Multitask Genetic Programming Based Generative Hyper-heuristics: A Case Study in Dynamic Scheduling

Fangfang Zhang, *Graduate Student Member, IEEE,* Yi Mei, *Senior Member, IEEE,* Su Nguyen, *Member, IEEE,* Kay Chen Tan, *Fellow, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Evolutionary multitask learning has achieved great success due to its ability to handle multiple tasks simultaneously. However, it is rarely used in the hyper-heuristic domain which aims at generating a heuristic for a class of problems rather than solving one specific problem. The existing multitask hyper-heuristic studies only focus on heuristic selection, which is not applicable to heuristic generation. To fill the gap, we propose a novel multitask generative hyper-heuristic approach based on genetic programming in this paper. Specifically, we introduce the idea in evolutionary multitask learning to genetic programming hyper-heuristics with a suitable evolutionary framework and individual selection pressure. In addition, an origin-based offspring reservation strategy is developed to maintain the quality of individuals for each task. To verify the effectiveness of the proposed approach, comprehensive empirical studies have been conducted on the homogeneous and heterogeneous multitask dynamic flexible job shop scheduling. The results show that the proposed algorithm can significantly improve the quality of scheduling heuristics for each task in all the examined scenarios. In addition, the evolved scheduling heuristics verify the mutual help among the tasks in a multitask scenario.**

***Index Terms*—Multitask Learning, Hyper-heuristic, Genetic Programming, Dynamic Scheduling, Job Shop Scheduling.**

I. Introduction

Multitask learning is a paradigm that aims at solving multiple self-contained tasks simultaneously. The paradigm of multifactorial optimisation toward evolutionary multitask (MFEA) was given in [1], [2] for solving multiple tasks simultaneously in evolutionary algorithms. The success of MFEA relies on the knowledge sharing mechanism [3] between tasks by assortative mating and vertical cultural transmission during the evolutionary process. Specifically, the candidate solutions in a population for tasks enhance each other by harnessing the hidden relationships between them via continuous genetic transfer in a unified search space. Based on the assumption that most problems in real life are interconnected, MFEA has been widely and successfully applied to solve different problems such as continuous numeric optimisation [2], [4], [5], symbolic regression [6], job shop scheduling [7], [8], feature selection [9] and timetabling [10].

A hyper-heuristic [11], [12] approach seeks to select or generate heuristics to solve hard computational search problems efficiently in a search space with heuristics. The unique characteristic of hyper-heuristic is that its search space consists of heuristics rather than solutions. The goal of a hyper-heuristic is to find a sequence of low-level heuristics (selective hyper-heuristic) or generate an informative high-level heuristic (generative hyper-heuristic) rather than finding the solution directly [13]. In other words, hyper-heuristic outputs a heuristic rather than a solution. Hyper-heuristic approaches have two main advantages. First, the learned high-level heuristics have good reusability, and can easily be applied to a range of problem scenarios. Second, generative hyper-heuristic approaches can handle the dynamic problems efficiently, because the evolved heuristics are typically used as priority functions and can make real-time decisions. Multitask selective hyper-heuristic has been investigated in [10] on exam timetabling and graph colouring problems, however, to the best of our knowledge, there is no study on multitask generative hyper-heuristic.

Genetic programming (GP) [14], as one of the most popular used evolutionary algorithms, is a promising hyper-heuristic approach [11], [15], especially in heuristic generation due to its flexibility of representation. Compared with heuristic approaches [16], hyper-heuristic approaches work on heuristic space with heuristics (e.g., the dispatching rule in scheduling) rather than solution space with solutions (e.g., the schedule in scheduling). It is noted that the obtained heuristics of hyper-heuristic approaches can be used to generate solutions depending on the investigated problem. More details about hyper-heuristics can be found in [13], [17]. A GP individual is a combination of terminals and functions. The terminals correspond to the inputs of the computer program, which

Manuscript received XXX; revised XXX and XXX; accepted XXX. This work is supported in part by the Marsden Fund of New Zealand Government under Contract VUW1509 and Contract VUW1614; in part by the Science for Technological Innovation Challenge Fund under Grant E3603/2903; and in part by the MBIE SSIF Fund under Contract VUW RTVU1914. This work is partially supported by the Research Grants Council of the Hong Kong SAR under grant No. CityU11202418 and No. CityU11209219. The work of Fangfang Zhang was supported by the China Scholarship Council/Victoria University Scholarship. This article was recommended by Associate Editor XXX. (*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).

Kay Chen Tan is with the Department of Computing, Hong Kong Polytechnic University (e-mail: kctan@polyu.edu.hk).

This article has supplementary downloadable material available at XXX, provided by the authors.

Colour versions of one or more of the figures in this article are available online at XXX.

Digital Object Identifier XXX

are specific to the problems. The functions can include any arithmetic operations, logical functions, and domain-specific functions. GP hyper-heuristic with tree-based representation has been successfully used for evolving scheduling heuristics for complex combinatorial optimisation problems such as dynamic job shop scheduling [18], [19], [20], [21], [22]. However, GP is rarely used in multitask learning. Multitask GP was successfully used to the symbolic regression problem in [6]. However, the GP approach in [6] worked on the solution space rather than heuristic space. In addition, gene expression representation [23], [24] with a fixed length of strings was used in [6], which is a vector-based representation rather than tree-based representation. It is noted that handling with tree-based representation in GP is more challenging than vector-based representation in genetic algorithms due to its variable-length characteristic. A slight change of an individual with tree-based representation in GP can lead to a severe change to its quality. There are a number of related dynamic tasks in real-world applications such as cloud computing [25] and job shop scheduling [26], [27] with a preference for hyper-heuristic approaches. Thus, this paper presents an attempt to fill the gap of multitask in generative hyper-heuristic by adapting the idea of MFEA to multitask GP hyper-heuristic for evolving scheduling heuristics in dynamic scheduling. The benefit of multitask GP learning in this paper is that by handling a number of tasks simultaneously, each task can be solved more effectively with the help from solving the other tasks than being solved independently.

GP and hyper-heuristics have their own features so that directly applying MFEA [2] to GP and hyper-heuristics may not achieve satisfactory effectiveness and efficiency. First, GP and genetic algorithm [28] use different ways to control the selection pressure. In GP, the selection pressure is typically implemented by the parent selection for breeding. However, MFEA follows a common genetic algorithm framework that combines the parent and offspring populations, and select the best individuals from the combined population to the next generation. Using both the parent selection in GP and environment selection in MFEA simultaneously will make MFGP too greedy and lose the population diversity. Second, in addition to the training performance, hyper-heuristic approaches ultimately aims to achieve good performance on the unseen test instances, which is known as generalisation. To improve generalisation, a commonly used strategy used in GP hyper-heuristic is to rotate the training instances at each generation [11], [29]. Concatenating parent population with offspring population requires to re-evaluate the individuals in parent population on the same training instances of the offspring. This can lead to a lengthy training process. Last but not least, the evaluations of heuristics are normally time-consuming, which requires to run a long simulation to measure the quality of the heuristics. The way in MFEA that allocates offspring to different tasks by calculating their fitness on all the tasks and assigns them to the best task at the first generation is not efficient. To address the above issues, this paper proposes an effective hyper-heuristic multitask algorithm based on the characteristics of GP. Specifically, we remove the concatenation operation by introducing multiple subpopulations for tasks with tournament selection and effective knowledge sharing mechanism between tasks.

The overall goal of this work is to develop a novel and effective multitask genetic programming based generative hyper-heuristic approach. The proposed algorithm is expected to improve the quality of the evolved high-level heuristics for all the tasks considered in the multitask scenario. Specifically, the contributions of this paper are given as follows:

1) Propose an effective multitask GP based generative hyper-heuristic approach according to the characteristics of GP. Specifically, tasks are solved independently via multiple subpopulations. An novel origin-based offspring reservation strategy is proposed to share knowledge between different tasks via crossover.
2) Verify the proposed algorithm on homogeneous and heterogeneous multitask scenarios in dynamic flexible job shop scheduling problems. The results show that the proposed algorithm can evolve highly-competitive scheduling heuristics for different tasks in all examined homogeneous and heterogeneous multitask dynamic scheduling scenarios.
3) Analyse the evolved scheduling heuristics and show that the proposed algorithm can solve multiple tasks collaboratively in terms of the structure and behaviour of obtained high-level scheduling heuristics.

II. Background

A. Multifactorial Evolutionary Algorithm

The main feature of MFEA is the use of skill factor τ to represent the allocation of individuals for different tasks. The skill factor represents the task that one individual is best at. The knowledge transfer is realised by the crossover operator between individuals with different skill factors explicitly.

Fig. 1 shows the flowchart of MFEA with k tasks [2]. In the beginning, a population of individuals are randomly initialised based on the predefined unified representation. Then, each individual is evaluated on all tasks, and the individual is allocated for its fittest task. As an evolutionary algorithm, the working of MFEA is based on the transmission of cultural genetic materials from parents to their offspring. This is an important step of MFEA that plays a role of transferring knowledge between tasks. In particular, assortative mating and vertical cultural transmission are used with two randomly selected parents to generate the offspring. Assortative mating states that individuals prefer to mate with those belonging to the same cultural background. In MFEA, the skill factor is regarded as an individual's cultural bias. The individuals with the same or different skill factor(s) have the same or different culture in MFEA. Vertical cultural transmission is a mode of inheritance that the phenotype of an offspring is directly affected by the phenotype of its parents. In MFEA, it is realised by allowing offspring to inherit the skill factor of their parents. Finally, the parent population P and offspring population P_{new} are concatenated as an intermediate population P_{imd}, and the top $popsize$ fittest individuals from P_{imd} are kept into the next generation. The output of MFEA is k individuals, each for one task.

