Task Relatedness Based Multitask Genetic Programming for Dynamic Flexible Job Shop Scheduling

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

***Abstract*—Multitasking learning has been successfully used in handling multiple related tasks simultaneously. In reality, there are often many tasks to be solved together, and the relatedness between them is unknown in advance. In this paper, we focus on multitask genetic programming for the dynamic flexible job shop scheduling problems, and address two challenges. The first is how to measure the relatedness between tasks accurately. The second is how to select task pairs to transfer knowledge during the multitask learning process. To measure the relatedness between dynamic flexible job shop scheduling tasks, we propose a new relatedness metric based on the behaviour distributions of the variable-length genetic programming individuals. In addition, for more effective knowledge transfer, we develop an adaptive strategy to choose the most suitable assisted task for the target task based on the relatedness information between tasks. The findings show that in all of the multitask scenarios studied, the proposed algorithm can substantially increase the effectiveness of the learned scheduling heuristics for all the desired tasks. The effectiveness of the proposed algorithm has also been verified by the analyses of task relatedness and structures of the evolved scheduling heuristics, and the discussions of population diversity and knowledge transfer.**

***Index Terms*—Multitask Learning, Genetic Programming, Dynamic Flexible Job Shop Scheduling, Adaptive Strategy, Assisted Task Selection.**

I. Introduction

Multitask learning aims at solving a number of related tasks simultaneously [1]. Multifactorial optimisation towards evolutionary multitasking (MFEA) was developed as a paradigm in [2], [3] for solving multiple tasks simultaneously with evolutionary algorithms. MFEA has been successfully used to solve a variety of problems including continuous numeric optimisation [3]–[6], feature selection [7], symbolic regression [8], sparse reconstruction [9], and timetabling [10]. Multitask learning is a type of transfer learning [11] that tackles multiple tasks simultaneously by knowledge transfer between tasks. The positive knowledge transfer between tasks is critical to MFEA's success. The negative transfer between unrelated tasks can even worsen the multitask solving ability, especially when there are more than two tasks. The relatedness [1], [12] between tasks plays an important role in transferring knowledge between tasks. Normally, transferring knowledge from more related tasks can lead to more positive transfer [13]. However, how to effectively transfer knowledge between tasks based on their relatedness is still an open issue, especially for combinatorial optimisation [14].

To fill this gap, this paper proposes a new algorithm for multitask dynamic combinatorial optimisation problems, which can adaptively select assisted tasks for a target task based on their relatedness. This paper takes dynamic flexible job shop scheduling (DFJSS) [15] as the investigated problem. DFJSS is an important combinatorial optimisation problem with wide real-world applications [16], and has been extensively studied in both academia and industry [17]–[20]. Solution optimisation methods such as evolutionary algorithm [21] cannot handle the dynamic events efficiently due to their high computational cost [22]. Scheduling heuristics such as dispatching rules [23] can make real-time decisions efficiently. However, manually designing effective dispatching rules heavily relies on domain experts, which are not always available. Genetic programming (GP) has shown success in learning scheduling heuristics automatically for DFJSS [19], [24], [25]. A multitask GP method [26] has been previously developed to solve multiple DFJSS tasks simultaneously. Its main contribution is to use one subpopulation for solving each task, and exchange information between different subpopulations to achieve transfer learning between tasks. However, the relatedness between tasks in [26] is ignored.

There are two main design issues to consider relatedness between DFJSS tasks with GP. The first is to design a proper relatedness measure between tasks for guiding knowledge transfer more effectively during the multitask GP learning/search process. The second is to design the multitask GP search process, particularly the manner of knowledge transfer, to take advantage of the relatedness between tasks and achieve better multitask optimisation performance.

To measure the relatedness between tasks, most existing

Manuscript received XXX; revised XXX and XXX; accepted XXX. This work is supported 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. (*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, yi.mei, 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, The 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

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2022.3199783

studies [27]–[30] proposed various relatedness measures for continuous optimisation problems based on the shape of their fitness landscape (e.g., objective function and constraints). These measures are not directly applicable to DFJSS, since some information required for calculating the measures, such as the arrival time and due date of all the jobs, are not exactly known in DFJSS due to dynamic job arrivals. A more generic relatedness measure [31] randomly samples a large number of individuals and evaluates them on each task. Then, the relatedness between two tasks is defined as the Spearman (rank) correlation between the fitness of these individuals on the two tasks. This generic measure is applicable to DFJSS. However, it might not be effective in guiding the multitask GP search, since it provides only a fixed value about the relatedness between tasks. A more proper relatedness measure needs to identify the relatedness between tasks with respect to the distribution of the current individuals during the GP process, so that the knowledge transfer is more consistent with the current individual distribution rather than being constant throughout the process. There are several recent studies [13], [32], [33] to estimate the relatedness between tasks in an online fashion and use it to guide the knowledge transfer. However, these measures are based on vector-based fixed-length individual representation, and are thus not applicable to GP with tree-based variable-length representation. To the best of our knowledge, there is no existing relatedness measure for multitask GP with tree-based variable-length representation.

The second issue is typically addressed by adaptively selecting the assisted tasks (the tasks to transfer knowledge from) for each task based on their relatedness. However, how to effectively select assisted tasks is still an open question.

To address the above issues, this work aims to develop an effective multitask GP algorithm to evolve scheduling heuristics for solving multiple DFJSS tasks simultaneously. Specifically, the paper has the following contributions:

1) We develop a new relatedness measure between tasks for GP with tree-based variable-length representation. We first use the phenotypic characterisation [34], [35] to represent the behaviour of each GP individual with a fixed-length vector. Then, we define the relatedness measure between tasks using the Kullback–Leibler divergence [36] between the distributions of the behaviours of the current individuals for different tasks. To the best of our knowledge, this is the first relatedness measure between tasks for GP with variable-length representation. The newly developed relatedness measure is generic, and can be applied to any problem with a proper phenotypic characterisation of individuals. This allows the proposed algorithm to handle multitask learning without knowing the relatedness between tasks in advance (the new algorithm will automatically calculate the relatedness between tasks), which is usually the case in real-world applications.
2) We propose a new multitask GP algorithm that adaptively selects assisted tasks based on the relatedness measure. The proposed algorithm can help practitioners obtain effective scheduling heuristics for a set of tasks simultaneously by automatically measuring the relatedness of different tasks, which is important in real-world applications. To the best of our knowledge, this is the first attempt to reduce the negative transfer in the dynamic combinatorial optimisation problems by adaptively selecting assisted tasks based on relatedness.
3) We verify the effectiveness of the newly proposed multitask GP algorithm by comparing it with the state-of-the-art algorithms on a wide range of DFJSS problems. The results show that the newly proposed algorithm significantly outperforms the state-of-the-art, and can learn more effective scheduling heuristics for all tasks.

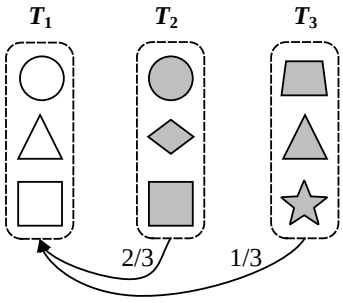


Fig. 1. An example of the relatedness (2/3 and 1/3) of the candidate assisted tasks (T_2 and T_3) to a target task (T_1) in multitask learning, where T_i indicates i^{th} task.

II. Background

A. Multifactorial Evolutionary Algorithm

The MFEA was proposed [3] for solving related tasks simultaneously with evolutionary algorithms. The use of the skill factor τ to assign individuals to tasks is a key feature of MFEA. The role that an individual excels at is represented by the skill factor. The crossover between individuals with different skill factors explicitly realises knowledge sharing between tasks. The effectiveness of the knowledge transfer arises the attention of researchers recently. The knowledge transfer between tasks can be positive or negative [14], [27], [28], [37]. From the perspective of relatedness between tasks, if the tasks are related, it is more likely to have *positive transfer*, while if the tasks are less related, the knowledge transfer between tasks may not help or even be harmful, i.e., *negative transfer*. However, in the traditional MFEA, for implementing knowledge transfer, it randomly chooses the assisted tasks by randomly selecting parents, which may not be an effective way for knowledge transfer, especially when there are more than two tasks.

An example of the relatedness of the candidate assisted tasks to a target task in multitask learning is shown in Fig. 1, where the relatedness is defined as the proportion of common shapes between two tasks to the total number of shapes for a task. In this example, there are three tasks (i.e., T_1, T_2, T_3), and the current target task is T_1 with three blank shapes. We can see that T_2 has two common shapes (i.e., circle and square) with T_1 and has a relatedness value of 2/3, while T_3 has only one common shape (i.e., triangle) with T_1 and its relatedness value is 1/3. It is obvious that it is easier to fill the blank shapes for T_1 with the shapes from T_2 than those from T_3.

This means that for a target task, it is better to learn more from more related tasks to improve its search effectiveness and efficiency. Note that each task can be a target task or a candidate assisted task at different stages of multitask learning.

B. Multitask Dynamic Flexible Job Shop Scheduling

In DFJSS, a number of n jobs need to be processed by a set of m machines. Each job j has a sequence of operations $(O_{j1}, O_{j2}, ..., O_{jl_j})$. The number of operations ranges from 1 to m ($1 \leq l_j \leq m$), which varies from one job to another. Each operation O_{ji} can only be done by one of its optional machines $\pi(O_{ji})$, and the time required to process it $\delta_M(O_{ji}), M \in \pi(O_{ji})$ depends on the machine [38]. New jobs can arrive dynamically [23], [25], [39]. The information about a new job is not known before the job comes to the job shop. The following are the constraints of DFJSS:

- A machine can process at most one operation at a time.
- Each operation is processed by one of its candidate machines, i.e., it cannot be split.
- An operation cannot be started until all its preceding operations have been completed.
- Once an operation is underway, it cannot be halted or paused until it is finished.

