances of six tasks at six generations, that is, generations 0, 10, 20, 30, 40, and 50 (each column corresponds to a multitask scenario).

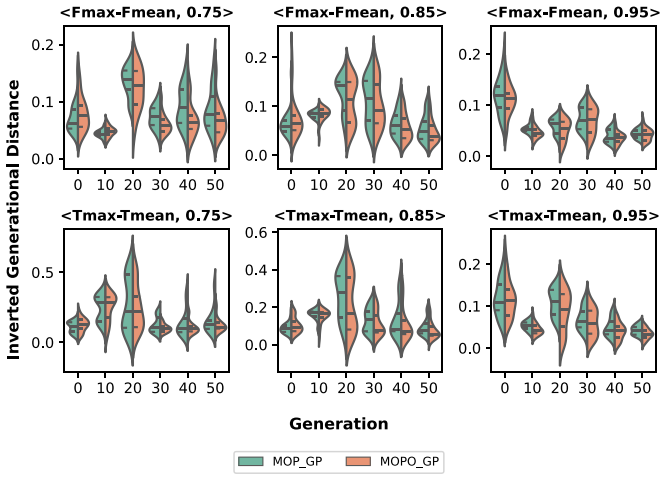


Fig. 10. Violin plots of the IGD of MOP_GP and MOPO_GP on the training instances of six tasks at six generations, that is, generations 0, 10, 20, 30, 40, and 50 (each column corresponds to a multitask scenario).

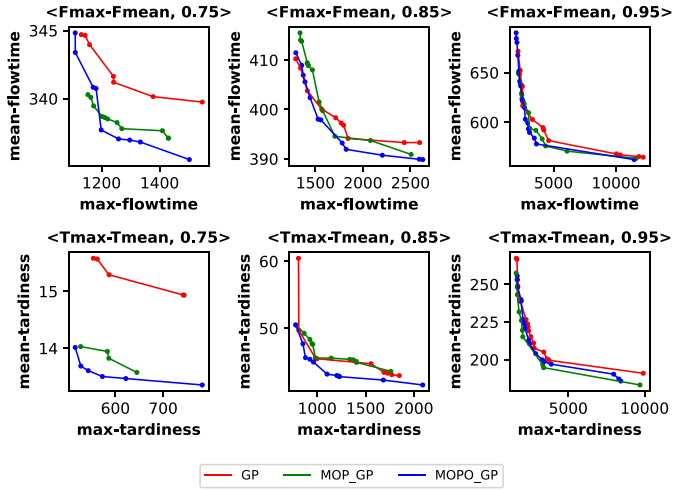


Fig. 11. Learned Pareto fronts of GP, MOP_GP, and MOPO_GP for six tasks (each column corresponds to a multitask scenario).

E. MOP_GP *Versus* MOPO_GP

Fig. 9 shows the violin plots of the HV values of MOP_GP and MOPO_GP at six generations (i.e., generations 0, 10, 20, 30, 40, and 50) during the training process. The results show that MOP_GP and MOPO_GP have similar performance at the very early stage such as generation 0 and generation 10. From generation 20, MOPO_GP starts to perform better than MOP_GP obviously. It is obvious that MOPO_GP performs significantly better than MOP_GP at the end of the evolutionary process as shown at generation 50, which verifies the effectiveness of proposed task-oriented knowledge-sharing strategy. This also indicates that it takes time to show the effectiveness of knowledge sharing between tasks in multitask multiobjective GP algorithms for DFJSS problems. This is consistent with our intuition because this article works in heuristic space rather than solution space, and the changes on heuristics are not directly reflected in solutions. We have the same findings in terms of IGD values along with generations, as shown in Fig. 10.

VI. Further Analysis

A. Learned Pareto Fronts

Fig. 11 shows the Pareto front obtained from the run with a medium HV value of GP, MOP_GP, and MOPO_GP. The results show that MOP_GP and MOPO_GP can learn better Pareto front than GP for most of the tasks. Furthermore, we find that MOPO_GP learns better scheduling heuristics that achieve smaller objective values on both objectives of all six multiobjective tasks. This indicates that the learned heuristics by MOPO_GP dominate the ones obtained by MOPO_GP, which verifies the effectiveness of MOPO_GP.

