A Unified Framework of Graph-based Evolutionary Multitasking Hyper-heuristic

Xingxing Hao, Rong Qu, *Senior Member, IEEE*, and Jing Liu, *Senior Member, IEEE*

***Abstract*—In recent research, hyper-heuristics have attracted increasing attention among researchers in various fields. The most appealing feature of hyper-heuristics is that they aim to provide more generalized solutions to optimization problems by searching in a high-level space of heuristics instead of direct problem domains. Despite the good generalities of hyper-heuristics, the design of more general search methodologies is still an emerging challenge. Evolutionary multitasking is a relatively new evolutionary paradigm that attempts to solve multiple optimization problems simultaneously. It exploits the underlying similarities among different optimization tasks by allowing the transmission of information among them, thus accelerating the optimization of all tasks. Inherently, hyper-heuristics and evolutionary multitasking share similarities in three ways. (1) They both operate on third-party search spaces. (2) High-level search methodologies are universal. (3) They both conduct cross-domain optimization. To integrate their advantages, i.e., the knowledge-transfer and the cross-domain optimization abilities of the evolutionary multitasking and the search in the heuristic spaces of hyper-heuristics, in this paper, a unified framework of evolutionary multitasking graph-based hyper-heuristic (EMHH) is thereby proposed. To assess the generality and effectiveness of EMHH, the integration of the population-based graph-based hyper-heuristics with the evolutionary multitasking for solving exam timetabling and graph-coloring problems, separately and simultaneously, is studied. The experimental results demonstrate the effectiveness, efficiency, and increased generality of the proposed unified framework compared with single-tasking hyper-heuristics.**

***Index Terms*—Hyper-heuristics, evolutionary multitasking, exam timetabling, graph coloring.**

I. Introduction

META-heuristics have shown to be highly effective in solving various combinational optimization problems [1], [2]. They quite often concern one particular problem, however, tend to perform poorly on other problems or even other instances of the same problem. The performance of these approaches also strongly depends on domain-specific knowledge and expertise such as complicated parameter tunings [3]–[5]. Such tailor-made settings limit the generality of meta-heuristics, making them expensive to develop and adapt to other problems.

This work was supported in part by the General Program of NSFC under Grant 61773300, in part by the Key Program of Fundamental Research Project of Natural Science of Shaanxi Province, China under Grant 2017JZ017, and in part by the Doctoral Students' Short-Term Study Abroad Scholarship Fund of Xidian University.

X. Hao and J. Liu are with the Key Laboratory of Intelligent Perception and Image Understanding of Ministry of Education, Xidian University, Xi'an 710071, China (e-mail: ystar1991@126.com; neouma@163.com).

R. Qu is with the Computational Optimisation and Learning (COL) Lab, School of Computer Science, University of Nottingham, United Kingdom (e-mail: rong.qu@nottingham.ac.uk)

Motivated by this, more recent research concerns generalized and adaptive algorithms [6]. Hyper-heuristics, which are heuristics that choose heuristics, can be regarded as such general algorithms [3], [6]–[10]. Instead of searching directly in the solution space like meta-heuristics, hyper-heuristics work at the higher-level search space of a set of low-level heuristics. The goal is to solve the problem at hand by selecting existing low-level heuristics or generating new low-level heuristics. The only requirement for developing a hyper-heuristic for a problem is a set of low-level heuristics that are easy-to-implement and a problem-specific objective function. These are used at the low-level in hyper-heuristics which are general addressing different problems.

After the term was first proposed in [4], hyper-heuristic approaches have been successfully used to solve a range of combinational optimization problems, such as Boolean satisfiability problems [11], [12], vehicle routing problems [13], [14], packing problems [15], [16], educational timetabling problems [9], and many more [8], [10]. The search space of hyper-heuristics either comprises of existing low-level heuristics or a set of components and operators that are used to construct low-level heuristics. Based on these properties, hyper-heuristics can be categorized into heuristic selection and heuristic generation hyper-heuristics, respectively [17]. Heuristic selection hyper-heuristics select a given set of low-level heuristics to construct or improve solutions; while heuristic generation hyper-heuristics generate new heuristics using a given set of components and operators. Furthermore, in both of the heuristic selection and heuristic generation hyper-heuristics, constructive and perturbative low-level heuristics can be used to build solutions step by step; or modify and improve complete solutions. This paper concerns a new framework of selection constructive hyper-heuristic.