In real-world applications, it is common that different customers have different scheduling objectives [40]. One may prefer to minimise flowtime to reduce the total cost. Others may minimise tardiness to hand out products to customers as early as possible. These optimisation objectives are related to improving the production efficiency, and may be solved simultaneously in a multitask framework. Three commonly used objectives are considered in this paper, as shown below:

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

where 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, w_j represents the weight (importance) of job J_j, and n represents the number of jobs.

C. Genetic Programming for DFJSS

GP has been successfully used to learn scheduling heuristics for combinatorial optimisation problems as a hyper-heuristic approach [24], [41]–[45] because of its flexible representation. In particular, tree-based GP [20], [46], [47] has been the most widely used GP representation, since the trees are naturally "mathematical functions" that can produce a real value, which is perhaps the easiest way to calculate the priority of a rule for job shop scheduling. Tree-based GP has shown advantage over other representations [48], [49] including linear representation and neural network. This means that the structures of heuristics (i.e., the optimal structure is normally unknown) do not need to be defined in advance. The tree-based GP individual can be considered as a priority function to give priority values for candidates efficiently, and tend to be interpretable. The goal of GP for DFJSS is to automatically evolve effective scheduling heuristics by selecting and combining terminals (i.e., low-level heuristics) and functions properly.

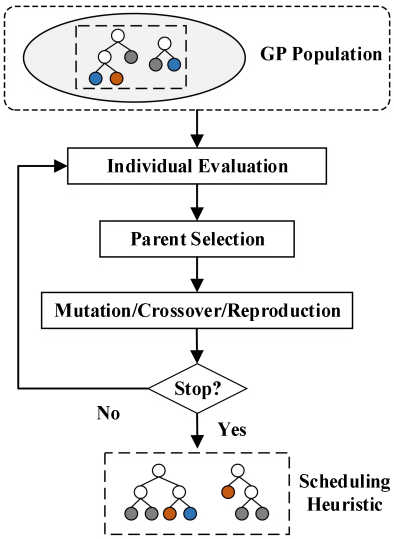


Fig. 2. The flowchart of GP for learning scheduling heuristics for DFJSS.

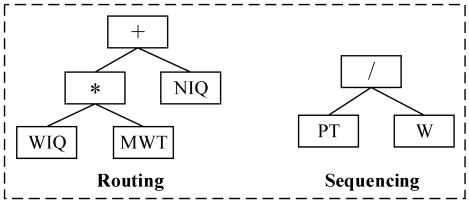


Fig. 3. An example of a routing and a sequencing rule for DFJSS.

Fig. 2 shows the flowchart of using GP with multi-tree representation to learn heuristics for DFJSS [50]. Each GP individual consists of two trees (i.e., one for evolving routing rule for machine assignment, and the other for evolving sequencing rule for operation sequencing). The fitness of an individual depends on the collaboration between the two rules. The coloured circles represent different terminals, and the uncoloured circles stand for functions. GP starts with randomly initialised individuals, and then all the individuals are evaluated with the training instances. When generating new offspring for the next generation, genetic operators including mutation, crossover, and reproduction are applied to the selected parents. Finally, if the stop criterion is met, the best individual so far is considered as the best evolved scheduling heuristic for DFJSS. Otherwise, the individuals are evaluated again, and turn into a new evolutionary loop.

An example of the scheduling heuristics including a routing rule and a sequencing rule is shown in Fig. 3. The routing rule is WIQ * MWT + NIQ, where WIQ is the required time of a machine to finish its allocated operations, MWT is the waiting time for a machine to become idle, and NIQ is the number of operations a machine has. The machine with the highest priority calculated by the routing rule is selected to process the operation. Similarly, the operation with the highest priority calculated by the sequencing rule (i.e., PT / W, where PT is the processing time, and W is the weight of an operation) is selected to be processed next on the idle machine.

D. Terminology

To avoid confusion due to ambiguity, below are the definitions of the terms commonly used in this paper:

- A *job* is a specific object that needs to be processed by a machine.
- An *operation* is a part of the job. A job is finished when all its operations are processed.
- An *instance* is a specific simulation with a random seed.
- A *scenario* represents a specific problem to be solved with the instances generated by the same problem configuration, e.g., the same objective and utilisation level. An instance is an example of a scenario.
- A *task* is a specific problem to be solved which is represented by a scenario. Solving the problems in different scenarios simultaneously is a multitask learning problem.

III. Literature Review

The study of the relationship between tasks can not only help develop effective multitask algorithms, but also investigate how a multitask algorithm works. A theoretical and empirical study can be found in [51]. However, it is still an open question, especially in combinatorial optimisation [14]. This paper groups the existing studies about the relationship between tasks in multitask learning into two categories, i.e., *task relatedness* and *task adaptation*.

1) Task relatedness: Task relatedness information is normally used to build multitask learning benchmarks for testing the multitask algorithms or to improve the quality of shared knowledge in multitask learning. An effective strategy was proposed to detect the behaviour of each task and to investigate how the success (or failure) of some tasks affects the performance of the other tasks in multitask learning in [52]. Multitask multiobjective and single objective test problems were proposed in [31] and [53] based on the characteristics of benchmark functions such as the fitness landscape and optimal solution, respectively. The transfer ratio was adjusted to minimise the negative interactions between distinct tasks based on the relatedness between benchmark function tasks in multitask settings [32], [33]. These kinds of studies take the relatedness between tasks as prior knowledge.

In [54], relatedness measure was investigated for continuous numeric optimisation based on the distance between best solutions with fixed length, fitness rank correlation, and fitness landscape analyses. However, with tree-based and variable-length representation of GP, it is not trivial to calculate the distance between solutions directly (e.g., different trees can correspond to the same rule). The extra fitness evaluations for calculating fitness rank correlation are time-consuming in DFJSS, and the fitness landscape information is not available in DFJSS. An adaptive archive-based assisted selection strategy was proposed to select a suitable assisted task for a given task by considering the relatedness between tasks in [13]. However, the proposed task relatedness measure works on a continuous search space with fixed length solutions rather than discrete search space with tree-based and variable-length heuristics. In addition, the way to adapt probability for tasks based on the number of successful transfer is not applicable to DFJSS. The reason is that the individuals in the previous generation are not comparable with the individuals in the current generation due to the rotation of training instance which is a commonly used way to improve the generalisation of evolved scheduling heuristics in dynamic scheduling [46]. These kinds of studies [32], [33], [55] try to measure the relatedness between tasks, and use the relatedness information to improve the effectiveness of multitask algorithms.

2) Task adaptation: Task adaptation aims to convert the search space of one task being good for another task to improve the ability of positive transfer between tasks. A linearised domain adaptation was developed to transform the search space of a simple task to the search space similar to its constitutive complex task [56]. The transformed search space resembled a high correlation with its constitutive task and provided a platform for efficient knowledge transfer. All tasks were transformed to new space while maintaining the same geometric properties of solutions in [37]. The individuals were transformed to fit another task with a task space mapping strategy for generating higher-quality offspring in [57]. These kinds of studies [58] focus on adapting solutions of one task being good for other tasks rather than working on the relatedness between tasks directly.

As a starting point for investigating task relationship in dynamic problem with GP hyper-heuristic, this paper focuses on the investigations on task relatedness for selecting assisted tasks in DFJSS with tree-based and variable-length GP individuals. DFJSS is a discrete minimisation problem. Unlike continuous numeric functions with known optimal solutions, the characteristics of the problem such as the fitness landscape information are not known. The existing studies mainly work on vector-based evolutionary algorithms with fixed-length solutions, and their relatedness measures are not directly applicable for GP individuals with tree-based and variable-length representation. In addition, the training instances are rotated at each generation [46], which is commonly used to improve the generalisation of the evolved scheduling heuristics. All the above characteristics make it challenging to measure the relatedness between tasks for solving DFJSS with GP.

IV. The New Multitask Genetic Programming with Adaptive Assisted Task Selection

A. The Proposed Algorithm's Framework

Algorithm 1 presents the main structure of the proposed algorithm. The input is a list of k tasks to be solved. The output is a collection of k best learned scheduling heuristics (h_1^*, h_2^*, ... , h_k^*) for each task. We use multiple subpopulations (i.e., $Subpop_1$, $Subpop_2$, ... , $Subpop_k$) to solve the k tasks simultaneously but keep knowledge sharing between them. $subpopsize_i$ represents the number of individuals of subpopulation i, respectively. The fitness of a heuristic h_i is represented by $fitness_{h_i}$. On one hand, each subpopulation is autonomous, and individuals within different subpopulations are evolved to solve different tasks. Each subpopulation can be thought of as a group of individuals with the same skill factor in MFEA. Different subpopulations, on the other hand, help each other by sharing their knowledge with others, which is realised with the genetic operator.

Algorithm 1: The Proposed Algorithm's Framework

Input : k tasks T_1, T_2, ... , T_k
Output: The learned scheduling heuristics for each task h_1^*, h_2^*, ... , h_k^*
// Initialise the k subpopulations at random
1: set h_1^*, h_2^*, ... , $h_k^* \leftarrow null$
2: set $fitness_{h_1^*}$, $fitness_{h_2^*}$, ... , $fitness_{h_k^*} \leftarrow +\infty$
3: $gen \leftarrow 0$
4: **while** $gen < maxGen$ **do**
// Fitness evaluation
5: **for** *i = 1 to* k **do**
6: **for** *j = 1 to* $subpopsize_i$ **do**
7: Calculate $fitness_{h_j}$ based on fitness function of T_i
8: **end**
9: **for** *j = 1 to* $subpopsize_i$ **do**
10: **if** $fitness_{h_j} < fitness_{h_i^*}$ **then**
11: $h_i^* \leftarrow h_j$
12: **end**
13: **end**
14: **end**
15: **if** $gen < maxGen - 1$ **then**
// Evolution and offspring generation
16: **for** *i = 1 to* k **do**
17: **for** *j = 1 to* $subpopsize_i$ **do**
18: **Calculate phenotypic characterisation for individual** h_j **in a set of decision situations**
19: **end**
20: **end**
21: **Build the behaviour matrix for each subpopulation** $Matrix_1$**,** $Matrix_2$**, ... ,** $Matrix_k$
22: **Calculate the relatedness between each pair of tasks based on the built behaviour matrices**
23: **for** *i = 1 to* k **do**
24: **if** *rand* $\leq$ *rmp* **then**
25: **Choose one task as assisted task** T_a **proportional to the calculated relatedness (task selection strategy)**
26: **for** *j = 1 to* $subpopsize_i$ **do**
27: **Choose the first parent from** $Subpop_i$ **with tournament selection**
28: **Choose the second parent from** $Subpop_a$ **with tournament selection**
29: Produce one offspring with the origin-based offspring reservation strategy [26]
30: **end**
31: **else**
32: Choose two parents from $Subpop_i$ with tournament selection, and produce two offspring by GP crossover
33: **end**
34: **end**
35: **end**
36: $gen \leftarrow gen + 1$
37: **end**
38: **return** h_1^*, h_2^*, ... , h_k^*