B. Diversity

We use entropy to measure the diversity of individuals for tasks. The entropy is calculated as $\text{entropy(task)} = -\sum_{c\in C}(|c|/|\text{inds}|)\log(|c|/|\text{inds}|)$, where C is the set of clusters obtained by the DBScan clustering algorithm [54] with the phenotypic distance measure [49] and a cluster radius of 0. A

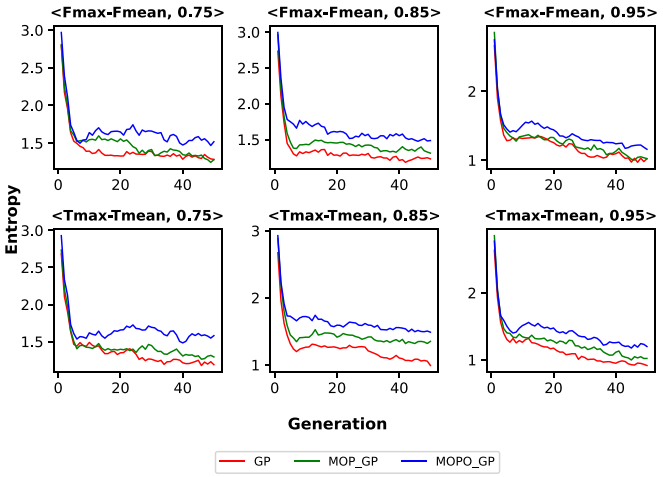


Fig. 12. Curves of entropy values of GP, MOP_GP, and MOPO_GP over 30 independent runs (each column corresponds to a multitask scenario).

larger entropy value indicates a higher diversity of individuals for a task.

Fig. 12 shows the entropy values of GP, MOP_GP, and MOPO_GP along with generations. The results show that the diversity of GP individuals for tasks reduces along with generations. The entropy decreases quickly at the early stage (around the first five generations), which is a common phenomenon for GP [19], [55]. In addition, before generation 5, the entropy values of GP, MOP_GP, and MOPO_GP are similar. However, after generation 5, the difference between them becomes clear. Compared with GP, MOP_GP has a higher diversity value, which indicates that the knowledge sharing between multiobjective tasks can help increase the diversity of all tasks in a multitask scenario. This is consistent with our intuition that it can bring new genetic materials by learning knowledge from other tasks. This can potentially increase the individuals' diversity for a task. MOPO_GP obtains the best diversity value among all three algorithms. This indicates that the proposed MOPO_GP with task-oriented knowledge sharing can maintain a better individual diversity for a particular task, which contributes to the effectiveness improvement of an algorithm.

C. Learned Scheduling Heuristics

This section investigates the learned scheduling heuristics for tasks in the most complex multitask scenario, that is, scenario 3 with <Fmax-Fmean, 0.95> and <Tmax-Tmean, 0.95>. Figs. 13 and 14 show the learned routing rules of task <Fmax-Fmean, 0.95> and task <Tmax-Tmean, 0.95>, respectively. We can see that there are a number of common building blocks between the two routing rules which are highlighted in blue. This indicates that the involved two multiobjective tasks are related and do manage to share knowledge with each other. In addition, each routing rule has its unique building blocks to make itself specific to its corresponding task. Figs. 15 and 16 show the learned sequencing rules of task <Fmax-Fmean, 0.95> and task <Tmax-Tmean, 0.95>, respectively. We have the same patterns/observations in the sequencing rules as found in the routing rules we just discussed.

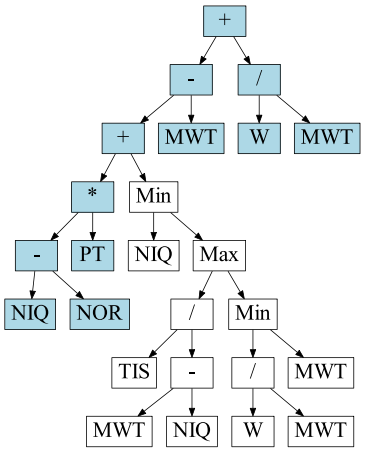


Fig. 13. Example of the best learned *routing rule* for <Fmax-Fmean, 0.95> by MOPO_GP in scenario 3.