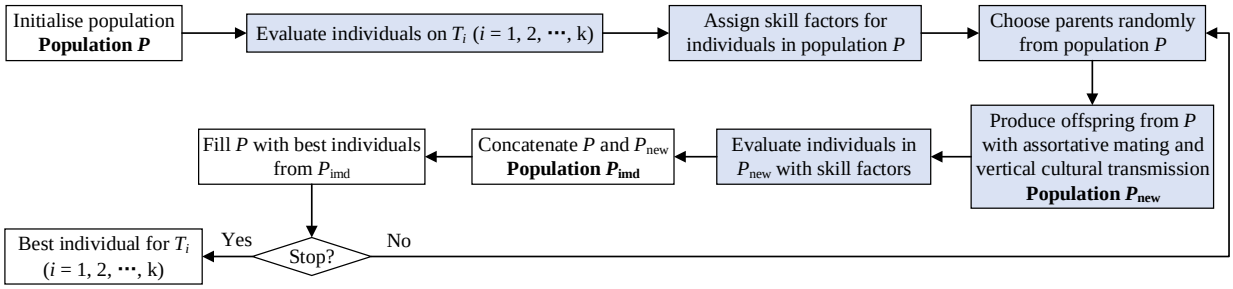


Fig. 1. The flowchart of MFEA with k tasks, where P, P_{new}, and P_{imd} denote the evaluated, newly generated offspring, and the concatenated population.

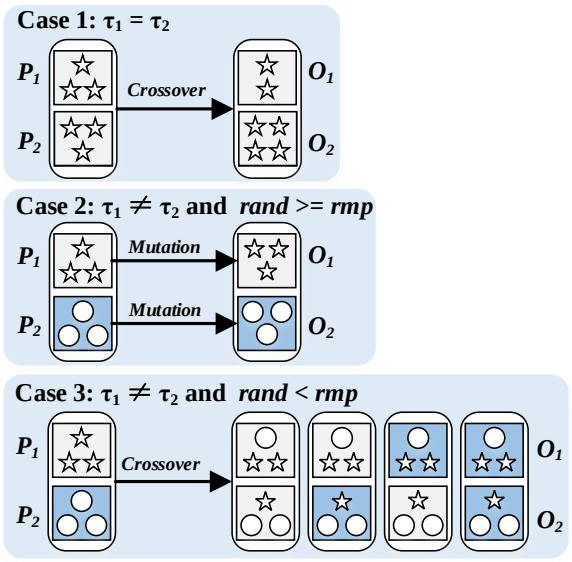


Fig. 2. An example of generating offspring of MFEA, where P_1 and P_2 indicate the selected parents, and O_1 and O_2 represent the generated offspring.

Fig. 2 shows an example of how the interaction of the knowledge is achieved. Assume there are two tasks in a multitask scenario. The individuals for task 1 and task 2 are in grey and dark blue, and the star and circle indicate different genetic materials of individuals for task 1 and task 2, respectively. Two parents (P_1 and P_2) are randomly selected from the parent population uniformly with a random value from 1 to $popsize$ (i.e., the index of individuals). If the selected parents are optimised for the same task ($\tau_1 = \tau_2$), then their offspring will inherit their culture for optimising the same task (Case 1). If the selected parents are designed for different tasks ($\tau_1 \neq \tau_2$), then there are two options. If the random value is larger than the predefined random mating probability (rmp), the mutation operator will be triggered to produce one offspring for each parent, and the corresponding offspring inherits the culture of the parent directly (Case 2). Otherwise, two offspring will be produced, and each of them has 50% probability of inheriting one of the parents' cultures (Case 3). There are four different situations according to how they inherit the skill factors.

B. Genetic Programming Hyper-heuristics

The optimal structures of heuristics are normally not known in real-world applications, which makes the heuristic learning process challenging. GP, as a hyper-heuristic method [30], has been successfully applied to evolve scheduling heuristics for combinatorial optimisation problems [31], [32], [33], [34]. Tree-based GP [11], [35] is a good candidate to learn heuristics for solving problems due to its flexible representation. This implies that the structures of heuristics do not need to be defined in advance. The goal of GP hyper-heuristic is to find an effective high-level heuristic by constructing low-level heuristics and functions properly.

Fig. 3 shows the flowchart of a typical GP to learn heuristic. Compared with Fig. 1, Fig. 3 shows that MFEA differs from GP mainly in the way of evaluating individuals, selecting parents and producing offspring. Different from MFEA, the GP individuals only need to be evaluated for a single task rather than multiple tasks. The parent(s) are selected using tournament selection in GP [6], [14], and the newly generated individuals do not need to be evaluated immediately. In addition, there is no population concatenation operation in GP.

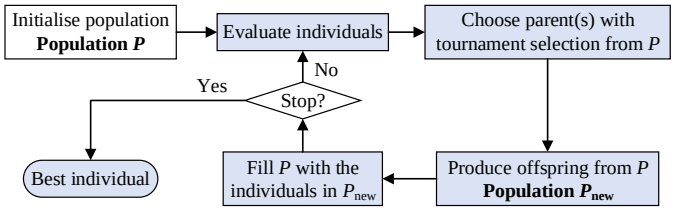


Fig. 3. The flowchart of a typical GP.

C. Knowledge Transfer in Genetic Programming

In the field of transfer learning in GP, regarding "what to transfer", there are two main schemes [36]. One is the "*FullTree*" that migrates a number of individuals with good quality directly. The other is to transfer the "*SubTree*" which is extracted from individuals. Different from the traditional transfer learning in GP [37], there are no source and target domains in this work. Thus, there is no knowledge extraction process from the source problem. The knowledge should be transferred between different tasks during the evolutionary process. Migrating individuals for sharing knowledge to other tasks [38], [39] highly relies on the quality of selected individuals. If poor-quality individuals are moved to other tasks, or good-quality individuals perform poorly on other tasks, they will be eliminated quickly and fail to play a role in knowledge sharing. Crossover is an important genetic operator in GP to produce offspring, which is an effective

carrier for knowledge sharing. In addition, changing building-blocks between individuals via the crossover can relieve the dependence on the quality of individuals. Therefore, taking the effectiveness and efficiency into consideration, we can transfer the knowledge between tasks by exchanging the subtrees of the individuals for different tasks via the crossover operator.

III. Related Works of Evolutionary Multitask Algorithms

Although evolutionary multitask [1], [2] is a relatively new paradigm, it has recently received much research interests in optimising multiple tasks simultaneously. Most existing multitask approaches aims at improving the qualities of solutions for all the tasks directly [40], [41], thus ignoring the hyper-heuristic research area. A unified framework of graph-based evolutionary multitask hyper-heuristic approach was proposed and examined on timetabling and graph-colouring problems [10]. However, the proposed approach was a heuristic selection approach rather than heuristic generation approach. In addition, it was only compared with the simple single-tasking hyper-heuristics, and there is no further analysis of the heuristic structure. From the perspective of involved research fields, evolutionary multitask has been successfully applied to continuous numeric optimisation with benchmarks [2], [4], [42], [43], and regression problems [6]. However, the studies on discrete, combinatorial problems which have more complex situations are still very limited. This limits its applications in practice. In terms of solution representation, most studies are conducted with the vector-based search space [43], [44], [45], [46] rather than tree-based search space.

Multitask GP has been investigated for combinatorial optimisation problems, such as team orienteering [38] and dynamic job shop scheduling [7], [39]. The problem investigated in [38] is static. The training instances are clustered for each island, and several individuals with better fitness are transferred between islands. However, the approach cannot be applied for dynamic problems with simulation, since the training instances are not available for clustering. In addition, in [39], a niching approach was proposed for dynamic job shop scheduling. However, the main drawback is that the niched individuals need further evaluations, which is not an efficient way. In [7], multitask GP was applied to dynamic flexible job shop scheduling, and the efficiency of solving multiple dynamic flexible job shop scheduling problems was dramatically improved. However, the quality of the evolved scheduling heuristics were not improved.

In summary, the research on hyper-heuristic multitask is still in its early stage. In this paper, we propose a multitask GP based generative hyper-heuristic approach. It is a good case study for enhancing the development of multitask learning into the hyper-heuristic domain.

IV. The Multitask Genetic programming Based Generative Hyper-heuristics

A. The Framework of the Proposed Algorithm

We propose to use multiple independent subpopulations to solve different tasks but keep knowledge sharing between them. The low-level heuristics are set as the terminal set of GP and combined with the function set to form a GP individual. To handle multiple tasks (T_1, T_2, ... , T_k) simultaneously, we group the GP individuals by equally dividing the entire population into k subpopulations ($Subpop_1$, $Subpop_2$, ... , $Subpop_k$). The individuals in the same (different) subpopulation are used to optimise the same (different) task. On the one hand, each subpopulation is independent from each other, and the individuals in different subpopulations are evolved for different tasks. Each subpopulation can be seen as the individuals with the same skill factor in MFEA. On the other hand, different subpopulations assist with each other by sharing their knowledge with others, which is realised by the genetic operator.