At the initialisation stage, each subpopulation is randomly initialised (line 1). The individuals in different subpopulations are evaluated, and the best individual for each task is recorded independently (lines 5–14) *during the evaluation process*. At each generation, we use a different training instance for fitness evaluation to improve generalisation. To calculate the fitness of an individual, the individual is applied to the training instance to generate a schedule. Then, its fitness is defined as the objective value of the generated schedule. Then, the phenotypic characterisation of each individual in each subpopulation is calculated with a set of decision situations along with the routing and sequencing reference rules (lines 16–20). The behaviour matrices for subpopulations are built based on the phenotypic characterisations of individuals in the corresponding subpopulation (line 21). The relatedness between each pair of tasks is calculated based on the built behaviour matrices (line 22). The crossover operator is used *during the evolution stage* to share knowledge between tasks. The offspring of each subpopulation are produced for the next generation with the assisted task if the knowledge sharing condition is met ($rand \leq rmp$) (lines 25–30). Specifically, for each target task, one of the candidate tasks $Subpop_a$ is selected as the assisted task according to the proposed assisted task selection strategy. When generating offspring, the individuals from the current subpopulation ($Subpop_i$) and the assisted subpopulation ($Subpop_a$) are used as the parents to realise knowledge transfer from the assisted tasks. If the knowledge sharing condition is not met ($rand > rmp$), two offspring will be generated using the traditional GP crossover (line 32). Finally, the best evolved scheduling heuristics for all the tasks obtained at the last generation are returned.

Compared with the existing multitask GP algorithm [26], the proposed algorithm contains four new components, as highlighted in bold in Algorithm 1. The first new component (line 18) calculates the phenotypic characterisation of individuals. The second one (line 21) builds the behaviour matrix for each task. The third one (line 22) calculates the relatedness between tasks based on the behaviour matrices. The last new component (line 25) adaptively selects the assisted tasks for each task based on the calculated task relatedness.

B. The Relatedness Between Tasks

Since the individuals in each subpopulation tend to specialise to the corresponding task over generations, the characteristics of a task can be represented by the behaviour of the individuals [33]. If the representative individuals for two tasks show similar behaviour, then the tasks are related to each other. The degree of relatedness between tasks depends on how similarly the representative individuals behave. In this paper, for each task, we use all the individuals in its subpopulation as the representative individuals.

1) Phenotypic Characterisation: Since a GP individual has the tree-based and variable-length representation, the individuals are not comparable directly. In this paper, we use the phenotypic characterisation [34], [59] to represent the behaviour of GP individuals, and compare individuals by their phenotype rather than genotype. This corresponds to line 18 of Algorithm 1. The phenotypic characterisation of a GP individual is a vector of the rank number of machines or operations which represents its decision making behaviour based on a set of decision situations. Briefly speaking, the phenotypic characterisation of a decision is defined as the rank of the machine or operation selected by an individual in the sorted list of the reference rules (i.e., WIQ (work in the queue) for routing decisions, and SPT (shortest processing time) for sequencing decisions). A routing (sequencing) decision situation includes a temporal job shop state, the given operation (machine) and the candidate machines (operations). The dimension of the phenotypic characterisation is equal to the number of decision situations. A smaller distance between the phenotypic characterisations of two individuals indicates that they behave more similarly. For the phenotypic

TABLE I
AN EXAMPLE OF CALCULATING THE PHENOTYPIC CHARACTERISATION OF A SEQUENCING RULE WITH FOUR DECISION SITUATIONS, EACH WITH THREE CANDIDATE OPERATIONS.

Decision Situation	Reference Rule	Sequencing Rule	PC_i
1 (O_1)	3	2	
1 (O_2)	2	3	1
1 (O_3)	1	1	
2 (O_1)	1	3	
2 (O_2)	3	1	3
2 (O_3)	2	2	
3 (O_1)	2	3	
3 (O_2)	3	2	1
3 (O_3)	1	1	
4 (O_1)	1	3	
4 (O_2)	2	1	2
4 (O_3)	3	2	

PC_i indicates the i^{th} dimension of phenotypic characterisation.

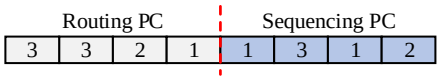


Fig. 4. An example of the phenotypic characterisation of a GP individual.

characterisations of individuals, we randomly sample a set of routing and sequencing decision situations with the same number of candidate machines or operations.

Table I shows an example of how to calculate the phenotypic characterisation of a sequencing rule with four decision situations, each consisting of three candidate operations. There are four steps to get the phenotypic characterisation. First, a reference rule (i.e., SPT, shortest processing time) is used to make a standard for all examined rules. The reference rule is used to rank the operations in each situation. For example, the operations O_1, O_2, and O_3 are ranked by the reference rule as 3, 2, 1 in the first decision situation. Second, the examined sequencing rule ranks the same operations in the same decision situation. For example, the sequencing rule ranks the operations O_1, O_2, and O_3 in the first decision situation as 2, 3, and 1. Third, the corresponding rank by the reference rule of the most prior operation by the examined sequencing rule is used as the phenotype of the examined rule in this decision situation. For example, the most prior operation of the sequencing rule is O_3, and O_3 is ranked as 1 by the reference rule. Therefore, the phenotype of the examined rule in this decision situation is 1. We can calculate the phenotypic characterisation for all the decision situations. Finally, we concatenate all the phenotypic values together to form the phenotypic vector of the examined sequencing rule. In the example of Table I, the phenotypic vector of the sequencing rule is (1, 3, 1, 2). The phenotypic characterisation converts the tree-based individuals into a fixed-length vector space, making them comparable.

Similarly, we can also calculate the phenotypic vector of a routing rule. The routing reference rule is WIQ (i.e., the least workload of a machine). Since each individual consists of a routing rule and a sequencing rule, we concatenate the phenotypic vector of the routing rule and the sequencing rule as the phenotypic vector of a GP individual. An example of the phenotypic vector of a GP individual is shown in Fig. 4. The individuals with both similar routing and sequencing behaviours tend to have more similar phenotypic vectors.

2) Behaviour Matrix: For each task, we build a behaviour matrix to capture the characteristics of a task by using the phenotypic vectors of all the individuals for that task (line 21 of Algorithm 1). An example of the behaviour matrix for task T_i is shown as follows, where DS_i means the i^{th} decision situation. In this example, the behaviour matrix is based on three individuals and four decision situations, which is a 3 * 4 matrix. Each row indicates the phenotypic vector of an individual, while each column represents the behaviour of different individuals in the same decision situation.

$$Matrix_i = \begin{array}{c} \\ Ind_1 \\ Ind_2 \\ Ind_3 \end{array} \begin{array}{c} \begin{array}{cccc} DS_1 & DS_2 & DS_3 & DS_4 \end{array} \\ \begin{pmatrix} 3 & 2 & 3 & 1 \\ 2 & 3 & 1 & 2 \\ 1 & 1 & 2 & 2 \end{pmatrix} \end{array}$$

3) Calculate the Relatedness Between Tasks: For each task, the distribution of the behaviours of the representative individuals can reflect the characteristics of the task. Therefore, we define the relatedness between tasks based on the difference between the distributions of the individual behaviours. This corresponds to line 22 of Algorithm 1.

Kullback–Leibler divergence, D_{KL} (also called relative entropy), is a measure of how one probability distribution is different from another probability distribution. Due to the property of D_{KL}, it is a good candidate to measure the relatedness between tasks with the probability of the decisions in each decision situation. It is noted that the behaviour matrices are discrete probability distributions. Assume there are two discrete probability distributions P and Q, the relatedness between them based on D_{KL} can be defined as follows.

$$D_{KL}(P||Q) = \sum_{x \in X} P(x) log(\frac{P(x)}{Q(x)}) \quad (1)$$

We use Eq. (1) to calculate the D_{KL} for each decision situation, and use the average value across all the decision situations as the relatedness value between two tasks. The D_{KL} value indicates how similar the individuals for different tasks behave in the same decision situation.