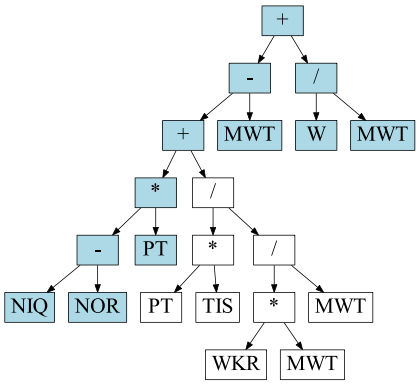


Fig. 14. Example of the best learned *routing rule* for <Tmax-Tmean, 0.95> by MOPO_GP in scenario 3.

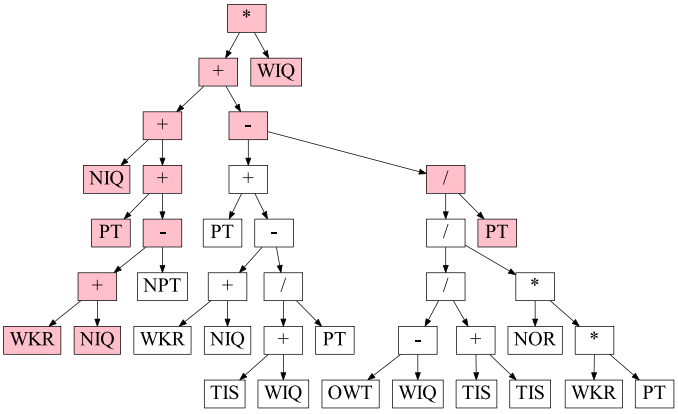


Fig. 15. Example of the best learned *sequencing rule* for <Fmax-Fmean, 0.95> by MOPO_GP in scenario 3.

In terms of the features for constructing the learned routing rules, Fig. 13 shows that MWT and NIQ are the top two used features of the routing rule for task <Fmax-Fmean, 0.95>, while Fig. 14 shows that MWT and PT are ranked as the top two features for task <Tmax-Tmean, 0.95>. We can see that

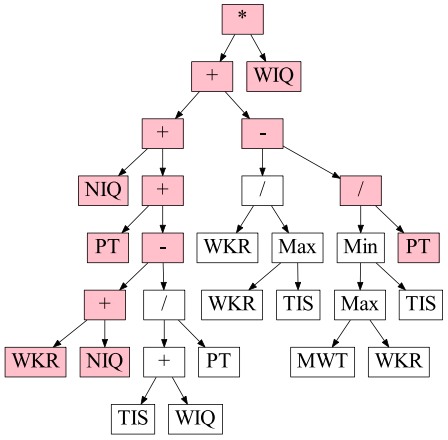


Fig. 16. Example of the best learned *sequencing rule* for <Tmax-Tmean, 0.95> by MOPO_GP in scenario 3.

TABLE VI
MEAN (STANDARD DEVIATION) OF TRAINING TIME (IN MINUTES) OF GP, MOP_GP, AND MOPO_GP ACCORDING TO 30 INDEPENDENT RUNS OF SIX TASKS

Scenario	Tasks	GP	MOP_GP	MOPO_GP
1	<Fmax-Fmean, 0.75> <Tmax-Tmean, 0.75>	90(18)	85(17)(≈)	89(16)(≈)(≈)
2	<Fmax-Fmean, 0.85> <Tmax-Tmean, 0.85>	94(11)	92(13)(≈)	93(12)(≈)(≈)
3	<Fmax-Fmean, 0.95> <Tmax-Tmean, 0.95>	94(18)	93(10)(≈)	93(12)(≈)(≈)

MWT is the most important feature for the routing rules, which is consistent with the findings in [56]. We also find that different from the routing rule for task <Fmax-Fmean, 0.95> that NIQ is the second important feature, PT plays an important role for task <Tmax-Tmean, 0.95>. One possible reason is that the tardiness related task, that is, <Tmax-Tmean, 0.95>, is more critical to the machine efficiency which is reflected by PT. This is consistent with our intuition. For the sequencing rules, PT and WKR are the two most important features for both task <Fmax-Fmean, 0.95> and task <Fmax-Fmean, 0.95>. There are no obvious differences between these two sequencing rules.

D. Training Time

Table VI shows the mean and standard deviation of training time (in minutes) of GP, MOP_GP, and MOPO_GP in six tasks over 30 independent runs. According to the results, they do not differ significantly from one another. This indicates that the proposed MOPO_GP with task-oriented knowledge sharing can improve the effectiveness of learning scheduling heuristics without requiring an extra computational cost. In other words, MOPO_GP can obtain better performance in a shorter time. This verifies the efficiency of MOPO_GP.