The main framework of the proposed algorithm is presented in Algorithm 1. The input is the k tasks that are expected to be solved. The output of a GP run consists of k best evolved rules (h_1^*, h_2^*, ... , h_k^*), each for a task. $subpopsize_i$ indicates the number of individuals of subpopulation i. The fitness of a heuristic h_i is denoted by $fitness_{h_i}$. There are three

Algorithm 1: The Framework of the Proposed Algorithm

Input : k tasks T_1, T_2, ... , T_k
Output: The best evolved heuristics for each task h_1^*, h_2^*, ... , h_k^*
Initialisation: Randomly initialise the population with k subpopulations
set h_1^*, h_2^*, ... , $h_k^* \leftarrow null$
set $fitness_{h_1^*}$, $fitness_{h_2^*}$, ... , $fitness_{h_k^*} \leftarrow +\infty$
$gen \leftarrow 0$
while $gen < maxGen$ **do**
  // **Evaluation**: Evaluate the individuals in the population
  for *i = 1 to* k **do**
    for *j = 1 to* $subpopsize_i$ **do**
      Calculate $fitness_{h_j}$ based on fitness function of T_i
    end
  end
  for *i = 1 to* k **do**
    for *j = 1 to* $subpopsize_i$ **do**
      if $fitness_{h_j} < fitness_{h_i^*}$ **then**
        $h_i^* \leftarrow h_j$
      end
    end
  end
  if $gen < maxGen - 1$ **then**
    // Instance rotation with a new random seed
    // **Evolution**: Generate offspring for each subpopulation
    for *i = 1 to* k **do**
      for *j = 1 to* $subpopsize_i$ **do**
        if *Crossover is applied* **then**
          if *rand* $\leq$ *rmp* **then**
            Choose the first parent from $Subpop_i$
            Choose the second parent from $Subpop_{\neg i}$
            Produce one offspring with the proposed ***origin-based offspring reservation*** strategy
          else
            Choose two parents from $Subpop_i$, and produce two offspring with traditional GP crossover
          end
        else
          Choose one parent from $Subpop_i$
          Do mutation or reproduction accordingly
        end
      end
    end
  end
  $gen \leftarrow gen + 1$
end
return h_1^*, h_2^*, ... , h_k^*

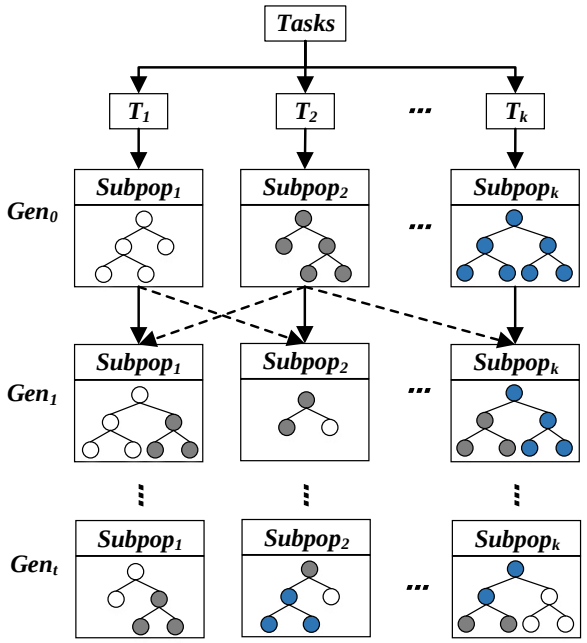


Fig. 4. The framework of the proposed multitask GP based generative hyper-heuristic with a focus on the knowledge transfer.

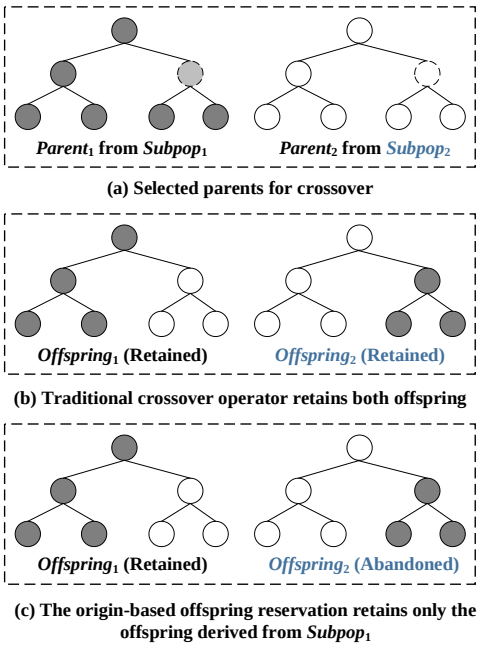


Fig. 5. An example of generating offspring for $Subpop_1$ by sharing knowledge from $Subpop_2$.

main differences between the proposed multitask GP hyper-heuristic and the traditional GP for a single task. Specifically, *at the initialisation stage*, the population consists of multiple subpopulations, and each subpopulation is designed to solve one task (line 1). *During the evaluation process*, the individuals in different subpopulations are evaluated independently (from line 6 to line 11). *During the evolution stage*, the crossover operator is applied to share knowledge between the tasks. If the knowledge sharing condition is met, the offspring of each subpopulation are generated according to the proposed knowledge sharing mechanism with the origin-based offspring reservation strategy (from line 25 to line 28). Otherwise, traditional GP crossover will be used to produce two offspring (from line 29 to line 32). It is noted that the proposed knowledge sharing mechanism via crossover is conducted on the individuals from different subpopulations (for different tasks), which the traditional GP crossover works on the individuals from the same subpopulation (for the same task). If the algorithm meets the stopping criterion, the current best heuristic will be selected as the final obtained high-level heuristic. Otherwise, the search process continues. The final high-level heuristic is the best generated heuristic in the population.

B. Knowledge Sharing

Fig. 4 shows the framework of the proposed multitask GP based generative hyper-heuristic with a focus on the knowledge transfer. At generation 0, a population with k subpopulations is initialised for solving k tasks. The individuals in each subpopulation are equally assigned for each task and fixed for that task during the whole evolutionary process. Specifically, the individuals with white, grey and blue colours are initialised for T_1 (task 1), T_2 (task 2), and T_k (task k), respectively. When generating offspring to the next generation, we use the crossover operator for knowledge transfer but keep the individuals for each task fixed, and the offspring for each subpopulation are produced sequentially. Finally, the best evolved high-level heuristic for one task consists of the genetic materials of individuals that originally belong to other tasks. This is a clear manifestation of mutual learning between tasks.

This paper uses a knowledge transfer ratio of rmp to control the frequency to gain knowledge from other subpopulations at each generation. If the knowledge share mechanism is triggered ($rand \leq rmp$), the first parent $parent_1$ will be selected from the current subpopulation, and the other parent $parent_2$ will be selected from one of the other subpopulations. Note that if there are more than two subpopulations, when producing offspring for one subpopulation, one of the remaining subpopulations will be selected randomly. In the traditional GP, the two parents will produce two offspring, as shown in Fig. 5 (b). Taking $Subpop_1$ as an example, a random subpopulation is chosen to share knowledge with $Subpop_1$. Assume that $Subpop_2$ is selected, the generated offspring with knowledge transfer for $Subpop_1$ consists of white (from $Subpop_2$) and grey (from $Subpop_1$) elements, thus, the knowledge transfer between tasks is realised. We can see that the newly generated offspring contain genetic materials from the individuals for different tasks. Otherwise ($rand > rmp$), two parents will be selected from the current subpopulation (i.e., the same subpopulation) to produce two offspring for a new subpopulation. However, for an individual, an effective knowledge sharing mechanism should not only make the individual obtain useful information but also maintain the original characteristics of the individuals. We call this process the *origin-based offspring reservation*. When producing offspring for the current task, only the offspring generated based on the parent from the corresponding subpopulation (e.g., $Subpop_1$ in this example) will be kept. Specifically, Fig. 5 (b) shows the traditional

crossover operator that retains both offspring. On the other hand, Fig. 5 (c) shows the origin-based offspring reservation, which retains only the offspring generated based on the parent from $Subpop_1$.

C. Summary

The proposed algorithm is an extension of the traditional multifactorial evolutionary multitask in [1] by optimising the multitask framework according to the characteristics of GP for developing hyper-heuristic approach. It is noted that the proposed algorithm is not problem-dependent, and can be applied to other domains but with proper construction of related tasks for the specific problems such as evolving routing rules for the arc routing problems, which will be investigated in our future work. There are a number of improvements compared with the traditional MFEA. First, in terms of the number of evaluations, the proposed algorithm requires fewer evaluations than MFEA. The individuals in the initialised population do not need to be evaluated for all tasks for deciding the skill factor for each individual (i.e., can save $popsize * k$ evaluations). Second, in terms of the individual selection pressure, there is no concatenation operation of the parent and offspring populations, which brings another benefit. If we handle dynamic problems with changed training instance at each generation, we will avoid the extra re-evaluation of the individuals in the parent population (i.e., potentially $popsize * maxGen$ evaluations). Third, in terms of evaluation resource, the number of individuals for each task is fixed and equal, which is easy to manage. This also reduces the parameters in the algorithm, such as skill factor. From the perspective of computational cost, the proposed algorithm can save up to $popsize * (k + maxGen)$ evaluations.

V. A study on Multitask Genetic Programming Based Generative Hyper-heuristic for dynamic scheduling

GP, as a hyper-heuristic approach, has been widely used to evolve scheduling heuristics for the dynamic scheduling problems [47]. Dynamic flexible job shop scheduling (DFJSS) [48], [49] is a typical dynamic scheduling problem with different but similar scheduling tasks which can benefit from multitask learning. To verify the performance of the proposed multitask GP based generative hyper-heuristic approach, we take DFJSS as a test bed.

A. Dynamic Flexible Job Shop Scheudling

In DFJSS, n jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$ need to be processed by m machines $\mathcal{M} = \{M_1, M_2, ..., M_m\}$. Each job J_j has an arrival time $at(J_j)$ and a sequence of operations $O_j = (O_{j1}, O_{j2}, ..., O_{ji})$. The completion of the last operation for a job means the job has been finished. Each operation O_{ji} can only be processed by one of its optional machines $\pi(O_{ji})$ and its processing time $\delta(O_{ji})$ depends on the machine that processes it. Therefore, machine assignment and operation sequencing need to be made simultaneously in DFJSS. This paper focuses on one dynamic event, i.e., dynamically and continuously arriving new jobs. This means that the information of a job is unknown until it arrives at the shop floor. The following constraints must be satisfied in the DFJSS problem:

- A machine can process at most one operation at a time.
- Each operation can be processed only by one of its candidate machines at a time.
- One cannot start processing an operation until all its precedent operations have been processed.
- The processing of an operation cannot be stopped or paused until it is completed.

The objective of the scheduling is the optimised performance criterion for a problem while satisfying all the above constraints. We consider three commonly used objective functions. The calculations of the objectives are shown as follows:

- Mean-flowtime $= \frac{\sum_{j=1}^{n}\{C_j - r_j\}}{n}$
- Mean-tardiness $= \frac{\sum_{j=1}^{n} Max\{0, C_j - d_j\}}{n}$
- Mean-weighted-tardiness $= \frac{\sum_{j=1}^{n} w_j * Max\{0, C_j - d_j\}}{n}$

where C_j is the completion time of job J_j, r_j is the release time of J_j, d_j is the due date of J_j, w_j is the weight of J_j, and n is the number of jobs that are expected to be processed.