An example of the calculation of the Kullback–Leibler divergence value of two tasks is shown as follows. There are two tasks (i.e., P and Q), and each task has three representative individuals. Assume the phenotypic vectors of the individuals for task P and Q in one (same) decision situation are $\overrightarrow{P}$ = (1, 2, 3) and $\overrightarrow{Q}$ = (1, 1, 1), respectively. Thus, the number of occurrences of 1, 2, 3 in $\overrightarrow{P}$ are $\overrightarrow{OP}(1) = 1$, $\overrightarrow{OP}(2) = 1$, and $\overrightarrow{OP}(3) = 1$. For $\overrightarrow{Q}$, we can have $\overrightarrow{OQ}(1) = 3$, $\overrightarrow{OQ}(2) = 0$, and $\overrightarrow{OQ}(3) = 0$. To avoid being divided by zero in Eq. (1), we add 1 to each number of the occurrence values for $\overrightarrow{P}$ and $\overrightarrow{Q}$ as suggested in [60], and thus $\overrightarrow{OP}(1) = 2$, $\overrightarrow{OP}(2) = 2$, $\overrightarrow{OP}(3) = 2$, $\overrightarrow{OQ}(1) = 4$, $\overrightarrow{OQ}(2) = 1$, and $\overrightarrow{OQ}(3) = 1$. The probabilities of all elements in $\overrightarrow{P}$ and $\overrightarrow{Q}$ are $\overrightarrow{P}(1) = \frac{1}{3}$, $\overrightarrow{P}(2) = \frac{1}{3}$, $\overrightarrow{P}(3) = \frac{1}{3}$, $\overrightarrow{Q}(1) = \frac{2}{3}$, $\overrightarrow{Q}(2) = \frac{1}{6}$, and $\overrightarrow{Q}(3) = \frac{1}{6}$. The relatedness between task P and Q in the same decision situation (i.e., $\overrightarrow{P}$

and $\overrightarrow{Q}$) is calculated as follows.

$$\begin{aligned} D_{KL}(P||Q) &= \overrightarrow{P}(1)log(\frac{\overrightarrow{P}(1)}{\overrightarrow{Q}(1)}) + \overrightarrow{P}(2)log(\frac{\overrightarrow{P}(2)}{\overrightarrow{Q}(2)}) \\ &+ \overrightarrow{P}(3)log(\frac{\overrightarrow{P}(3)}{\overrightarrow{Q}(3)}) \\ &= \frac{1}{3}log(\frac{\frac{1}{3}}{\frac{2}{3}}) + \frac{1}{3}log(\frac{\frac{1}{3}}{\frac{1}{6}}) + \frac{1}{3}log(\frac{\frac{1}{3}}{\frac{1}{6}}) \\ &= 0.1 \end{aligned} \tag{2}$$

A smaller $D_{KL}(P||Q)$ means the probability distribution (indicating the decision marking behaviour) in P is more similar with the probability distribution in Q, i.e., the two tasks are more related. Thus, we define the relatedness measure $R(P, Q)$ between tasks P and Q as follows,

$$R(P, Q) = \frac{1}{1 + D_{KL}(P||Q)} \tag{3}$$

where $D_{KL}(P||Q)$ is the Kullback–Leibler divergence value between tasks P and Q. Since $D_{KL}(P||Q) \geq 0$, the range of $R(P, Q)$ is (0, 1]. A larger $R(P, Q)$ implies that P and Q are more related.

It is noted that a number of decision situations are used, and the average D_{KL} based on all the decision situations are used for calculating the relatedness between two tasks. The value of D_{KL} is only a tool to indicate the relatedness between tasks, while the relative relatedness between all assisted tasks is more important for multitask learning.

C. The Assisted Task Selection Strategy

The probabilities of candidate assisted tasks for a target task are designed proportionally to the relatedness between the candidate tasks and the target task. This corresponds to line 25 of Algorithm 1. $R(T_i, T_t)$ is the relatedness value of a candidate task T_i to a target task T_t. We give the more related tasks a higher chance to be selected as the assisted tasks. The probabilities of tasks to be selected (i.e., indicated by $Prob(T_i)$) as the assisted task are shown in Eq. (4).

$$Prob(T_i) = \frac{R(T_i, T_t)}{\sum_{j=1}^{|candidates|} R(T_j, T_t)} \tag{4}$$

First, we sum up the relatedness values of all the candidate assisted tasks of a target task. Then, the probability of each candidate assisted task $Prob(T_i)$ is assigned proportionally to its relatedness to the target task.

V. Experiment Design

In the experiments, we compare the proposed algorithm with existing GP algorithms on a range of multitask DFJSS scenarios. Each scenario contains a training phase and a test phase. During the training process, following the suggestions in [46], [61], the training instance is generated by assigning a new random seed of the simulation at each generation to increase the generalisation of the learned scheduling heuristics. Thus, the number of training instances equals the number of generations of GP. Existing literature [46], [61] have already demonstrated that the strategy of rotating a small number of training instances at each generation can greatly reduce overfitting. In particular, Hildebrandt et al. [61] have shown that using 1 training instance and rotating the training instance at each generation showed the best results for dynamic scheduling [61]. During the test process, the best scheduling heuristic evolved by GP is applied to 50 unseen DFJSS test instances, and the mean objective value over these test instances is calculated as its test performance. The way to generate instances both for training and test is the same, but the instances used in the test process are unseen in the training process. It is noted that the process of applying a heuristic to a training instance and a test instance is the same.

A. Simulation Model

Simulation is widely used in dynamic JSS to measure the quality of the scheduling heuristics [62]. Refer to widely used DFJSS simulation [19], [59], and following the settings in the previous study [24], the simulation assumes that 5000 jobs need to be processed by 10 machines. According to a Poisson process with the rate λ, new jobs will arrive over time. A uniform discrete distribution between 1 and 10 is used to generate the number of operations for a job. The number of candidate machines for a given operation is distributed uniformly between 1 and 10. A uniform discrete distribution with the range [1, 99] is used to assign the processing time of each operation. A job's due date is set to be 1.5 times its production time. *Utilisation level* (p) is a critical factor in simulating job shop scenarios [63]. It is the approximate amount of time that a computer will be occupied. The utilisation level is calculated using Eq. (5), where μ represents the average machine processing time, and P_M represents the likelihood of a job visiting a machine. For example, if each job has two operations, P_M will be 2/10. A busier job shop is associated with a higher degree of utilisation.

$$\lambda = \mu * P_M / p \tag{5}$$

The first 1000 jobs are warm-up jobs, and are omitted from the objective calculation. This simulation gathers information from the following 5000 jobs. When the 6000th job is completed, the simulation ends.

B. Design of Comparisons

We consider three multitask scenarios, each with three tasks. The tasks in each multitask scenario have different objectives, i.e., mean-weighted-tardiness (denoted as WTmean), mean-weighted-flowtime (denoted as WFmean), and max-weighted-tardiness (denoted as WTmax). For the tasks with objectives of WTmean and MFmean, the weights (importance) of 20%, 60%, and 20% of the jobs are set as 1, 2, and 4, respectively [34]. For the tasks with objective of WTmax, the weights of 20%, 60%, and 20% of the jobs are set to 1, 5, 10, respectively. Three utilisation levels (i.e., 0.75, 0.85, and 0.95) are used in different multitask scenarios, since they are three typical distinct configurations in DFJSS [43], [64]. Table II shows the details of the designed multitask scenarios, as represented by the optimised objective and utilisation level.

The baseline algorithm is the GP algorithm, which uses k subpopulations for solving k tasks independently. The second

TABLE II
THE DESIGNED THREE MULTITASK SCENARIOS REFLECTED BY THE OPTIMISED OBJECTIVE AND UTILISATION LEVEL.

Scenario	task 1	task 2	task 3
Scenario 1	<WTmean, 0.75>	<WFmean, 0.75>	<WTmax, 0.75>
Scenario 2	<WTmean, 0.85>	<WFmean, 0.85>	<WTmax, 0.85>
Scenario 3	<WTmean, 0.95>	<WFmean, 0.95>	<WTmax, 0.95>

TABLE III
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 IV
THE PARAMETER SETTINGS IN GP.

Parameter	Value
*Number of subpopulations	k
*Subpopulation size	1000
*The number of elites for each subpopulation	10
*Parent selection	Tournament selection with size 7
*Crossover / Mutation / Reproduction rate	80% / 15% / 5%
**Number of tasks	k
**Population size	1000 * 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	100
The transfer ratio rmp	0.3

* : for GP, MTGP, and ATMTGP with multiple subpopulations only
** : for MFGP with one population only

algorithm to be compared is named MFGP, which combines MFEA [3] with GP. In addition, the state-of-the-art MTGP [26] with multiple subpopulations and offspring reservation crossover is also compared. It is noted that MTGP selects the assisted task randomly for a target task. The proposed multitask GP with an adaptive assisted selection strategy is named ATMTGP, since it involves *assisted task* selection. It is noted that although ATMTGP prefers to select more related tasks as the assisted task, it gives probabilities for selecting other tasks as the assisted task. MTGP randomly selects the assisted task. Thus, there are overlaps of the selected assisted task between MTGP and ATMTGP.

C. Parameter Settings in Genetic Programming

The job shop's features are regarded as GP terminals [65]. 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 are typically used to extract features. Table III shows the details of the terminals. Following the setting in [65], the function set is set to $\{+, -, *, /, max, min\}$. There are two arguments for each function. The "/" function is a protected division that returns one when divided by zero. The max and min functions return the maximum and minimum of their arguments, respectively. The other parameter settings of GP as suggested in [47], [66], are shown in Table IV. The transfer ratio indicates the frequency of a task to learn from other tasks, which is set to 0.3 as suggested in [3]. It is noted that the random parent selection for MFGP was suggested in [3].

For calculating the phenotypic characterisation, following the suggestions in [24], we use 20 routing decision situations with 7 candidate machines, and 20 sequencing decision situations with 7 candidate operations. Thus, the dimension of the phenotypic characterisation of an individual is 40. The decision situations are fixed to get the phenotypic characterisations of all the individuals, either evolved for the same task or different tasks.

VI. RESULTS AND DISCUSSIONS

The results of the compared algorithms are shown in Table V. Friedman's test and Wilcoxon rank-sum test with a significance level of 0.05 are used to examine the performance of the proposed algorithm with 30 independent runs. "Win, Tie, Lose" means how many tasks a compared algorithm is statistically better, similar, or worse than (to) ATMTGP. "Average Rank" based on Friedman's test, displays the algorithm's average rating across all tasks. An algorithm is also compared with the algorithm(s) that come(s) before it in the table. "↑", "↓", and "≈" mean that the corresponding result is statistically significantly better than, worse than or similar to its counterpart in the following results.

A. Quality of the Evolved Scheduling Heuristics

1) The Quality of the Evolved Best Scheduling Heuristics: Table V shows the mean and standard deviation of the objective values on test instances according to 30 independent runs of GP, MFGP, MTGP, and ATMTGP in the three multitask scenarios. The results show that the proposed ATMTGP performs the best with the smallest average rank value according to the Friedman's test. MFGP, which incorporates the idea of MFEA into MFGP directly, performs even worse than GP in most of the scenarios. One possible reason is that the tasks investigated in this paper are quite different since they have different objectives (i.e., an important factor to lead to the search direction), the traditional way that allocates individuals for different tasks via skill factor are not applicable any more. We can also see that by using the framework of multi-population with offspring reservation strategy [26], MTGP achieves better performance than GP and MFGP. These findings are consistent with the observations in [26]. It is noted that MTGP selects the assisted task randomly for a target task.