VII. CONCLUSION AND FUTURE WORK

The goal of this article was to develop effective multitask multiobjective GP algorithms to learn scheduling heuristics for DFJSS. The goal has been successfully achieved by proposing an effective multipopulation-based multitask multiobjective GP with the task-oriented knowledge-sharing strategy. This article can significantly contribute to the development of multitask multiobjective learning on dynamic discrete combinatorial optimization problems. This can help researchers understand multitask learning in different domains, especially in dynamic optimization and hyperheuristic learning.

First, this article confirmed that tournament selection is good for multitask multiobjective GP in DFJSS rather than random selection. Second, this article found that using instance rotation and individual reevaluation is a good way for multitask multiobjective GP in DFJSS. This provides a foundation of multitask multiobjective GP on dynamic problems, which can provide guidance for others to work in this direction. Third, we introduced a multipopulation-based multitask multiobjective GP algorithm. We observed that the developed multipopulation-based GP algorithm is much better than the traditional one population-based one in DFJSS. Last, we proposed a multipopulation-based multitask multiobjective GP algorithm with the task-oriented knowledge-sharing strategy. The results indicated that the proposed MOPO_GP can achieve significantly better scheduling heuristics for all the examined scenarios. The effectiveness of the proposed MOPO_GP was also examined by comparing the learned Pareto fronts, the effect of individual diversities on tasks, and the changes of HV and IGD along with generations. It has been observed that the proposed MOPO_GP manages to enhance its performance by maintaining the quality of individuals for tasks, and improving the diversity of individuals for tasks.

Some interesting directions can be further studied in the future. We would like to investigate how to measure the relatedness between multiobjective tasks. In addition, we plan to develop an effective knowledge adaption strategy to help share knowledge in heuristic space efficiently [57].

REFERENCES

[1] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: Toward evolutionary multitasking," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 343–357, Jun. 2016.
[2] Y.-S. Ong and A. Gupta, "Evolutionary multitasking: A computer science view of cognitive multitasking," *Cogn. Comput.*, vol. 8, no. 2, pp. 125–142, 2016.
[3] A. Gupta, Y.-S. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Trans. Cybern.*, vol. 47, no. 7, pp. 1652–1665, Jul. 2017.
[4] A. Gupta, J. Mańdziuk, and Y.-S. Ong, "Evolutionary multitasking in bi-level optimization," *Complex Intell. Syst.*, vol. 1, nos. 1–4, pp. 83–95, 2015.
[5] H. Li, Y.-S. Ong, M. Gong, and Z. Wang, "Evolutionary multitasking sparse reconstruction: Framework and case study," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 733–747, Oct. 2019.
[6] Z. Tang, M. Gong, Y. Wu, W. Liu, and Y. Xie, "Regularized evolutionary multitask optimization: Learning to intertask transfer in aligned subspace," *IEEE Trans. Evol. Comput.*, vol. 25, no. 2, pp. 262–276, Apr. 2021.
[7] J. Zhong, L. Feng, W. Cai, and Y.-S. Ong, "Multifactorial genetic programming for symbolic regression problems," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 50, no. 11, pp. 4492–4505, Nov. 2020.
[8] Z. Xu, X. Liu, K. Zhang, and J. He, "Cultural transmission based multiobjective evolution strategy for evolutionary multitasking," *Inf. Sci.*, vol. 582, pp. 215–242, Jan. 2022.

[9] Q. Chen, X. Ma, Y. Yu, Y. Sun, and Z. Zhu, "Multi-objective evolutionary multi-tasking algorithm using cross-dimensional and prediction-based knowledge transfer," *Inf. Sci.*, vol. 586, pp. 540–562, Mar. 2022.

[10] K. Chen, B. Xue, Z. Mengjie, and F. Zhou, "An evolutionary multitasking-based feature selection method for high-dimensional classification," *IEEE Trans. Cybern.*, vol. 52, no. 7, pp. 7172–7186, Jul. 2022, doi: 10.1109/TCYB.2020.3042243.

[11] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 858–869, Oct. 2019.

[12] S. Bag, S. Gupta, and Z. Luo, "Examining the role of logistics 4.0 enabled dynamic capabilities on firm performance," *Int. J. Logist. Manag.*, vol. 31, no. 3, pp. 923–928, 2020.