The paradigm of evolutionary multitasking optimization (EMO) was first proposed in [18] for solving multifactorial optimization (MFO) problems, which are categorized as the third category of optimization problems besides single-objective and multi-objective optimization problems. EMO has been successfully extended since then to several domains including continuous optimization, discrete optimization, combinational optimization, and multi-objective optimization [18]–[26]. EMO can optimize two or more tasks simultaneously instead of evaluating every task at each step of evaluation. Under the assumption that each individual is at least skilled at one task, the population in EMO is split into different skill groups. The success of EMO lies in the knowledge transfer among different skill groups; that is, the genetic experience within one group can be transferred to other groups, thus to accelerate the convergence of all tasks [18], [24]. Moreover, the computational cost can be greatly decreased compared with that in solving all tasks separately and sequentially.

Inspired by EMO, a generalized evolutionary multitasking optimization framework was proposed in [24], where the knowledge learned from computationally cheap problems is utilized to assist the optimization of computationally expensive problems via the knowledge transfer mechanism. Li *et al.* extended the evolutionary multitasking framework to a multitasking sparse reconstruction framework for solving the sparse reconstruction problems [27]. In [28], instead of performing knowledge transfer implicitly via genetic operators, a denoising autoencoder was designed to explicitly transfer the solutions among different tasks. To utilize the limited resources more efficiently, Gong *et al.* deigned an online dynamic resource allocation strategy that can allocate computational resources to tasks based on their computational complexities [29]. The literature above extends the original EMO in various aspects, including but not limited to knowledge transfer efficiency, computational efficiency, and its applications. However, they built on the direct solution space of the problems, which is still subject to specific problems.

This paper proposes a unified framework of evolutionary multitasking graph-based hyper-heuristic (EMHH) which inherently integrates hyper-heuristics and EMO based on three synergies between them. Firstly, both of them operate in a third-party search space rather than the search space of direct problem solutions. In EMO, variables of different tasks are mapped into a unified representation, while hyper-heuristics search in a high-level space of heuristics. Secondly, the high-level algorithms in hyper-heuristics are equivalent to the general solvers in EMO. Thirdly, both methodologies concern cross-domain search, however, with different mechanisms. Inspired by these synergies and similarities, EMHH combines their advantages, namely, the knowledge transfer and the cross-domain search of EMO and the high-level search of hyper-heuristics, thus to further enhance the generality of EMHH to a new higher level.

Among the different categories of hyper-heuristics, selection constructive approaches might be intuitively the easiest to comprehend. Moreover, application of graph-based hyper-heuristics to solving educational timetabling problems is one of the most studied topics [10], [30]. Thus, this paper concerns the integration of EMO with the population-based graph-based hyper-heuristics as the precursor of multitasking the hyper-heuristics. The Carter's benchmark [31] is used to assess the effectiveness of the EMHH. In particular, an application on solving exam timetabling problems (ETTPs) and graph coloring problems (GCPs) derived from the Carter's benchmark, in separate and simultaneous manners, is investigated. The experimental results demonstrate that the EMHH provides superior effectiveness, efficiency, and generality compared with single-tasking hyper-heuristics, especially the generality on asynchronous optimization and synchronous cross-domain optimization. More importantly, the results also reveal that solutions of different problems or problem instances in higher-level search spaces may share common structures that could be reused to reduce the computational cost.