TABLE V
THE MEAN (STANDARD DEVIATION) OF THE OBJECTIVE VALUES ON TEST INSTANCES OF GP, MFGP, MTGP AND ATMTGP OVER 30 INDEPENDENT RUNS IN THREE MULTITASK SCENARIOS.

Scenario	Task	GP	MFGP	MTGP	ATMTGP
1	<WTmean, 0.75>	27.45(1.19)	29.03(3.15)(↓)	27.51(2.19)(≈)(↑)	26.58(1.06)(↑)(↑)(↑)
	<WFmean, 0.75>	737.59(5.32)	737.44(5.20)(≈)	735.15(5.01)(↑)(↑)	731.30(3.51)(↑)(↑)(↑)
	<WTmax, 0.75>	2602.40(208.92)	2851.52(329.12)(↓)	2470.90(110.57)(↑)(↑)	2499.93(139.77)(↑) (↑)(≈)*
2	<WTmean, 0.85>	76.97(2.32)	77.64(2.64)(≈)	77.10(2.87)(≈)(≈)	75.55(2.41)(↑)(↑)(↑)
	<WFmean, 0.85>	834.37(7.49)	832.87(4.25)(≈)	834.49(7.87)(≈)(≈)	830.45(4.29)(↑)(↑)(↑)
	<WTmax, 0.85>	3570.48(288.81)	3745.45(294.60)(↓)	3486.91(236.47)(↑)(↑)	3441.59(264.23)(↑)(↑)(≈)
3	<WTmean, 0.95>	298.60(7.45)	299.98(8.01)(≈)	298.26(6.76)(≈)(≈)	295.09(6.62)(↑)(↑)(↑)
	<WFmean, 0.95>	1115.75(11.13)	1119.36(12.61)(↓)	1111.11(8.49)(↑)(↑)	1106.07(7.63)(↑)(↑)(↑)
	<WTmax, 0.95>	5605.81(486.15)	5697.84(614.56)(↓)	5419.40(450.42)(≈)(↑)	5413.16(539.69)(↑)(↑)(≈)
	Win / Tie / Lose	0 / 0 / 9	0 / 0 / 9	0 / 3 / 6	N/A
	Average Rank	2.8	3.16	2.26	**1.77**

*: (↑)(↑)(≈) indicates that ATMTGP is significantly better than GP and MFGP, while it has no significant difference from MTGP.

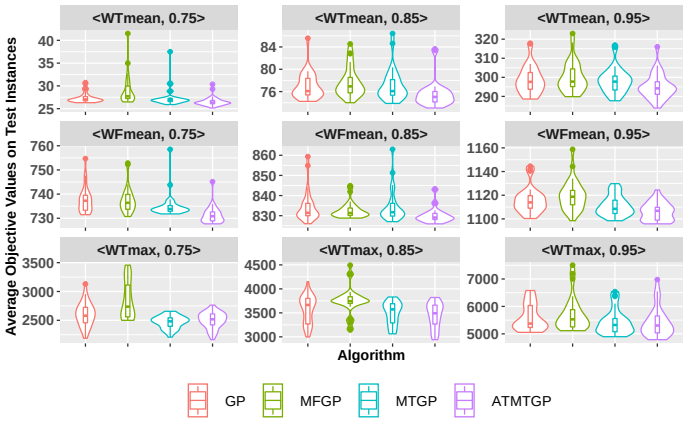


Fig. 5. The violin plot of the average objective values on test instances of GP, MFGP, MTGP, and ATMTGP based on 30 independent runs in three multitask scenarios (each column is a multitask scenario).

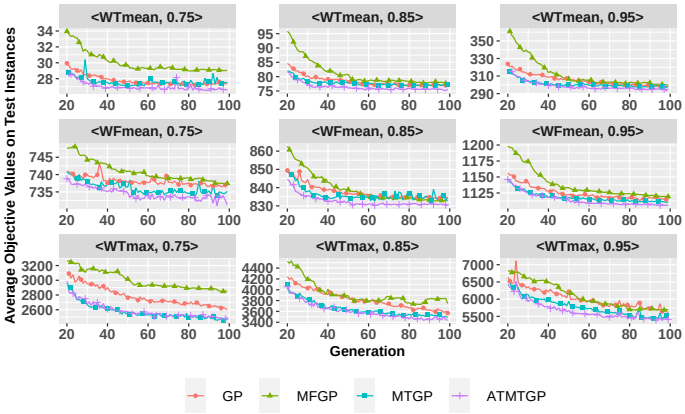


Fig. 6. The curves of the average objective values on test instances of GP, MFGP, MTGP and ATMTGP based on 30 independent runs in three multitask scenarios (each column is a multitask scenario).

ATMTGP performs better than MTGP for six (i.e., <WTmean, 0.75>, <WFmean, 0.75>, <WTmean, 0.85>, <WFmean, 0.85>, <WTmean, 0.95> and <WFmean, 0.95>) out of the nine tasks. This verifies the effectiveness of the proposed assisted task selection strategy, since the only difference between MTGP and ATMTGP is the assisted task selection strategy. We can also find that ATMTGP achieves similar performance as MTGP for the remaining three tasks (i.e., <WTmax, 0.75>, <WTmax, 0.85>, and <WTmax, 0.95>). This is consistent with our expectations, since the tasks with objective WTmean and the tasks with objective WFmean can help each other (i.e., have a higher relatedness). However, the tasks with objective WTmean, and the tasks with objective WFmean cannot help the tasks with objective WTmax (i.e., have a lower relatedness). More details of the relatedness analyses between them will be provided in Section VI-B.

2) The Violin Plot of the Evolved Best Scheduling Heuristics: Fig. 5 shows the violin plot of the average objective values on test instance of GP, MFGP, MTGP and ATMTGP in the three multitask scenarios. It is clear that the proposed algorithm ATMTGP shows its superiority with smaller objective values for the tasks with the objective of WTmean or WFmean among all algorithms as shown in the first and the second rows in Fig. 5. More importantly, in general, the obtained objective values of ATMTGP on unseen instances are smaller than the values achieved by MTGP. This indicates that ATMTGP can detect the useful tasks as the assisted tasks properly for positive transfer, which verifies the effectiveness of the proposed assisted task selection strategy from the perspective of accelerating positive knowledge transfer. It is noted that, for the tasks with the objective of WTmax in all scenarios, ATMTGP does not achieve significantly better performance than MTGP, since these tasks are less related to other tasks [13]. More details of the relatedness analyses between them will be provided in Section VI-B. However, we can still see some promising patterns that the obtained objective values of ATMTGP tend to be smaller than all other algorithms as shown in tasks <WTmax, 0.85> and <WTmax, 0.95>.

3) The Curves of the Average Objective Values on Test Instances: Fig. 6 shows the curves of the average objective values on test instances of GP, MFGP, MTGP, and ATMTGP in the three multitask scenarios. The results show that ATMTGP can achieve better scheduling heuristics than other algorithms from an early stage for the desired tasks (i.e., <WTmean, 0.75>, <WFmean, 0.75>, <WTmean, 0.85>, <WFmean, 0.85>, <WTmean, 0.95> and <WFmean, 0.95>), and keep

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2022.3199783

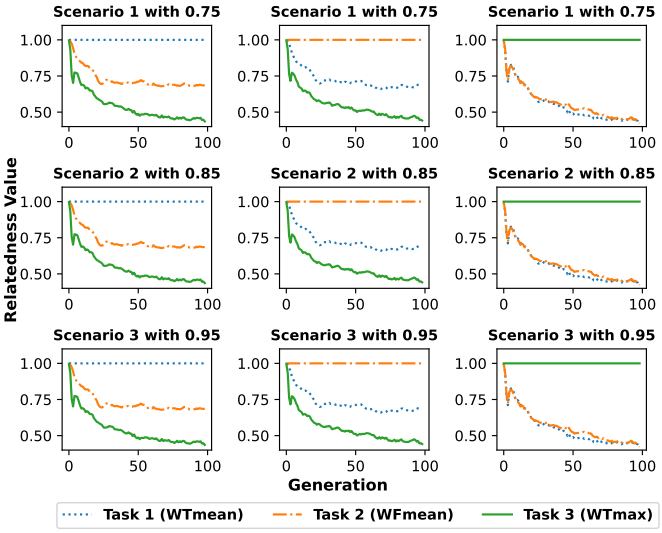


Fig. 7. The curves of the relatedness between tasks of ATMTGP based on 30 independent runs in three multitask scenarios (each row is a multitask scenario, the i^{th} column of the figure represents the relatedness between task T_i and other tasks.).

this advantage throughout the whole evolutionary process. ATMTGP also shows its superiority in the undesired task scenarios. i.e., <WTmax, 0.85> and <WTmax, 0.95>. This further verifies the effectiveness of the proposed assisted task selection strategy.

In summary, the proposed algorithm ATMTGP can achieve better performance in the examined multitask scenarios. It can select not only helpful tasks to realise positive knowledge transfer for a target task if there is a helpful task, but also reduce negative knowledge transfer for a target task if there are no useful assisted tasks.