[13] O. Bello, A. M. Abu-Mahfouz, Y. Hamam, P. R. Page, K. B. Adedeji, and O. Piller, "Solving management problems in water distribution networks: A survey of approaches and mathematical models," *Water*, vol. 11, no. 3, pp. 1–29, 2019.

[14] W. Li, L. He, and Y. Cao, "Many-objective evolutionary algorithm with reference point-based fuzzy correlation entropy for energy-efficient job shop scheduling with limited workers," *IEEE Trans. Cybern.*, early access, Apr. 19, 2021, doi: 10.1109/TCYB.2021.3069184.

[15] F. Zhang, S. Nguyen, Y. Mei, and M. Zhang, *Genetic Programming for Production Scheduling: An Evolutionary Learning Approach* (Machine Learning: Foundations, Methodologies, and Applications). Singapore: Springer, 2021, pp. 303–338, doi: 10.1007/978-981-16-4859-5.

[16] Q. Zhang, M. Cao, F. Zhang, J. Liu, and X. Li, "Effects of corporate social responsibility on customer satisfaction and organizational attractiveness: A signaling perspective," *Bus. Ethics Environ. Responsibility*, vol. 29, no. 1, pp. 20–34, 2020.

[17] F. Zhang, Y. Mei, and M. Zhang, "Evolving dispatching rules for multi-objective dynamic flexible job shop scheduling via genetic programming hyper-heuristics," in *Proc. IEEE Congr. Evol. Comput.*, 2019, pp. 1366–1373.

[18] 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 Trans. Cybern.*, early access, Mar. 22, 2021, doi: 10.1109/TCYB.2021.3065340.

[19] J. R. Koza and R. Poli, "Genetic programming," in *Search Methodologies*. Boston, MA, USA: Springer, 2005, pp. 127–164.

[20] 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 Trans. Evol. Comput.*, vol. 25, no. 3, pp. 552–566, Jun. 2021, doi: 10.1109/TEVC.2021.3056143.

[21] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," *Comput. Ind. Eng.*, vol. 54, no. 3, pp. 453–473, 2008.

[22] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Instance rotation based surrogate in genetic programming with brood recombination for dynamic job shop scheduling," *IEEE Trans. Evol. Comput.*, early access, Jun. 7, 2022, doi: 10.1109/TEVC.2022.3180693.

[23] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, 1990.

[24] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "A preliminary approach to evolutionary multitasking for dynamic flexible job shop scheduling via genetic programming," in *Proc. Genet. Evol. Comput. Conf.*, 2020, pp. 107–108.

[25] M. Durasevic and D. Jakobović, "A survey of dispatching rules for the dynamic unrelated machines environment," *Expert Syst. Appl.*, vol. 113, pp. 555–569, Dec. 2018.

[26] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling," *IEEE Trans. Cybern.*, vol. 52, no. 8, pp. 8142–8156, Aug. 2022, doi: 10.1109/TCYB.2021.3050141.

[27] M. L. Pinedo, *Scheduling*, vol. 29. New York, NY, USA: Springer, 2012.

[28] G. Conroy, "Handbook of genetic algorithms," *Knowl. Eng. Rev.*, vol. 6, no. 4, pp. 363–365, 1991.

[29] G. Mweshi and N. Pillay, "An improved grammatical evolution approach for generating perturbative heuristics to solve combinatorial optimization problems," *Expert Syst. Appl.*, vol. 165, Mar. 2021, Art. no. 113853.

[30] M. Dumić and D. Jakobović, "Ensembles of priority rules for resource constrained project scheduling problem," *Appl. Soft Comput.*, vol. 110, Oct. 2021, Art. no. 107606.

[31] S. Nguyen, D. Thiruvady, M. Zhang, and D. Alahakoon, "Automated design of multipass heuristics for resource-constrained job scheduling with self-competitive genetic programming," *IEEE Trans. Cybern.*, early access, Mar. 12, 2021, doi: 10.1109/TCYB.2021.3062799.

[32] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 110–124, Feb. 2016.

[33] Y. Zheng, Z. Zhu, Y. Qi, L. Wang, and X. Ma, "Multi-objective multifactorial evolutionary algorithm enhanced with the weighting helper-task," in *Proc. Int. Conf. Ind. Artif. Intell.*, 2020, pp. 1–6.