The simulation in this paper assumes that 5000 jobs need to be processed by 10 machines. New jobs will arrive over time according to a Poisson process with rate λ. The number of operations of a job is randomly generated from a uniform discrete distribution between 1 and 10. The number of candidate machines for an operation follows a uniform discrete distribution between 1 and 10. In addition, the importance of jobs varies, and the weights of 20%, 60%, and 20% of jobs are set as 1, 2, and 4, respectively [50]. The processing time of each operation is assigned by a uniform discrete distribution with the range [1, 99]. The due time of a job is set at 1.5 times of its total processing time. To improve the generalisation of the evolved heuristics, the training sample used at each generation is changed by assigning a new random seed for sampling from these random distributions [26].

Utilisation level (p) is an essential factor to represent different job shop scenarios [32]. It indicates the proportion of time that a machine is expected to be busy. The expression of λ is shown in Eq. (1), where μ is the average processing time of the machines, and P_M is mean the probability of a job visiting a machine. For example, P_M is 2/10 if each job has two operations. It is noted that λ is fixed for all the instances for a task with the same utilisation level, and a higher utilisation level tends to lead to a busier job shop.

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

To estimate the steady-state performance, the first 1000 jobs are considered as warm-up jobs and discarded in the objective calculations. This work collects data from the next 5000 jobs. The simulation stops when the 6000th job is finished.

B. Multitask DFJSS Task Definition

Although there are lots of multitask studies, most of them work on benchmark problems, and the relatedness between tasks is widely studied [51]. However, what kinds of DFJSS

problems are related and can be optimised in a multitask scenario is not clear. In this subsection, we define the related tasks based on the characteristics of DFJSS.

1) Tasks with the same objective but different utilisation levels: In real applications, the demand for a specific product varies over time rather than fixed [52]. For example, the amount of orders of T-shirt in summer is likely to be larger than that in winter. A larger amount leads to more complex scheduling. Although the complexities of job shops can be different, they are commonly similar in production, and have the same goal such as minimising the total production time. Thus, we define the tasks with different utilisation levels (i.e., indicate different complexities) but with the same objective to be naturally related tasks for building a multitask scenario.

2) Tasks with different objectives but the same utilisation level: For a scheduling task, different customers may have different requirements [53]. One may require to minimise the flowtime to reduce the total cost. Others may prefer to minimise the tardiness to hand out to products to the customers in time. Although the objectives are different, they both involve reducing the idle time of the machines in the shop floor. The knowledge learned from one objective might be also helpful for the others. Therefore, we define the tasks with different objectives but with the same utilisation level to be the related tasks considered in a multitask scenario. In this way, we can focus on verifying the effectiveness of the proposed algorithm in heterogeneous scenarios with different objectives.

For simplicity, we name the multitask problem with the same objective but with the different utilisation levels as *homogeneous multitask*, while the multitask problem with different objectives but with the same utilisation level as *heterogeneous multitask*. Intuitively, the tasks in the heterogeneous multitask tend to be less related than the tasks in the homogeneous multitask, since the objective is an important indicator to guide the optimisation direction and the objectives in heterogeneous multitask are different.

C. Scheduling Heuristics for DFJSS

The solution of a multitask problem is a set of scheduling heuristics $P_s = \{h_1, h_2, ..., h_k\}$, each for a task. Each scheduling heuristic is composed of a routing rule for machine assignment and a sequencing rule for operation sequencing in DFJSS [49]. The routing rule or sequencing rule will be used to prioritise the candidate machines or operations to make a schedule. The most prior machine or operation will be selected. Least work in the queue (WIQ) and shortest processing time (SPT) are the commonly used scheduling heuristics for machine assignment and operation sequencing, respectively [54]. Taking these two rules as an example, when a new operation comes, the routing rule WIQ will be triggered to prioritise all its candidate machines, and assign the operation to the most prior machine. The new operation will be assigned to the machine whose has the least workload. Similarly, when a machine is idle, the sequencing rule SPT will be used to prioritise the operations in its queue, and the most prior operation will be chosen to be processed next. The operation which needs the shortest processing time will be chosen to be processed next.

TABLE I
THE TERMINAL SET.

Notation	Description
NIQ	The number of operations in the queue
WIQ	Current work in the queue
MWT	Waiting time of a machine
PT	Processing time of an operation on a specified machine
NPT	Median processing time for the next operation
OWT	The waiting time of an operation
WKR	Median amount of work remaining for a job
NOR	The number of operations remaining for a job
W	Weight of a job
TIS	Time in system

TABLE II
THE PARAMETER SETTINGS IN GP.

Parameter	Value
*Number of subpopulations	k
*Subpopulation size	400
*The number of elites for each subpopulation	10
*Parent selection	Tournament selection with size 5
*Crossover / Mutation / Reproduction rate	80% / 15% / 5%
**Number of tasks	k
**Population size with re-evaluation	200 * k
**Population size without re-evaluation	400 * k
**The number of elites for each task	10
**Parent selection	Random selection
Method for initialising population	ramped-half-and-half
Initial minimum / maximum depth	2 / 6
maximal depth of programs	8
Terminal / non-terminal selection rate	10% / 90%
The number of generations	51
The transfer ratio	0.3

* : for the algorithms with multiple subpopulations only
** : for the algorithms with one population only

D. Parameter Settings in Genetic Programming

The features of the job shop are considered as the terminals of GP. The features are commonly extracted based on the characteristics of machines (e.g., NIQ, WIQ, and MWT), operations (e.g., PT, NPT, and OWT), and jobs (e.g., WKR, NOR, W, and TIS) in the job shop floor [55]. The low-level heuristics are usually designed based on the features. The details are shown in Table I. The function set is set to $\{+, -, *, /, max, min\}$, following the setting in [56]. Each function takes two arguments. The "/" function is protected division, returning one if divided by zero. The max and min functions take two arguments and return the maximum and minimum of their arguments, respectively. The other parameter settings of GP are shown in Table II.

E. Design of Comparisons

For the homogeneous multitask, we consider three multitask scenarios, and each with a different objective, i.e., mean flowtime (denoted as Fmean), mean tardiness (denoted as Tmean), and mean weighted tardiness (denoted as WTmean). The utilisation level is a commonly used parameter [57], [58] to represent different job shop scenarios for measuring the effectiveness of the algorithms. The utilisation levels of 0.75, 0.85, and 0.95 are used in the homogeneous multitask scenarios, since they are three typical distinct configurations in

TABLE III
THE DESIGNED **HOMOGENEOUS** MULTITASK SCENARIOS WITH TASKS REPRESENTED BY OPTIMISED OBJECTIVE AND UTILISATION LEVEL.

Scenario	task 1	task 2	task 3
Scenario 1	<Fmean, 0.75>	<Fmean, 0.85>	<Fmean, 0.95>
Scenario 2	<Tmean, 0.75>	<Tmean, 0.85>	<Tmean, 0.95>
Scenario 3	<WTmean, 0.75>	<WTmean, 0.85>	<WTmean, 0.95>

TABLE IV
THE DESIGNED **HETEROGENEOUS** MULTITASK SCENARIOS WITH TASKS REPRESENTED BY OPTIMISED OBJECTIVE AND UTILISATION LEVEL.

Scenario	task 1	task 2
Scenario 1	<Fmax, 0.95>	<Tmax, 0.95>
Scenario 2	<Fmean, 0.95>	<Tmean, 0.95>
Scenario 3	<WFmean, 0.95>	<WTmean, 0.95>

DFJSS [31], [47] with different complexities. Each utilisation level represents a task in the homogeneous multitask [59], which is used to verify the effectiveness of the proposed multitask learning algorithm on the tasks with the same objective but different complexities. The objective and the utilisation level are two important factors to represent the characteristics of a task in a scenario. It is noted that a higher utilisation level will lead to a more complex task. The details of the designed homogeneous multitask scenarios represented by optimised objective and utilisation level are shown in Table III. For the heterogeneous multitask, we consider three heterogeneous multitask scenarios and choose the most complex scenario with a utilisation level of 0.95 for investigation [21]. For each heterogeneous multitask scenario, two different objectives are involved without varying the utilisation levels between tasks. In this way, we can focus on verifying the effectiveness of the proposed algorithm on the tasks with different objectives in heterogeneous scenarios. Except for the objectives introduced earlier, max-flowtime, max-tardiness, mean-weighted-flowtime are denoted as Fmax, Tmax, and WFmean, respectively. The details are shown in Table IV.

The GP system with k subpopulations to solve k tasks independently named GP is used as the baseline algorithm without multitask. The second compared algorithm combines MFEA in [2] with GP without rotating training instances named MFGP. In addition, MFGP with *rotating* training instances but without re-evaluation named $\mathrm{MFGP}^{\mathrm{r}-}$, which MFGP with rotating training instances and re-evaluation named $\mathrm{MFGP}^{\mathrm{r}+}$. The proposed multitask GP based generative hyper-heuristic approach without the proposed offspring reservation strategy named $\mathrm{M}^2\mathrm{GP}$, since it involves both multitask and multi-population. $\mathrm{M}^2\mathrm{GP}$ with the proposed *offspring* reservation strategy named $\mathrm{M}^2\mathrm{GP}^{\mathrm{f}}$.