B. The Relatedness Between Tasks

To investigate why ATMTGP performs better than other algorithms, an important point is to analyse the relatedness between tasks. Fig. 7 shows the curves of the calculated relatedness between tasks during the evolutionary process of ATMTGP in different multitask scenarios according to 30 independent runs. The relatedness between one task and itself is 1, and the main focus is to see the relatedness between other tasks with a target task. Overall, the results show that the relatedness values between tasks are quite large at the beginning of the evolutionary process, and become smaller and smaller along with the generations. This is consistent with the intuition that the individuals in the early generations are not specific to a specific task, and they have a similar quality for all tasks. A specific case is the individuals at generation 0 which are randomly initialised, and they do not make a big difference for different tasks. However, in the latter generations, the individuals are improved, and the individuals for each task in the same subpopulation become good for the corresponding task. Therefore, the relatedness between tasks reduces along with the generations, since the tasks are different and the individuals tend to be good for the corresponding task only. For task 1 (i.e., WTmean) in different scenarios as shown the first column in Fig. 7, the results show that it is more

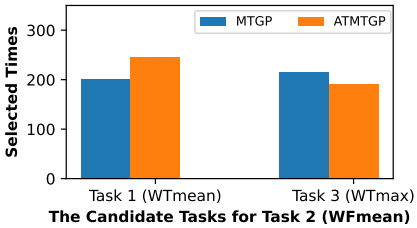


Fig. 8. The bar plot of the number of selected times of the assisted tasks (i.e., task 1 or task 3) for a target task (i.e., task 2) in scenario 2 with utilisation level 0.85 of ATMTGP based on 30 independent runs.

related to task 2 (i.e., WFmean). The task 2 (i.e., WFmean) in different scenarios as shown in the second column, which is more related to task 1 (i.e., WTmean). This is consistent with the findings for task 1, which shows that task 1 and task 2 are related. For task 3, the results show that task 1 and task 2 has the same and low relatedness values with task 3. Taking the performance as shown in Section VI-A into consideration (i.e., task 1 and task 2 do not significantly help task 3), it concludes that both task 1 and task 2 are less related to task 3. For convenience, if the tasks cannot help each other and they have a relatively small relatedness value, we will say that they are not related below. Otherwise, they are related tasks.

C. The Selected Assisted Tasks

To investigate why the proposed ATMTGP performs better than other algorithms, another important point is to investigate whether the assisted tasks are properly selected, i.e., more related tasks have higher chances to be selected as expected. We take scenario 2 with utilisation level 0.85 as an example in this subsection. Fig. 8 shows the bar plot of the number of selected times of candidate assisted tasks (i.e., task 1 and task 3) for the target task 2 in scenario 2 according to 30 independent runs. The results show that ATMTGP can successfully give a higher chance than MTGP to select the related task (i.e., task 1, WTmean) for task 2 (i.e., WFmean). In addition, ATMTGP gives a lower chance than MTGP to select an unrelated task (i.e., task 3, WTmax) for task 2 (i.e., WFmean). The same patterns can also be found in other scenarios. Combining with Section VI-B, these two subsections detail how the proposed algorithm ATMTGP works.

D. Training Time

Table VI shows the mean and standard deviation of the training time of all involved algorithms over 30 independent runs in three multitask scenarios. The training time is the duration of the entire GP evolutionary training process, including the initialisation, heuristic generation by genetic operators and applying heuristics to the training simulations for fitness evaluation in all generations. The results show that there is no significant difference among the compared four algorithms. This means the proposed algorithm ATMTGP does not need extra computational cost to achieve better performance than compared algorithms. It is noted that applying a learned heuristic to make a decision in an unseen test instance takes less than 0s. Therefore, we do not compare the test time of

TABLE VI
THE MEAN (STANDARD DEVIATION) OF THE TRAINING TIME (IN MINUTES) OF GP, MFGP, MTGP, AND ATMTGP OVER 30 INDEPENDENT RUNS IN THREE MULTITASK SCENARIOS.

Scenario	GP	MFGP	MTGP	ATMTGP
1	331(25)	335(24)(≈)	336(23)(≈)(≈)	334(22)(≈)(≈)(≈)
2	368(19)	369(22)(≈)	367(21)(≈)(≈)	368(21)(≈)(≈)(≈)
3	391(25)	393(23)(≈)	394(24)(≈)(≈)	392(21)(≈)(≈)(≈)

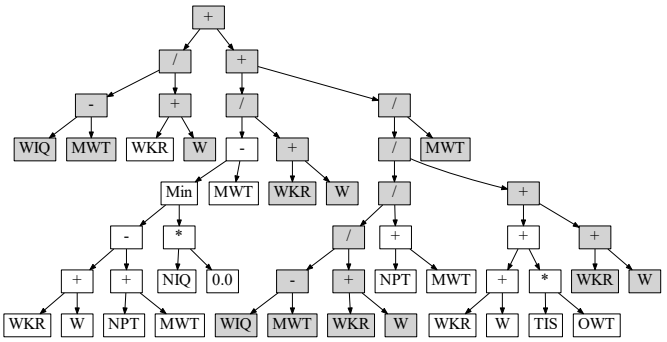


Fig. 9. One of the best evolved sequencing rules for **task 1 <WTmean, 0.75>** in multitask scenario 1.

the algorithms, as they all can learn heuristics that react to the dynamic environment in real time.

VII. FURTHER ANALYSES

A. The Evolved Scheduling Heuristics

The success of multitask learning lies in the knowledge transfer between different tasks. We choose the learned scheduling heuristics for sequencing the operations for different tasks in a multitask scenario as an example to learn more about how the tasks can help with each other. Figs. 9, 10 and 11 show one of the evolved sequencing rules for task 1 <WTmean, 0.75>, task 2 <WFmean, 0.75> and task 3 <WTmax, 0.75> in multitask scenario 1 from one run, respectively. It is obvious that the two sequencing rules for task 1 and task 2 share more knowledge between each other, since the major structures of the rules are the same, which are highlighted in grey. However, the sequencing rule for task 3 is quite different from the sequencing rules for task 1 and task 2, according to the structure of the rule. One possible reason is that the tasks that minimise mean weighted objectives (i.e., WTmean and WFmean) have similar features, while they have different features with the tasks that minimise max weighted objectives (i.e., WTmax). This verifies the relatedness between task 1 and task 2, and the irrelevance of task 1 and task 2 from task 3. This relatedness finding is consistent with our previous observations in Section VI-A and VI-B.

B. Population Diversity Versus Knowledge Transfer

In general, increasing the diversity of the population can improve the performance of an algorithm for handling a task to some extent. Learning from different tasks tends to improve the population diversity of multitask algorithms by getting genetic materials from other subpopulations. If the tasks are not related, is it possible to improve the performance of the algorithm by purely increasing diversity? Does the performance improvement of multitask algorithms mainly benefits from population diversity or transferred knowledge [67]?

TABLE VII
THE MEAN (STANDARD DEVIATION) OF THE OBJECTIVE VALUES ON TEST INSTANCES OF GP AND MTGP OVER 30 INDEPENDENT RUNS IN THREE MULTITASK SCENARIOS WITH TWO **UNRELATED TASKS**.

Scenario	Task	GP	MTGP
1	<WTmean, 0.75>	27.20(0.94)	27.15(0.93)(≈)
	<WTmax, 0.75>	3133.52(865.39)	2762.74(298.65)(↑)
2	<WTmean, 0.85>	75.66(2.09)	76.59(4.11)(≈)
	<WTmax, 0.85>	4097.84(1015.73)	3883.28(1358.91)(↑)
3	<WTmean, 0.95>	298.64(12.56)	293.91(6.35)(≈)
	<WTmax, 0.95>	5818.60(384.19)	5998.40(378.14)(≈)

* WTmean and WTmax are unrelated tasks.

To answer these two questions, we carry out experiments that use multitask algorithm MTGP to solve multitask scenarios with related and unrelated tasks, respectively. Each multitask scenario consists of two tasks. The multitask scenarios with unrelated tasks are designed to optimise WTmean (mean-weighted-tardiness) with job weights 1, 2, 4, and WTmax (max-weighted-tardiness) with job weights 1, 5, 10. The multitask scenarios with related tasks are designed to optimise WTmean (mean-weighted-tardiness) with job weights 1, 2, 4, and WFmean (mean-weighted-flowtime) with job weights 1, 2, 4. If the tasks in the multitask scenarios with related tasks can help each other, and the tasks in the multitask scenarios with unrelated tasks cannot help each other, we can claim that the effectiveness of multitask learning mainly benefits from transferred knowledge. The experiment settings are the same as in Section V.

Table VII shows the mean and standard deviation of the objective values of GP and MTGP for multitask scenarios with two unrelated tasks (i.e., WTmean and WTmax) over 30 independent runs. The results show that for the tasks with objective WTmax, MTGP performs better than GP in two (i.e., <WTmax, 0.75> and <WTmax, 0.85>) out of three scenarios. However, the tasks with objective WTmax are not helpful for the tasks with objective WTmean, which is consistent with our findings in Section VI-A. According to Section VI-B, we know that the tasks with objective WTmax and WTmean are not related. For the tasks with objective WTmean, according to Table VII, they do not get benefit from either transferred knowledge or diversity. This shows that even transferring knowledge from unrelated tasks can help sometimes, but the effectiveness cannot be guaranteed. Table VIII shows the mean and standard deviation of the objective values of GP and MTGP for the multitask scenarios with two related tasks (i.e., WTmean and WFmean) based on 30 independent runs. The results show that the performance of the multitask algorithm MTGP is significantly improved for all tasks in all scenarios.

Taking the results in Table VII and Table VIII into consideration, we can draw the conclusions that learning knowl-

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2022.3199783

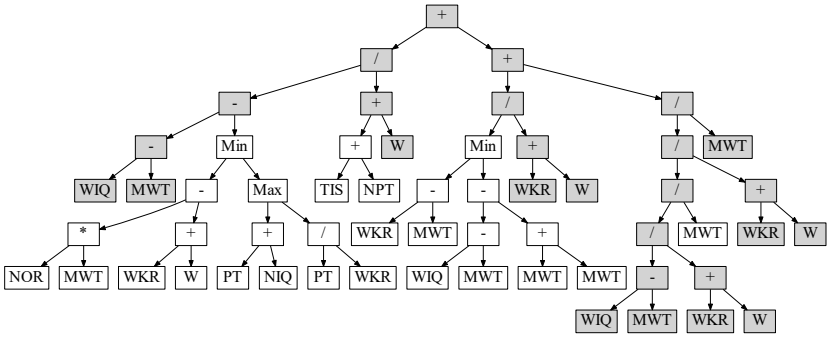


Fig. 10. One of the best evolved sequencing rules for **task 2** <**WTmean, 0.75**> in multitask scenario 2.