[34] Y. Chen, J. Zhong, L. Feng, and J. Zhang, "An adaptive archive-based evolutionary framework for many-task optimization," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 3, pp. 369–384, Jun. 2020, doi: 10.1109/TETCI.2019.2916051.

[35] T. Wei and J. Zhong, "Towards generalized resource allocation on evolutionary multitasking for multi-objective optimization," *IEEE Comput. Intell. Mag.*, vol. 16, no. 4, pp. 20–37, Nov. 2021.

[36] K. K. Bali, A. Gupta, Y.-S. Ong, and P. S. Tan, "Cognizant multitasking in multiobjective multifactorial evolution: MO-MFEA-II," *IEEE Trans. Cybern.*, vol. 51, no. 4, pp. 1784–1796, Apr. 2021.

[37] Q. Shang, Y. Huang, Y. Wang, M. Li, and L. Feng, "Solving vehicle routing problem by memetic search with evolutionary multitasking," *Memetic Comput.*, vol. 14, pp. 31–44, Jan. 2022.

[38] L. Feng *et al.*, "Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3171–3184, Jun. 2021.

[39] L. Feng *et al.*, "Explicit evolutionary multitasking for combinatorial optimization: A case study on capacitated vehicle routing problem," *IEEE Trans. Cybern.*, vol. 51, no. 6, pp. 3143–3156, Jun. 2021.

[40] E. Osaba, A. D. Martinez, J. L. Lobo, I. Laña, and J. Del Ser, "On the transferability of knowledge among vehicle routing problems by using cellular evolutionary multitasking," in *Proc. Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–8.

[41] L. Feng *et al.*, "Evolutionary multitasking via explicit autoencoding," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3457–3470, Sep. 2019.

[42] X. Hao, R. Qu, and J. Liu, "A unified framework of graph-based evolutionary multitasking hyper-heuristic," *IEEE Trans. Evol. Comput.*, vol. 25, no. 1, pp. 35–47, Feb. 2021.

[43] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with TSP, QAP, LOP, and JSP," in *Proc. IEEE Region 10 Conf.*, 2016, pp. 3157–3164.

[44] 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 Trans. Evol. Comput.*, vol. 25, no. 4, pp. 651–665, Aug. 2021.

[45] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios: A genetic programming approach," in *Proc. Conf. Genet. Evol. Comput.*, 2010, pp. 257–264.

[46] N. Wang, Q. Xu, R. Fei, J. Yang, and L. Wang, "Rigorous analysis of multi-factorial evolutionary algorithm as multi-population evolution model," *Int. J. Comput. Intell. Syst.*, vol. 12, no. 2, pp. 1121–1133, 2019.

[47] S. Nguyen, M. Zhang, D. Alahakoon, and K. C. Tan, "Visualizing the evolution of computer programs for genetic programming," *IEEE Comput. Intell. Mag.*, vol. 13, no. 4, pp. 77–94, Nov. 2018.

[48] 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 Trans. Cybern.*, vol. 51, no. 4, pp. 1797–1811, Apr. 2021.

[49] T. Hildebrandt and J. Branke, "On using surrogates with genetic programming," *Evol. Comput.*, vol. 23, no. 3, pp. 343–367, 2015.

[50] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: A survey with a unified framework," *Complex Intell. Syst.*, vol. 3, no. 1, pp. 41–66, 2017.

[51] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic programming via iterated local search for dynamic job shop scheduling," *IEEE Trans. Cybern.*, vol. 45, no. 1, pp. 1–14, Jan. 2015.

[52] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.

[53] C. A. C. Coello and M. R. Sierra, "A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm," in *Proc. Int. Conf. Artif. Intell.*, 2004, pp. 688–697.

[54] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. KDD*, vol. 96, 1996, pp. 226–231.

[55] S. M. Gustafson, "An analysis of diversity in genetic programming," Ph.D. dissertation, Dept. Doctor Philos., Univ. Nottingham, Nottingham, U.K., 2004.

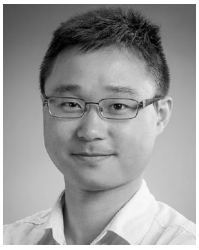
[56] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Genetic programming with adaptive search based on the frequency of features for dynamic flexible job shop scheduling," in *Proc. Eur. Conf. Evol. Comput. Comb. Optim.*. 2020, pp. 214–230.