To verify the adaptability of MFEA to GP, $\mathrm{MFGP}^{\mathrm{r}-}$, $\mathrm{MFGP}^{\mathrm{r}+}$ and MFGP are compared. To verify the effectiveness of the proposed $\mathrm{M}^2\mathrm{GP}$ and the origin-based offspring reservation strategy, GP, MFGP, $\mathrm{M}^2\mathrm{GP}$, and $\mathrm{M}^2\mathrm{GP}^{\mathrm{f}}$ are compared in both homogeneous and heterogeneous multitask scenarios. In addition, the effectiveness of the proposed $\mathrm{M}^2\mathrm{GP}^{\mathrm{f}}$ on the common tasks between homogeneous and heterogeneous scenarios are further compared with MFGP. The effectiveness of the multitask mechanism is examined by analysing the evolved scheduling heuristics for each task in a multitask scenario. The evolved rule is tested on 50 unseen instances, and the average objective value across the 50 test instances is reported as the test performance of the rule, which can be a good approximation of the true performance of the rule.

TABLE V
THE MEAN (STANDARD DEVIATION) OF THE OBJECTIVE VALUES ON **TEST INSTANCES** OF $\mathrm{MFGP}^{\mathrm{r}-}$, $\mathrm{MFGP}^{\mathrm{r}+}$ AND MFGP OVER 30 INDEPENDENT RUNS IN THREE HOMOGENEOUS MULTITASK SCENARIOS.

Sce.	Task	$\mathrm{MFGP}^{\mathrm{r}-}$	$\mathrm{MFGP}^{\mathrm{r}+}$	MFGP
1	<Fmean, 0.75>	339.97(1.11)	337.01(1.39)(–)	336.60(1.21)(–)(≈)
	<Fmean, 0.85>	396.21(2.90)	387.91(3.72)(–)	386.67(2.93)(–)(≈)
	<Fmean, 0.95>	586.77(6.50)	560.04(8.64)(–)	556.55(5.83)(–)(≈)
2	<Tmean, 0.75>	16.08(0.95)	13.90(0.66)(–)	13.60(0.25)(–)(≈)
	<Tmean, 0.85>	46.32(2.74)	41.51(1.92)(–)	40.54(0.66)(–)(≈)
	<Tmean, 0.95>	202.54(3.40)	182.88(5.60)(–)	180.39(4.46)(–)(≈)
3	<WTmean, 0.75>	33.56(2.55)	28.44(1.87)(–)	27.26(0.61)(–)(–)
	<WTmean, 0.85>	97.12(5.22)	79.90(4.79)(–)	76.95(2.19)(–)(–)
	<WTmean, 0.95>	381.03(29.32)	310.91(15.34)(–)	303.05(8.84)(–)(–)

VI. RESULTS AND DISCUSSIONS

Friedman's test with a significance level of 0.05 is applied to rank the algorithms based on their performance. If Friedman's test gives significance results, we further conduct Wilcoxon rank-sum test with Bonferroni correction between the proposed algorithm and other algorithms with a significance level of 0.05 for the post-hoc pairwise comparisons. In the following results, "–", "+", and "≈" indicate that the corresponding result is significantly better than, worse than or similar to its counterpart. An algorithm will be compared with the algorithm(s) before it one by one. Since the tasks in this paper are minimisation problems, a smaller value indicates a better performance. "Win, Draw, Lose" means the number of scenarios that a compared algorithm is statistically better, similar, or worse than $\mathrm{M}^2\mathrm{GP}^{\mathrm{f}}$. "Average Rank" shows the average ranking of the algorithm on all the examined scenarios.

A. The Adaptation of MFEA to GP in Dynamic Scheduling

The performance (i.e., objective value) on unseen data is commonly used to examine the quality of the evolved scheduling heuristics. Table V shows the mean and standard deviation of the objective values on unseen instances of $\mathrm{MFGP}^{\mathrm{r}-}$, $\mathrm{MFGP}^{\mathrm{r}+}$ and MFGP according to 30 independent runs in three homogeneous multitask scenarios. Compared with $\mathrm{MFGP}^{\mathrm{r}-}$, the performance of $\mathrm{MFGP}^{\mathrm{r}+}$ becomes significantly better. This indicates that rotating training instances requires the re-evaluation of the individuals in the next generation to provide accurate fitness. The results show that MFGP without rotating training instances can achieve similar or better performance with $\mathrm{MFGP}^{\mathrm{r}+}$ in six or three scenarios, respectively. In addition, MFGP can achieve significant better performance than $\mathrm{MFGP}^{\mathrm{r}+}$ in three scenarios (i.e., <WTmean, 0.75>, <WTmean, 0.85>, and <WTmean, 0.95>). We can see that the multitask in dynamic scheduling of both rotating training with re-evaluation and without rotating training instances can

TABLE VI
THE MEAN (STANDARD DEVIATION) OF THE OBJECTIVE VALUES ON TEST INSTANCES OF GP, MFGP, M^2GP, AND M^2GP^f OVER 30 INDEPENDENT RUNS IN THREE **HOMOGENEOUS MULTITASK** SCENARIOS.

Scenario	Task	GP	MFGP	M^2GP	M^2GP^f
1	<Fmean, 0.75>	337.57(1.80)	336.60(1.21)(≈)	335.86(0.91)(–)(–)	336.17(1.04)(–)(≈)(≈)
	<Fmean, 0.85>	388.79(4.30)	386.67(2.93)(≈)	385.14(1.94)(–)(–)	385.73(2.33)(–)(–)(≈)
	<Fmean, 0.95>	561.35(9.16)	556.55(5.83)(≈)	553.11(4.26)(–)(–)	552.74(4.75)(–)(–)(≈)
2	<Tmean, 0.75>	14.08(1.10)	13.60(0.25)(≈)	13.34(0.27)(–)(–)	13.33(0.25)(–)(–)(≈)
	<Tmean, 0.85>	41.61(2.73)	40.54(0.66)(≈)	39.75(0.87)(–)(–)	39.78(0.84)(–)(–)(≈)
	<Tmean, 0.95>	182.34(7.72)	180.39(4.46)(≈)	176.84(3.10)(–)(–)	176.65(4.17)(–)(–)(≈)
3	<WTmean, 0.75>	28.81(2.66)	27.26(0.61)(≈)	27.27(0.99)(–)(–)	26.92(0.72)(–)(–)(≈)
	<WTmean, 0.85>	81.23(7.63)	76.95(2.19)(≈)	76.43(3.19)(–)(–)	75.34(2.06)(–)(–)(–)
	<WTmean, 0.95>	312.26(15.86)	303.05(8.84)(–)	297.72(10.38)(–)(–)	295.67(8.44)(–)(–)(≈)
	Win / Draw / Lose	0 / 0 / 9	0 / 1 / 8	0 / 8 / 1	N/A
	Average Rank	3.17	2.89	2.03	**1.91**

get comprising performance. The number of individuals evaluations in $MFGP^{r+}$ and MFGP is the same, it is a trade-off between sampling more individuals and re-evaluation. However, overall, MFGP outperforms $MFGP^{r+}$. Therefore, MFGP will be used for comparison later. Note that it does not mean rotating training instances is not useful for DFJSS. Rotating training instances with traditional GP evolutionary process can achieve similar performance with individual selection pressure by concatenating parent and offspring populations for DFJSS.

B. Quality of the Evolved Scheduling Heuristics

1) Homogeneous Multitask: Table VI shows the mean and standard deviation of the objective values on the test instances of GP, MFGP, M^2GP, and M^2GP^f over 30 independent runs in three homogeneous multitask scenarios. Overall, M^2GP^f is the best algorithm based on the average ranking according to the Friedman test. MFGP does not show any significant difference from GP in most of the scenarios, which indicates that applying the idea of MFEA into GP directly is not effective in the context of GP hyper-heuristic. It is consistent with our intuition, since the characteristics of GP are quite different from other evolutionary algorithms. M^2GP performs significantly better than both GP and MFGP in all the examined scenarios. This verifies the effectiveness of the proposed M^2GP in homogeneous multitask scenarios. M^2GP^f also shows it superiority compared with GP and MFGP. This verifies the effectiveness of the proposed origin-based offspring reservation strategy.

Fig. 6 shows the violin plot of the average objective values on test instances based on 30 independent runs of GP, MFGP, M^2GP, and M^2GP^f in three homogeneous multitask scenarios. According to the distribution of the 30 objective values, MFGP can achieve smaller values than that of GP. This indicates that the idea of MFEA [2] does help, however, MFEA with GP does not perform well. We can see that M^2GP shows its superiority with smaller objective values in general among the three involved algorithms. This shows that tasks with the same objective but different utilisation levels in DFJSS can be solved simultaneously in a mutually reinforcing way. In addition, the objective values of M^2GP^f tend to be smaller than M^2GP in most of the scenarios (e.g., <Fmean, 0.95>, <Tmean, 0.75>, <Tmean, 0.95>, <WTmean, 0.75>,

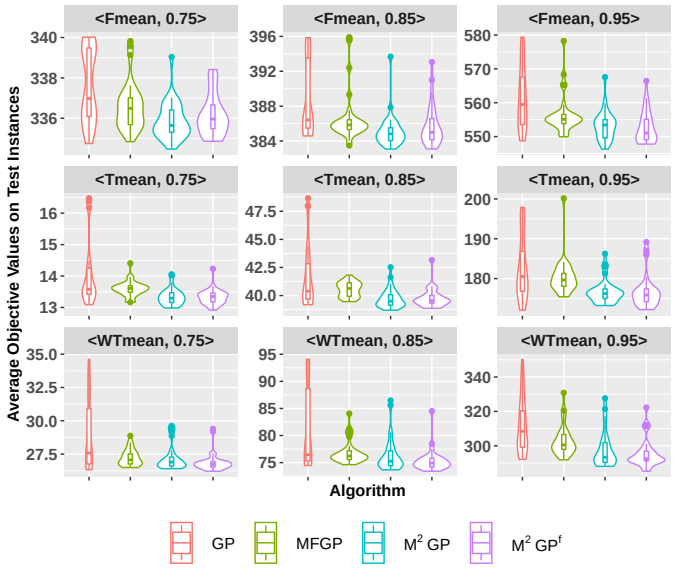


Fig. 6. The violin plot of the average objective values on test instances of GP, MFGP, and M^2GP based on 30 independent runs in three *homogeneous multitask* scenarios (each row is a multitask scenario).