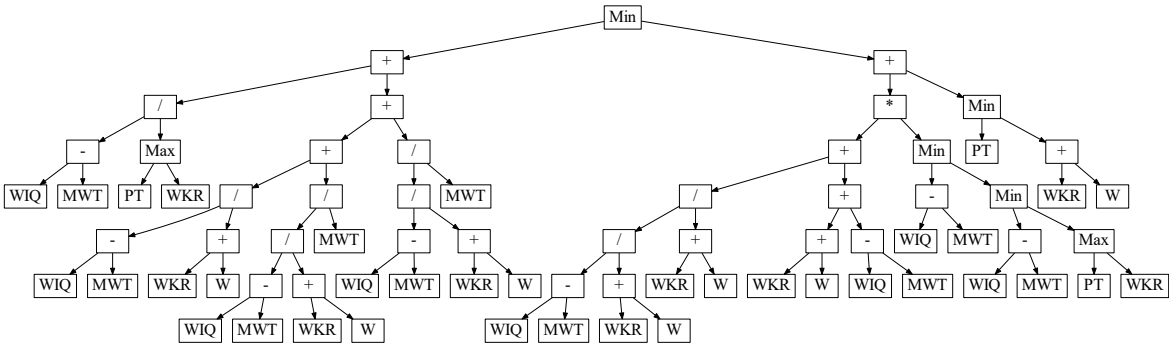


Fig. 11. One of the best evolved sequencing rules for **task 3** <**WTmax, 0.75**> in multitask scenario 2.

TABLE VIII
THE MEAN (STANDARD DEVIATION) OF THE OBJECTIVE VALUES ON TEST INSTANCES OF GP AND MTGP OVER 30 INDEPENDENT RUNS IN THREE MULTITASK SCENARIOS WITH TWO **RELATED TASKS**.

Scenario	Task	GP	MTGP
1	<WTmean, 0.75>	27.32(1.44)	26.78(0.82)(↑)
	<WFmean, 0.75>	736.17(3.62	733.58(1.63)(↑)
2	<WTmean, 0.85>	75.80(2.98)	74.65(0.86)(↑)
	<WFmean, 0.85>	833.42(8.07)	828.47(1.73)(↑)
3	<WTmean, 0.95>	297.27(9.41)	290.91(4.37)(↑)
	<WFmean, 0.95>	1113.27(13.11)	1106.28(6.88)(↑)

* WTmean and WFmean are related tasks.

TABLE IX
THE MEAN (STANDARD DEVIATION) OF THE OBJECTIVE VALUES ON TEST INSTANCES OF ATMTGP AND ATMTGP* OVER 30 INDEPENDENT RUNS IN THREE MULTITASK SCENARIOS.

Scenario	Task	ATMTGP	ATMTGP*
1	<WTmean, 0.75>	26.58(1.06)	27.92(1.56)(↓)
	<WFmean, 0.75>	731.30(3.51)	735.34(6.22)(≈)
	<WTmax, 0.75>	2499.93(139.77)	2513.76(138.12)(≈)
2	<WTmean, 0.85>	75.55(2.41)	76.45(2.31)(≈)
	<WFmean, 0.85>	830.45(4.29)	831.00(4.33)(≈)
	<WTmax, 0.85>	3441.59(264.23)	3524.38(231.24)(≈)
3	<WTmean, 0.95>	295.09(6.62)	297.22(5.98)(≈)
	<WFmean, 0.95>	1106.07(7.63)	1113.56(8.69)(↓)
	<WTmax, 0.95>	5413.16(539.69)	5726.75(534.54)(≈)

edge from other tasks may benefit a task by improving the diversity or sharing the useful knowledge. However, the key to improving the effectiveness of a multitask algorithm is to get useful knowledge from other tasks. This is consistent with the findings in [67]. The diversity effect investigated in this paper cannot guarantee to make a significant improvement for handling the tasks. This further verifies the effectiveness of the proposed adaptive assisted task selection strategy for finding useful tasks to learn knowledge from.

C. Is It Good to Always Choose the Most Related Task?

ATMTGP selects the assisted tasks with a probability proportional to their relatedness. Intuitively, it is beneficial to transfer more knowledge if the task is more related. Therefore, an interesting question is whether it is the best to transfer all the knowledge from the most related task. To answer this question, we modify the assisted task selection scheme in ATMTGP so that it always select the most related task as the assisted task. We denote the ATMTGP with such "winner-take-all" assisted task selection as ATMTGP*.

Table IX shows the mean and standard deviation of the objective values on test instances of ATMTGP and ATMTGP* over 30 independent runs in the three multitask scenarios. The results show that ATMTGP* performs worse than ATMTGP in two out of the nine tasks (i.e., <WTmean, 0.75> and <WFmean, 0.95>). ATMTGP* does not show any superiority compared with ATMTGP on other tasks. This indicates that only learning from the most related task is not the most effective way. One possible reason is that if two tasks are too similar, there might be too limited new genetic materials to learn from each other. This verifies the effectiveness of the proposed ATMTGP, which still gives the less related tasks some probability to be selected as assisted tasks.

This article has been accepted for publication in IEEE Transactions on Evolutionary Computation. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2022.3199783

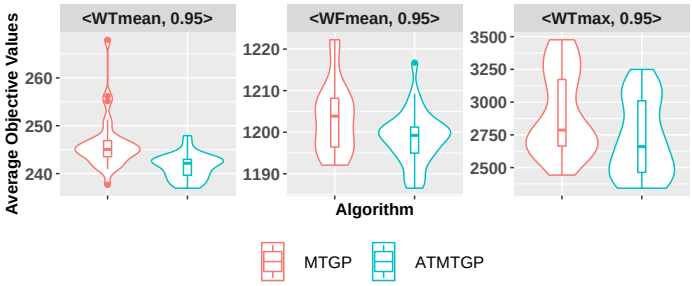


Fig. 12. The violin plot of the average objective values on test instances of MTGP and ATMTGP based on 30 independent runs in one multitask scenario (the processing time of operations follows a uniform discrete distribution with the range [1, 49]).

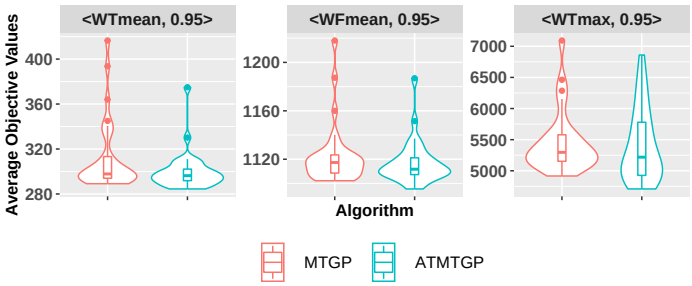


Fig. 13. The violin plot of the average objective values on test instances of MTGP and ATMTGP based on 30 independent runs in one multitask scenario (the machine flexibility equals 50%).

D. The Performance of Algorithms on Other Instances

It is common to use simulation configuration (e.g., utilisation level, objective function, distribution of random variables, machine flexibility) to represent different dynamic scheduling problems. Many existing studies from both computer science and operations research (e.g., [17], [34], [68]–[70]) adopt these dynamic simulation configurations for their studies.

First, we have applied the proposed algorithm to the problem in [34]. Specifically, we change the distribution of processing time for each operation from U[1,99] to U[1, 49] (i.e., used in [34]). Taking the most complex job shop scenario in this paper (i.e., with a utilisation level of 0.95), Fig. 12 shows the violin plots of the average objective values on test instances of MTGP and ATMTGP based on 30 independent runs. Note that we do not include the results of GP and MFGP here, since they are not effective as compared with MTGP and ATMTGP. From Fig. 12, we can see that the proposed algorithm ATMTGP achieves better performance than MTGP, which is consistent with our findings in Section VI-A.

Second, we have applied the proposed algorithm to the problem in [68]. Specifically, we change machine flexibility from 100% to 50% (i.e., used in [68]), i.e., an operation can be processed on half the machines. Fig. 13 shows the violin plots of the average objective values on test instances of MTGP and ATMTGP based on 30 independent runs. Fig. 13 shows that the proposed algorithm ATMTGP achieves better performance than MTGP.

We can see that the proposed algorithm does not bias specific problem instances. We believe the performance of the proposed algorithm has been well verified.

VIII. Conclusions

The goal of this paper was to develop an effective adaptive assisted task selection strategy for choosing a proper task for multitask GP in DFJSS where the relatedness between tasks is unknown. The goal was successfully achieved by proposing an effective strategy for calculating task relatedness based on phenotypic characterisation and Kullback–Leibler divergence, and a properly designed algorithm to select the assisted tasks based on the relatedness information.

The results showed that the developed algorithm ATMTGP can achieve significantly better scheduling heuristics in all the tested multitask scenarios for all the desired tasks. We also found that the boundary of relatedness level of tasks is problem dependent. For example, the relatedness with a value of 0.5 might indicate that a task is not related to another task in one scenario, while it may mean a middle level relatedness relationship between tasks in another scenario. Therefore, for developing assisted task selection strategy, the relative relatedness between tasks is more important than the absolute relatedness value. The effectiveness of the proposed adaptive assisted task selection algorithm ATMTGP was assessed by contrasting not only the effectiveness of evolved scheduling heuristics, but also mainly the relatedness analyses, the structures of the evolved scheduling heuristics for all the tasks in a multitask scenario, and the discussions about diversity and knowledge transfer. It has been discovered that the developed algorithm solves the tasks in a mutually reinforcing manner.

First, the presented algorithm broadens the scope of multitask learning on assessing task relatedness according to individuals with tree-based and variable-length representation. Specifically, it extends the investigations of multitask with other popular but non-vector based evolutionary algorithms. Second, a new assisted task selection strategy has been proposed that the probabilities of tasks to be selected are proportional to the relatedness information. It can provide further guidance for designing the assisted task selection strategy rather than simply choosing the most related one, and can encourage people to dig more in this direction. Last, it is challenging to extract multitask problems when we do not know the characteristics of problems clearly (e.g., like the benchmark problems). The way of extracting multitask scenarios based on optimised objectives in DFJSS can provide guidance for using the multitask to other problems, especially other combinatorial optimisation problems.