[57] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Dynamic multi-objective job shop scheduling: A genetic programming approach," in *Automated Scheduling and Planning*. Berlin, Germany: Springer, 2013, pp. 251–282.

Fangfang Zhang (Member, IEEE) received the B.Sc. and M.Sc. degrees from Shenzhen University, Shenzhen, China, in 2014 and 2017, respectively, and the Ph.D. degree in computer science from the Victoria University of Wellington, Wellington, New Zealand, in 2021.

She is a Research Fellow of Artificial Intelligence with the School of Engineering and Computer Science, Victoria University of Wellington. She has over 40 journal and conference papers. Her current research interests include evolutionary computation, hyperheuristic learning/optimization, job-shop scheduling, and surrogate and multitask learning.

Dr. Zhang is a member of the IEEE Computational Intelligence Society and Association for Computing Machinery, and has been severing as a Reviewer for top international journals, such as the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and IEEE TRANSACTIONS ON CYBERNETICS, and conferences, including the Genetic and Evolutionary Computation Conference and IEEE Congress on Evolutionary Computation. She is a Vice-Chair of the Task Force on Evolutionary Scheduling and Combinatorial Optimisation. She is also a Committee Member and the Chair of Professional Activities Coordinator, and the Treasurer of Young Professional Affinity Group of the IEEE New Zealand Central Section.



Yi Mei (Senior Member, IEEE) received the B.Sc. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively.

He is a Senior Lecturer with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. He has more than 150 fully referred publications, including the top journals in EC and Operations Research, such as the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE TRANSACTIONS ON CYBERNETICS, *Evolutionary Computation*, *European Journal of Operational Research*, and *ACM Transactions on Mathematical Software*. His research interests include evolutionary scheduling and combinatorial optimization, machine learning, genetic programming, and hyperheuristics.

Dr. Mei was a Vice-Chair of the IEEE CIS Emergent Technologies Technical Committee and a member of the Intelligent Systems Applications Technical Committee. He is an editorial board member/associate editor of three international journals, and a Guest Editor of a special issue of the *Genetic Programming and Evolvable Machines*. He serves as a reviewer of over 50 international journals.

Su Nguyen (Member, IEEE) received the Ph.D. degree in artificial intelligence and data analytics from the Victoria University of Wellington, Wellington, New Zealand, in 2013.

He is a Senior Lecturer and an Algorithm Lead with the Centre for Data Analytics and Cognition, La Trobe University, Melbourne, VIC, Australia. He has more than 70 publications in top EC/OR peer-reviewed journals and conferences. His expertise includes evolutionary computation, simulation optimization, automated algorithm design, interfaces of AI/OR, and their applications in logistics, energy, and transportation. His current research focuses on novel people-centric artificial intelligence to solve dynamic and uncertain planning tasks by combining the creativity of evolutionary computation and power of advanced machine-learning algorithms.

Dr. Nguyen was the Chair of the IEEE Task Force on Evolutionary Scheduling and Combinatorial Optimisation from 2014 to 2018 and is a member of the IEEE CIS Data Mining and Big Data Technical Committee. He delivered technical tutorials about EC and AI-based visualization at the Parallel Problem Solving from Nature Conference in 2018 and IEEE World Congress on Computational Intelligence in 2020. He served as an Editorial Member for the *Complex and Intelligence Systems* and the Guest Editor of the special issue on *Automated Design and Adaption of Heuristics for Scheduling and Combinatorial Optimization in Genetic Programming and Evolvable Machines*.

Mengjie Zhang (Fellow, IEEE) received the B.E. and M.E. degrees from the Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 2000.

He is currently a Professor of Computer Science, the Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) with the Faculty of Engineering, Victoria University of Wellington, Wellington, New Zealand. He has published over 800 research papers in refereed international journals and conferences. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multiobjective optimization, feature selection and reduction, job-shop scheduling, and transfer learning.

Prof. Zhang was the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, IEEE CIS Emergent Technologies Technical Committee, and Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is a Vice-Chair of the Task Force on Evolutionary Computer Vision and Image Processing and the Founding Chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a Committee Member of the IEEE NZ Central Section. He is a Fellow of the Royal Society, a Fellow of Engineering of New Zealand, and an IEEE Distinguished Lecturer.