<WTmean, 0.85>, and <WTmean, 0.95>). This indicates that the proposed origin-based offspring strategy does can benefit the proposed algorithm M^2GP. In addition, M^2GP and M^2GP^f can safely remove the repeated evaluations of individuals on all the tasks at the first generation without sacrificing the performance.

2) Heterogeneous Multitask: Table VII shows the mean and standard deviation of the objective values on test instances of GP, MFGP, M^2GP and M^2GP^f based on 30 independent runs in three heterogeneous multitask scenarios. Overall, the proposed M^2GP^f is the first according to the average rank obtained by the Friedman's. Different from the observation in homogeneous multitask, MFGP and M^2GP do not outperform GP, since they are not significantly better than GP in most scenarios. M^2GP^f, however, achieves significantly better results than GP and MFGP in most of the scenarios, and it outperforms M^2GP in one scenario (e.g., <Tmean, 0.95>). One possible reason is that the tasks with different objectives on heterogeneous multitask are less related, and the proposed origin-based offspring reservation strategy can

TABLE VII
THE MEAN (STANDARD DEVIATION) OF THE OBJECTIVE VALUES ON TEST INSTANCES OF GP, MFGP, $\mathrm{M^2GP}$, AND $\mathrm{M^2GP^f}$ OVER 30 INDEPENDENT RUNS IN THREE **HETEROGENEOUS MULTITASK** SCENARIOS.

Scenario	Task	GP	MFGP	$\mathrm{M^2GP}$	$\mathrm{M^2GP^f}$
1	<Fmax, 0.95>	2032.96(98.29)	2081.77(76.40)**(+)**	1991.15(88.56)**(–)**(–)	1981.28(37.19)(–)(–)(≈)
	<Tmax, 0.95>	1580.81(54.13)	1647.75(55.45)**(+)**	1576.03(54.94)(≈)**(–)**	1575.13(37.84)(≈)(–)(≈)
2	<Fmean, 0.95>	560.72(10.18)	556.10(6.10)(≈)	556.52(9.11)(≈)(≈)	553.79(7.43)(–)(–)(≈)
	<Tmean, 0.95>	180.81(6.83)	178.76(3.47)(≈)	180.20(6.51)(≈)(≈)	177.55(5.72)(–)(–)(–)
3	<WFmean, 0.95>	1136.33(25.65)	1121.67(12.74)(–)	1123.87(20.45)(–)(≈)	1121.05(22.20)(–)(–)(≈)
	<WTmean, 0.95>	311.20(16.82)	301.12(8.07)(–)	303.06(15.23)(–)(≈)	300.00(15.38)(–)(–)(≈)
	Win / Draw / Lose	0 / 1 / 5	0 / 0 / 6	0 / 5 / 1	N/A
	Average Rank	2.86	2.77	2.41	**1.97**

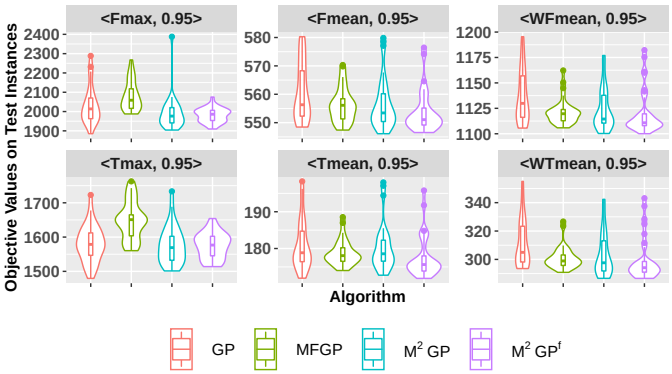


Fig. 7. The violin plot of the average objective values on test instances of GP, MFGP, $\mathrm{M^2GP}$, and $\mathrm{M^2GP^f}$ based on 30 independent runs in three *heterogeneous multitask* scenarios (each column is a multitask scenario).

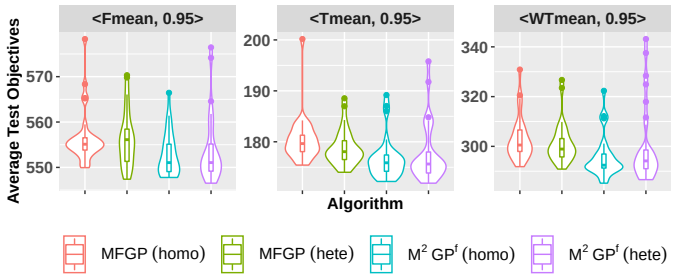


Fig. 8. The violin plot of the average objective values on test instances of MFGP and $\mathrm{M^2GP^f}$ in both homogeneous (denoted as homo) and heterogeneous (denoted as hete) multitask scenarios for their common tasks based on 30 independent runs.

maintain the quality of individuals for each task well, since it aims to keep the main characteristics of the individuals for the corresponding tasks.

Fig. 7 shows the violin plot of the average objective values on unseen instances of GP, MFGP, $\mathrm{M^2GP}$, and $\mathrm{M^2GP^f}$ based on 30 independent runs in three heterogeneous multitask scenarios. It is obvious that the objective values achieved by $\mathrm{M^2GP}$ tend to be much smaller than that of GP and MFGP. This indicates that tasks with different objectives but the same utilisation level in DFJSS can also be solved simultaneously in a mutually reinforcing way. In addition, we can see that the objective values obtained by $\mathrm{M^2GP^f}$ tend to be smaller than $\mathrm{M^2GP}$, which confirms the positive effect of the proposed origin-based offspring reservation strategy.

3) Homogeneous Versus Heterogeneous Multitask: There are three common tasks (i.e., <Fmean, 0.95>, <Tmean, 0.95>, and <WTmean, 0.95>) solved in both the homogeneous and heterogeneous multitask scenarios. It is interesting to know the quality of the evolved scheduling heuristics obtained for the same task in different types of multitask scenarios. Note that they are comparable because the number of evaluations for the corresponding tasks is equal.

Fig. 8 shows the violin plot of the average objective values on unseen data of MFGP and $\mathrm{M^2GP^f}$ for the common tasks between the homogeneous and heterogeneous multitask scenarios. Overall, for all scenarios, $\mathrm{M^2GP^f}$ performs better than MFGP in both homogeneous and heterogeneous multitask. Based on the distributions of the achieved objective values, both MFGP and $\mathrm{M^2GP^f}$ show it superiority in heterogeneous multitask scenarios rather than homogeneous multitask scenarios for most of the common tasks. The objective values obtained from heterogeneous multitask are distributed in a relatively lower position.

In summary, the proposed algorithm $\mathrm{M^2GP^f}$ can achieve better performance with both homogeneous and heterogeneous multitask scenarios. In addition, we find that learning in a heterogeneous multitask scenario has more potential to improve the quality of the evolved scheduling heuristics.

C. The Evolved High-level Scheduling Heuristics

We choose the evolved scheduling heuristics, including routing and sequencing rules for tasks in a heterogeneous multitask scenario to investigate how the tasks help with each other from the perspective of the genotypes of individuals. The second heterogeneous multitask scenario is selected, since it contains two out of three common tasks (e.g., <Fmean, 0.95>, <Tmean, 0.95>, and <WTmean, 0.95>) with homogeneous multitask scenarios. The routing rules are the best evolved rules for task 1 and task 2 from the same run in heterogeneous multitask scenario 2, and the sequencing rules are the corresponding sequencing rules of the routing rules mentioned above. Note that a machine or an operation with a smaller priority value is more prior.

1) Routing Rules: Fig. 9 and Fig. 10 show one of the evolved routing rules for task <Fmean, 0.95> and <Tmean, 0.95> in the second scenario of heterogeneous multitask, respectively. It is obvious that these two scheduling heuristics share knowledge between each other, since the major part of

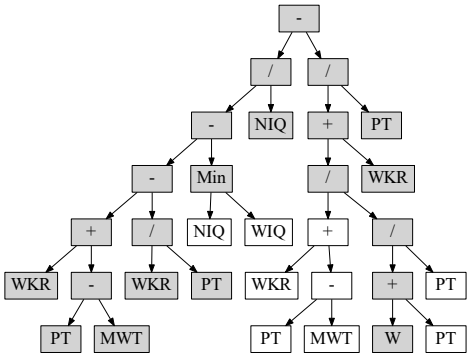


Fig. 9. One of the best evolved routing rules for task 1 <Fmean, 0.95> in heterogeneous multitask scenario 2.

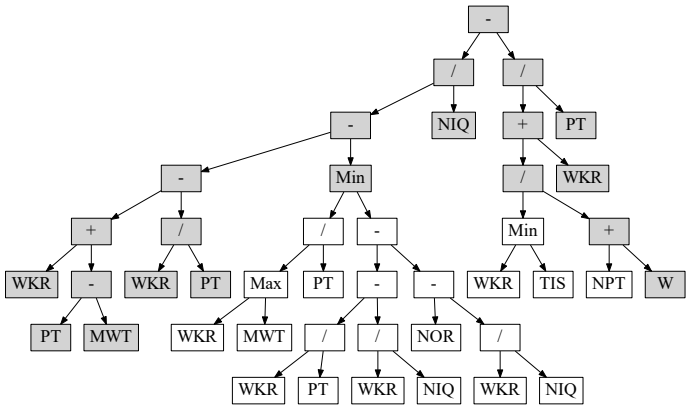


Fig. 10. One of the best evolved routing rules for task 2 <Tmean, 0.95> in heterogeneous multitask scenario 2.

the rules is the same which is highlighted in grey. WKR and PT are the most commonly appeared terminals in the grey area, which are both important for the machine assignment in minimising mean-flowtime and mean-tardiness.