In the future, several interesting directions can be investigated further. We plan to extend the proposed algorithm to other dynamic combinatorial optimisation problems such as vehicle routing, cloud scheduling and online packing. We will also investigate other more analytical relatedness measures between tasks based on inherent problem characteristics such as fitness landscape. The current GP methods do not consider the information about future jobs. We will consider the estimated information about future jobs from historical data (e.g., design new terminals) to further enhance the GP training effectiveness. Last but not least, we will develop a more advanced knowledge sharing framework to improve knowledge transferability based on task relatedness.

REFERENCES

[1] R. Caruana, "Multitask learning," *Machine learning*, vol. 28, no. 1, pp. 41–75, 1997.

[2] 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.

[3] 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.

[4] J. Yi, J. Bai, H. He, W. Zhou, and L. Yao, "A multifactorial evolutionary algorithm for multitasking under interval uncertainties," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 908–922, 2020.

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

[6] 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.

[7] K. Chen, B. Xue, Z. Mengjie, 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.

[8] 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.

[9] 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.

[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] A. Gupta, Y. S. Ong, and L. Feng, "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.

[12] S. Ben-David and R. Schuller, "Exploiting task relatedness for multiple task learning," in *Learning Theory and Kernel Machines*. Springer, 2003, pp. 567–580.

[13] Y. Chen, J. Zhong, L. Feng, and J. Zhang, "An adaptive archive-based evolutionary framework for many-task optimization," *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2019, Doi: 10.1109/TETCI.2019.2916051.

[14] Y. S. Ong and A. Gupta, "Evolutionary multitasking: a computer science view of cognitive multitasking," *Cognitive Computation*, vol. 8, no. 2, pp. 125–142, 2016.

[15] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Collaborative multifidelity based surrogate models for genetic programming in dynamic flexible job shop scheduling," *IEEE Transactions on Cybernetics*, 2021. Doi: 10.1109/TCYB.2021.3050141.

[16] H. Engemann, S. Du, S. Kallweit, P. Cönen, and H. Dawar, "OMNIVIL—An autonomous mobile manipulator for flexible production," *Sensors*, vol. 20, no. 24, p. 7249, 2020.

[17] X. Cai and S. Zhou, "Stochastic scheduling on parallel machines subject to random breakdowns to minimize expected costs for earliness and tardy jobs," *Operations Research*, vol. 47, no. 3, pp. 422–437, 1999.

[18] 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.

[19] 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.

[20] K. Jaklinović, M. Durasević, and D. Jakobović, "Designing dispatching rules with genetic programming for the unrelated machines environment with constraints," *Expert Systems with Applications*, p. 114548, 2020.

[21] P. A. Vikhar, "Evolutionary algorithms: A critical review and its future prospects," in *Procedings of the Conference on Global Trends in Signal Processing, Information Computing and Communication*. IEEE, 2016, pp. 261–265.

[22] Z. Wang, J. Zhang, and S. Yang, "An improved particle swarm optimization algorithm for dynamic job shop scheduling problems with random job arrivals," *Swarm and Evolutionary Computation*, vol. 51, p. 100594, 2019.

[23] M. Durasevic and D. Jakobovic, "A survey of dispatching rules for the dynamic unrelated machines environment," *Expert Systems with Applications*, vol. 113, pp. 555–569, 2018.

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

[25] ——, "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.

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

[27] 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.

[28] 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.

[29] 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.

[30] J. Lin, H.-L. Liu, B. Xue, M. Zhang, and F. Gu, "Multi-objective multitasking optimization based on incremental learning," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 5, pp. 824–838, 2020.

[31] 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.

[32] K. K. Bali, Y. S. Ong, A. Gupta, and P. S. Tan, "Multifactorial evolutionary algorithm with online transfer parameter estimation: Mfea-ii," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 69–83, 2019.

[33] K. K. Bali, A. Gupta, Y. S. Ong, and P. S. Tan, "Cognizant multitasking in multiobjective multifactorial evolution: Mo-mfea-ii," *IEEE Transactions on Cybernetics*, 2020, Doi: 10.1109/TCYB.2020.2981733.

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

[35] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Instance rotation based surrogate in genetic programming with brood recombination for dynamic job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2022, Doi: 10.1109/TEVC.2022.3180693.

[36] S. Kullback, *Information theory and statistics*. Courier Corporation, 1997.

[37] 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.

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

[39] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "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.

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

[41] 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.

[42] 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.

[43] ——, "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.

[44] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Learning iterative dispatching rules for job shop scheduling with genetic programming," *The International Journal of Advanced Manufacturing Technology*, vol. 67, no. 1-4, pp. 85–100, 2013.

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

[46] 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.

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

[48] 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.

[49] Z. Huang, Y. Mei, and M. Zhang, "Investigation of linear genetic programming for dynamic job shop scheduling," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, 2021.

[50] 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.

[51] Y. Li, X. Tian, T. Liu, and D. Tao, "On better exploring and exploiting task relationships in multitask learning: Joint model and feature learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1975–1985, 2017.

[52] C. Zhang, D. Tao, T. Hu, and B. Liu, "Generalization bounds of multitask learning from perspective of vector-valued function learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2020. Doi: 10.1109/TNNLS.2020.2995428.

[53] Y. Yuan, Y. S. Ong, L. Feng, A. K. Qin, A. Gupta, B. Da, Q. Zhang, K. C. Tan, Y. Jin, and H. Ishibuchi, "Evolutionary multitasking for multiobjective continuous optimization: Benchmark problems, performance metrics and baseline results," *arXiv preprint arXiv:1706.02766*, 2017.

[54] L. Zhou, L. Feng, J. Zhong, Z. Zhu, B. Da, and Z. Wu, "A study of similarity measure between tasks for multifactorial evolutionary algorithm," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2018, pp. 229–230.

[55] X. Zheng, A. K. Qin, M. Gong, and D. Zhou, "Self-regulated evolutionary multitask optimization," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 16–28, 2019.

[56] K. K. Bali, A. Gupta, L. Feng, Y. S. Ong, and T. P. Siew, "Linearized domain adaptation in evolutionary multitasking," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2017, pp. 1295–1302.

[57] Z. Liang, J. Zhang, L. Feng, and Z. Zhu, "A hybrid of genetic transform and hyper-rectangle search strategies for evolutionary multi-tasking," *Expert Systems with Applications*, vol. 138, p. 112798, 2019.

[58] Z. Chen, Y. Zhou, X. He, and J. Zhang, "Learning task relationships in evolutionary multitasking for multiobjective continuous optimization," *IEEE Transactions on Cybernetics*, 2020, Doi: 10.1109/TCYB.2020.3029176.

[59] 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.

[60] S. J. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach, Third International Edition*. Pearson Education, 2010.

[61] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach," in *Proceedings of the Conference on Genetic and Evolutionary Computation*. ACM, 2010, pp. 257–264.

[62] A. Baykasoğlu, M. Göçken, and L. Özbakir, "Genetic programming based data mining approach to dispatching rule selection in a simulated job shop," *Simulation*, vol. 86, no. 12, pp. 715–728, 2010.

[63] 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.

[64] 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.

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

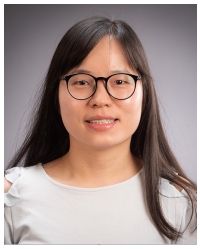
[66] F. Zhang, S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: An evolutionary learning approach," in *Machine Learning: Foundations, Methodologies, and Applications*. Springer, 2021, DOI: 10.1007/978-981-16-4859-5, pp. XXXIII+338 pages.

[67] A. Gupta and Y. S. Ong, "Genetic transfer or population diversification? deciphering the secret ingredients of evolutionary multitask optimization," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, 2016, pp. 1–7.

[68] L. Nie, L. Gao, P. Li, and X. Li, "A gep-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates," *Journal of Intelligent Manufacturing*, vol. 24, no. 4, pp. 763–774, 2013.

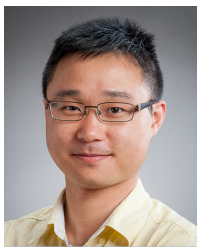
[69] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: a survey with a unified framework," *Complex & Intelligent Systems*, vol. 3, no. 1, pp. 41–66, 2017.

[70] X. N. Shen and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Information Sciences*, vol. 298, pp. 198–224, 2015.



Fangfang Zhang (M'19, IEEE) received the B.Sc. and M.Sc. degrees from Shenzhen University, Shenzhen, China, and the Ph.D. degree in Computer Science from Victoria University of Wellington, New Zealand, in 2014 and 2017, and 2021, respectively.

She is a Research Fellow in 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, hyper-heuristic learning/optimisation, job shop scheduling, surrogate and multitask learning.

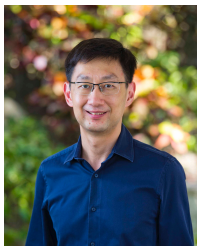


Yi Mei (M'09-SM'18) received the B.Sc. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively.

He is a Senior Lecturer with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. He has more than 150 fully referred publications. His research interests include evolutionary scheduling and combinatorial optimisation, machine learning, genetic programming, and hyper-heuristics.

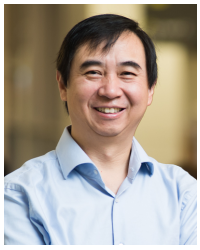
Su Nguyen (M'13) received his Ph.D. degree in Artificial Intelligence and Data Analytics from Victoria University of Wellington, New Zealand, in 2013.

He is a Senior 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 journals and conferences. His expertise includes evolutionary computation, simulation optimisation, automated algorithm design, and their applications in logistics, energy, and transportation.



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. He has published over 300 refereed articles and 8 books. He is an IEEE Fellow, an Honorary Professor at the 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, Victoria University of Wellington, New Zealand. He has published over 800 research papers. He is a Fellow of the Royal Society, and a Fellow of Engineering of New Zealand, a Fellow of IEEE and an IEEE Distinguished Lecturer.