In this paper, we investigate the behaviour of the scheduling heuristics with a focus on the rules for minimising mean-tardiness. The routing rule for minimising mean-tardiness in Fig. 10 can be further simplified, as shown in Eq. (2).

$$\begin{aligned} R =& \{WKR + PT - MWT - \frac{WKR}{PT} - Min\{ \\ & \frac{Max\{WKR, MWT\}}{PT}, \frac{WKR}{PT} - NOR\}\}/NIQ \\ & - \frac{Min\{WKR, TIS\}}{PT(NPT + W)} - \frac{WKR}{PT} \\ \approx& \{PT - MWT - \frac{WKR}{PT} - Min\{ \\ & \frac{Max\{WKR, MWT\}}{PT}, \frac{WKR}{PT} - NOR\}\}/NIQ \\ & - \frac{Min\{WKR, TIS\}}{PT(NPT + W)} - \frac{WKR}{PT} \end{aligned} \tag{2}$$

WKR and PT are the two most commonly used low-level heuristics for this routing rule based on the occurrences of heuristics. However, the value of WKR (i.e., the remaining work of the corresponding job of an operation for all the machines) is the same. This indicates that WKR is not an important factor for this rule to distinguish the machines, and it can be considered as a constant. Similarly, W (i.e., the

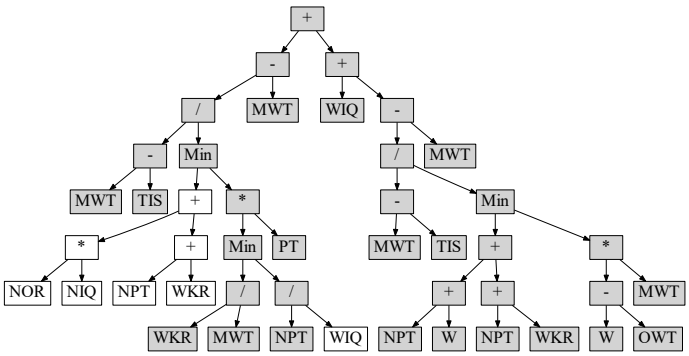


Fig. 11. One of the best evolved sequencing rules for task 1 <Fmean, 0.95> in heterogeneous multitask scenario 2.

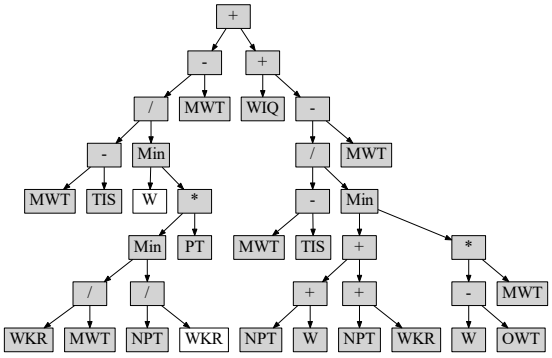


Fig. 12. One of the best evolved sequencing rules for task 2 <Tmean, 0.95> in heterogeneous multitask scenario 2.

importance of a job), NOR (i.e., the number of remaining operations of a job), NPT (i.e., the median processing time of the next operation), and TIS (i.e., the time that an operation in the job shop floor) are also considered as constants here. Without considering WKR, W and NOR if possible, this rule can be further simplified as shown in step 2 in Eq. (2).

This rule tends to choose the machine with smaller processing time (i.e., efficient machine for an operation) and a longer waiting time (i.e., the time that a machine is idle). It is consistent with our intuition, since we would like to use efficient machines and make better use of the machine resources to reduce the producing time for products. In addition, this rule prefers to allocate an operation to a machine with a larger number of operations (NIQ). Note that a larger NIQ does not mean an overhead workload for a machine, since the processing time of the operations can be small. In this case, if an operation is allocated to a machine with a larger NIQ, it might have higher chance to be processed soon, because the sequencing decision is more often to be triggered.

2) Sequencing Rules: Fig. 11 and Fig. 12 show the corresponding sequencing rules of the routing rules, as shown in Fig. 9 and Fig. 10, respectively. The main framework of these two sequencing rules for different tasks in a multitask scenario is the same as shown in grey, and only two smaller parts of them are different. This means that these two sequencing rules strongly share their knowledge in the evolutionary process. In the grey area, MWT and NPT are the two most important terminals based on the frequency of terminals, which are both important operation sequencing in minimising mean-flowtime

and mean-tardiness.

The corresponding sequencing rule (Fig. 12) of the routing rule in Fig. 10 is shown in Eq. (3). From step 1 to step 2, "Min{W, PT * Min{$\frac{WKR}{MWT}$, $\frac{NPT}{WKR}$}}" can be further simplified to W, since W (e.g., 1, 2 and 4) is more likely to be smaller than "PT * Min{$\frac{WKR}{MWT}$, $\frac{NPT}{WKR}$}". "(W - OWT) * MWT" is more likely to be a negative number, because W is usually smaller than OWT. Therefore, "Min{2NPT+W+WKR, (W-OWT)*MWT}" can be further simplified as "(W - OWT) * MWT". Finally, since the value of W is much smaller than OWT, W is removed, as shown in the last step in Eq. (3).

$$\begin{aligned} S =& \frac{MWT - TIS}{Min\{W, PT * Min\{\frac{WKR}{MWT}, \frac{NPT}{WKR}\}\}} - \\ & \frac{MWT - TIS}{Min\{2NPT + W + WKR, (W - OWT) * MWT\} \\ & - 2MWT + WIQ} \\ \approx& \frac{MWT - TIS}{W} + \frac{MWT - TIS}{(OWT - W) * MWT} \\ \approx& \frac{MWT - TIS}{W} + \frac{MWT - TIS}{OWT * MWT} \end{aligned} \tag{3}$$

For the operations that in the queue of a machine, MWT (i.e., machine waiting time) is the same for all the operations, and can be considered as a constant. This sequencing rule suggests to choose the operation that comes to the job shop floor for a long time (i.e., large TIS), and waits in the queue of a machine for a long time (i.e., large OWT). It is consistent with our intuition that a long time waiting will delay the production, and it is conducive to minimise the tardiness. In addition, the machine prefers to choose the important operation with a large W. This is also consistent with our intuition that important jobs should be processed earlier to reduce the delay for improving customer satisfaction.

In summary, both the routing and sequencing rules for tasks in heterogeneous multitask scenarios share lots of knowledge with each other. We observe the same pattern in the homogeneous multitask scenarios, but it is not included in the paper due to page limit. We can conclude that the proposed algorithm can solve the tasks in a mutually reinforcing way.

VII. Conclusions

This paper has successfully proposed an effective multitask genetic programming based generative hyper-heuristic approach. The effectiveness of the proposed algorithm was examined on both homogeneous and heterogeneous multitask scenarios with dynamic flexible job shop scheduling problems. The contributions of the proposed generative hyper-heuristic multitask algorithm are three-fold.

First, the proposed algorithm broadens the study of multitask on the hyper-heuristic domain. It extends the application of the multitask approach to evolving high-level heuristics for the complex dynamic combinatorial optimisation problems. Second, a new variation of the traditional multitask framework was proposed based on the characteristics of genetic programming. It expands the paradigm of evolutionary multitask to other popular evolutionary algorithms but with different features such as selection pressure and representation. Last, the way of defining multitask scenarios in dynamic combinatorial optimisation problem can provide guidance for employing multitask to real-world applications.

The results showed that the proposed $\mathrm{M^2GP^f}$ can achieve scheduling heuristics with competitive quality in all of the homogeneous and heterogeneous multitask scenarios. In addition, $\mathrm{M^2GP^f}$ is robust in terms of the performance in both homogeneous and heterogeneous multitask scenarios. We also found that the task with a heterogeneous multitask scenario has more potential to be optimised well. The effectiveness of the proposed multitask GP hyper-heuristic was examined by not only comparing the quality of evolved scheduling heuristics, but also the structures and behaviours of the evolved scheduling heuristics for all tasks in a multitask scenario. It has also been observed that the proposed algorithm does manage to solve the tasks in a mutually reinforcing way.

Some interesting directions can be further studied in future. We would like to investigate the effectiveness of $\mathrm{M^2GP^f}$ on other problem domains such as the arc routing problems. We plan to work on the multitask scenarios with more than three tasks. We will find effective ways to construct multitask tasks in the complex problems with unknown fitness landscape. We will propose an effective way to measure the relatedness of tasks to develop a more effective approach based on the current multitask genetic programming based generative hyper-heuristics framework. We will also investigate more on what is transferred between tasks in the multitask scenarios from different perspectives such as the phenotype of the individual. In addition, more advanced knowledge sharing mechanism will be designed based on the relatedness of tasks.

References

[1] A. Gupta, Y. Ong, L. Feng, and K. C. Tan, "Multiobjective multifactorial optimization in evolutionary multitasking," *IEEE Transactions on Cybernetics*, vol. 47, no. 7, pp. 1652–1665, 2017.

[2] A. Gupta, Y.-S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2015.

[3] ——, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 51–64, 2017.

[4] G. Li, Q. Lin, and W. Gao, "Multifactorial optimization via explicit multipopulation evolutionary framework," *Information Sciences*, vol. 512, pp. 1555–1570, 2020.

[5] L. Zhou, L. Feng, K. C. Tan, J. Zhong, Z. Zhu, K. Liu, and C. Chen, "Toward adaptive knowledge transfer in multifactorial evolutionary computation," *IEEE Transactions on Cybernetics*, 2020, Doi: 10.1109/TCYB.2020.2974100.

[6] J. Zhong, L. Feng, W. Cai, and Y.-S. Ong, "Multifactorial genetic programming for symbolic regression problems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2018.

[7] 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 *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2020, pp. 107–108.

[8] 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 *Proceedings of the IEEE Region 10 Conference*. IEEE, 2016, pp. 3157–3164.

[9] K. Chen, B. Xue, M. Zhang, and F. Zhou, "An evolutionary multitasking-based feature selection method for high-dimensional classification," *IEEE Transactions on Cybernetics*, 2020. Doi: 10.1109/TCYB.2020.3042243.

[10] X. Hao, R. Qu, and J. Liu, "A unified framework of graph-based evolutionary multitasking hyper-heuristic," *IEEE Transactions on Evolutionary Computation*, 2020. Doi: 10.1109/TEVC.2020.2991717.

[11] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 110–124, 2016.

[12] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," *Journal of the Operational Research Society*, vol. 64, no. 12, pp. 1695–1724, 2013.

[13] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of metaheuristics*. Springer, 2010, pp. 449–468.

[14] J. R. Koza, *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science Stanford, CA, 1990, vol. 34.

[15] 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*, 2020. Doi: 10.1109/TCYB.2021.3050141.

[16] S. Salhi, *Heuristic search: The emerging science of problem solving*. Springer, 2017.

[17] N. Pillay and R. Qu, *Hyper-heuristics: Theory and applications*. Springer, 2018.

[18] M. Durasevic, D. Jakobovic, and K. Knezevic, "Adaptive scheduling on unrelated machines with genetic programming," *Applied Soft Computing*, vol. 48, pp. 419–430, 2016.

[19] 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, 2019, pp. 347–355.

[20] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Genetic programming for evolving due-date assignment models in job shop environments," *Evolutionary Computation*, vol. 22, no. 1, pp. 105–138, 2014.

[21] S. Nguyen, M. Zhang, D. Alahakoon, and K. C. Tan, "Visualizing the evolution of computer programs for genetic programming," *IEEE Computational Intelligence Magazine*, vol. 13, no. 4, pp. 77–94, 2018.

[22] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Correlation coefficient based recombinative guidance for genetic programming hyper-heuristics in dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2021. Doi: 10.1109/TEVC.2021.3056143.

[23] J. Zhong, Y.-S. Ong, and W. Cai, "Self-learning gene expression programming," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 65–80, 2015.

[24] L. Nie, X. Shao, L. Gao, and W. Li, "Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems," *The International Journal of Advanced Manufacturing Technology*, vol. 50, no. 5-8, pp. 729–747, 2010.

[25] M. Guzek, P. Bouvry, and E.-G. Talbi, "A survey of evolutionary computation for resource management of processing in cloud computing," *IEEE Computational Intelligence Magazine*, vol. 10, no. 2, pp. 53–67, 2015.

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

[27] F. Zhang, Y. Mei, and M. Zhang, "Can stochastic dispatching rules evolved by genetic programming hyper-heuristics help in dynamic flexible job shop scheduling?" in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2019, pp. 41–48.

[28] G. Conroy, "Handbook of genetic algorithms," *The Knowledge Engineering Review*, vol. 6, no. 4, pp. 363–365, 1991.

[29] F. Zhang, Y. Mei, and M. Zhang, "Surrogate-assisted genetic programming for dynamic flexible job shop scheduling," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 766–772.

[30] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Proceedings of the Computational intelligence*. Springer, 2009, pp. 177–201.

[31] F. Zhang, Y. Mei, and 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, 2019, pp. 33–49.

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

[33] M. Durasevic and D. Jakobovic, "Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment," *Genetic Programming and Evolvable Machines*, vol. 19, no. 1-2, pp. 9–51, 2018.

[34] F. Zhang, Y. Mei, and 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*. IEEE, 2019, pp. 1366–1373.

[35] J. R. Koza and R. Poli, "Genetic programming," in *Search Methodologies*. Springer, 2005, pp. 127–164.

[36] T. T. H. Dinh, T. H. Chu, and Q. U. Nguyen, "Transfer learning in genetic programming," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2015, pp. 1145–1151.

[37] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[38] D. Karunakaran, Y. Mei, and M. Zhang, "Multitasking genetic programming for stochastic team orienteering problem with time windows," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, 2019, pp. 1598–1605.

[39] J. Park, Y. Mei, S. Nguyen, G. Chen, and M. Zhang, "Evolutionary multitask optimisation for dynamic job shop scheduling using niched genetic programming," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 739–751.

[40] J. Lin, H.-L. Liu, K. C. Tan, and F. Gu, "An effective knowledge transfer approach for multiobjective multitasking optimization," *IEEE Transactions on Cybernetics*, 2020. Doi: 10.1109/TCYB.2020.2969025.

[41] J. Ding, C. Yang, Y. Jin, and T. Chai, "Generalized multitasking for evolutionary optimization of expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 1, pp. 44–58, 2017.

[42] ——, "Generalized multitasking for evolutionary optimization of expensive problems," *IEEE Transations on Evolutionary Computation*, vol. 23, no. 1, pp. 44–58, 2019.

[43] H. Li, Y.-S. Ong, M. Gong, and Z. Wang, "Evolutionary multitasking sparse reconstruction: Framework and case study," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 733–747, 2018.

[44] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y.-S. Ong, K.-C. Tan, and A. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE Transactions on Cybernetics*, vol. 49, no. 9, pp. 3457–3470, 2018.

[45] L. Zhou, L. Feng, J. Zhong, Y.-S. Ong, Z. Zhu, and E. Sha, "Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, 2016, pp. 1–8.

[46] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 858–869, 2019.

[47] 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*, 2020. Doi: 10.1109/TCYB.2020.3024849.

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

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

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

[51] B. Da, Y.-S. Ong, L. Feng, A. K. Qin, A. Gupta, Z. Zhu, C.-K. Ting, K. Tang, and X. Yao, "Evolutionary multitasking for single-objective continuous optimization: Benchmark problems, performance metric, and baseline results," *arXiv:1706.03470*, 2017.

[52] M. Fisher and A. Raman, "Reducing the cost of demand uncertainty through accurate response to early sales," *Operations research*, vol. 44, no. 1, pp. 87–99, 1996.

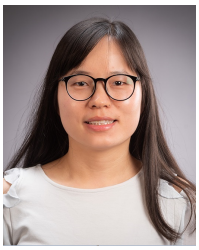
[53] M. Pinedo, *Scheduling*. Springer, 2012, vol. 29.

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

[55] 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 *Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2020, pp. 214–230.

[56] ——, "Guided subtree selection for genetic operators in genetic programming for dynamic flexible job shop scheduling," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2020, pp. 262–278.

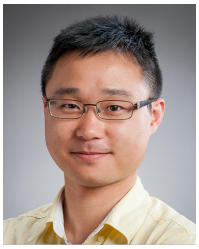
[57] C. W. Pickardt, T. Hildebrandt, J. Branke, J. Heger, and B. Scholz-Reiter, "Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems," *International Journal of Production Economics*, vol. 145, no. 1, pp. 67–77, 2013.
[58] J. Branke, T. Hildebrandt, and B. Scholz-Reiter, "Hyper-heuristic evolution of dispatching rules: a comparison of rule representations," *Evolutionary Computation*, vol. 23, no. 2, pp. 249–277, 2015.
[59] M. E. Leusin, E. M. Frazzon, M. Uriona Maldonado, M. Kück, and M. Freitag, "Solving the job-shop scheduling problem in the industry 4.0 era," *Technologies*, vol. 6, no. 4, p. 107, 2018.



Fangfang Zhang (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. degrees from Shenzhen University, Shenzhen, China, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree in computer science with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand.

She has over 25 journal and conference papers. Her current research interests include evolutionary computation, hyper-heuristic learning / optimisation, job shop scheduling, and multitask learning.

Ms. Zhang is a member of the IEEE Computational Intelligence Society and Association for Computing Machinery, and has been severing as reviewers for top international journals such as the IEEE Transactions on Evolutionary Computation and the IEEE Transactions on Cybernetics, and conferences including the Genetic and Evolutionary Computation Conference and the IEEE Congress on Evolutionary Computation. She is also a committee member of the IEEE NZ Central Section.



Yi Mei (M'09-SM'18) received the B.Sc. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively.

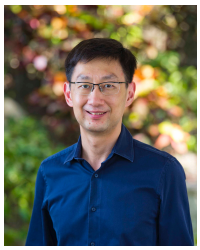
He is a Senior Lecturer with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. He has more than 100 fully referred publications, including the top journals in EC and Operations Research, such as IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, IEEE 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 journal. He serves as a reviewer of over 30 international journals.

Su Nguyen (M'13) received his Ph.D. degree in Artificial Intelligence and Data Analytics from Victoria University of Wellington, New Zealand, in 2013.

He is a Senior Research Fellow and an Algorithm Lead with the Centre for Data Analytics and Cognition, La Trobe University, Melbourne, VIC, Australia. His expertise includes evolutionary computation (EC), simulation optimization, automated algorithm design, interfaces of AI/OR, and their applications in logistics, energy, and transportation. He has more than 70 publications in top EC/OR peer-reviewed journals and conferences. 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.

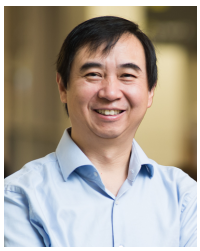
Dr. Nguyen was the Chair of IEEE Task Force on Evolutionary Scheduling and Combinatorial Optimisation from 2014 to 2018 and is a member of the IEEE CIS Data Mining and Big Data Technical Committee. He delivered technical tutorials about EC and AI-based visualization at Parallel Problem Solving from Nature Conference in 2018 and IEEE World Congress on Computational Intelligence in 2020. He served as an Editorial Member of 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 journal.



Kay Chen Tan (SM'08-F'14) received the B.Eng. (First Class Hons.) degree in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is currently a Chair Professor with the Department of Computing at the Hong Kong Polytechnic University, Hong Kong SAR. He has published over 300 refereed articles and 7 books. Dr. Tan was the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2015 to 2020, and currently serves as the Editorial Board Member of over 10 journals.

Prof. Tan is an IEEE Fellow, an Honorary Professor at University of Nottingham in UK, and the Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.



Mengjie Zhang (M'04-SM'10-F'19) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively.

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. 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. She has published over 500 research papers in refereed international journals and conferences.

Prof. Zhang was the Chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, the IEEE CIS Emergent Technologies Technical Committee, and the 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. She is also a Committee Member of the IEEE NZ Central Section. He is a Fellow of the Royal Society of New Zealand and an IEEE Distinguished Lecturer.