The innovation of this paper is that we introduce for the first time the concept of evolutionary multitasking into hyper-heuristics. This leads to the following two contributions: First, a unified framework, EMHH, is developed integrating coherently two methodologies towards a higher level of generality. While the existing hyper-heuristics focus on handling one task at a time, which is still of limited generality, the proposed EMHH is capable of handling multiple tasks, either intra-domain or cross-domain, simultaneously, thus to extend the generality and scope of general algorithms addressing multiple optimization problems. Moreover, the generality of the concept of unification in multitasking is extended, i.e., the unified representation should not be limited to the solution representation space. Other unification schemes may be promising in exploiting the potential of multitasking as well. Second, this paper aims to explore the underlying communalities in the selections of heuristics in solving different problems via evolutionary multitasking with knowledge transfer in the heuristic space.

The rest of this paper is organized as follows. Section II introduces the background of graph-based hyper-heuristics and evolutionary multitasking optimization, followed by the motivations. The EMHH framework is presented in Section III. Section IV presents experimental studies on ETTPs and GCPs and the discussions. The potential applications and future challenges are also provided in this section. Finally, Section V concludes this paper and provides directions for future work.

II. Background

This section introduces the concept of graph heuristics, the definition of selection constructive hyper-heuristic, and the evolutionary multitasking optimization, followed by the motivations of multitasking the hyper-heuristics.

A. Graph Heuristics

Welsh and Powell [32] established the connection between timetabling and scheduling problems with the graph coloring, which subsequently inspired the application of graph heuristics in solving these problems. Graph heuristics order the vertexes in a graph according to the difficulties of coloring them using feasible colors. By representing the events in timetabling problems as vertices and the edges as clashes between the events, the timetabling of events with timeslots is transferred to the problem of assigning vertices with colors. The degree of an event indicates the number of conflicted events, i.e. two events share common students. Graph heuristics can then be used to order the events according to the difficulties of scheduling them into feasible timeslots.

The following five graph heuristics have been widely used in timetabling [9], [10], [33].

- Saturation Degree (SD) indicates the number of feasible timeslots for an event. A smaller SD value indicates less alternative feasible timeslots for an event, thus more difficult to schedule it compared with those with more available timeslots. The smallest SD heuristic selects and schedules events with the smallest SD value first, after which the SD values for the remaining events will be updated.

- Largest Degree (LD) heuristic selects the event with the highest degree, i.e. most conflicts.
- Largest Colored Degree (LCD) heuristic selects and schedules at each step an event with the highest number of conflicts with those scheduled in the timetable. The LCD values of the remaining unscheduled events will be updated afterward.
- Largest Weighted Degree (LWD) indicates the LD weighted by the number of participants (such as students in exam timetabling) involved in the conflicted events. This heuristic selects and schedules the event with the largest weighted degree at each step.
- Largest Enrollment (LE) heuristic selects and schedules the event with the largest number of participants involved.
- Random Ordering (RO) randomly selects an event from the unscheduled set.

The EMHH framework developed in this paper is a selection constructive hyper-heuristic, which is defined as Definition 1 [10]. In this paper, the low-level construction heuristic set H in Definition 1 is comprised of the above graph heuristics. Besides, the EMHH is population-based, i.e. it evolves on a set of low-level heuristics in a population.

Definition 1: Given a problem β and a set of low-level construction heuristics $H = \{h_0, h_1, \ldots, h_n\}$ for the problem domain, a selection constructive hyper-heuristic constructs a complete solution s for β from its initial state s_0 to the final state s by repeatedly selecting and applying a low-level heuristic from H to change from one solution state s_i to the next s_{i+1}.

B. Evolutionary Multitasking Optimization (EMO)

EMO is a new branch of evolutionary algorithms that was first proposed in [18] for addressing multifactorial optimization problems defined as Equation (1).

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_K\} = argmin\{\boldsymbol{f}_1(\boldsymbol{x}_1), \boldsymbol{f}_2(\boldsymbol{x}_2), \ldots, \boldsymbol{f}_K(\boldsymbol{x}_K)\}$$
$$s.t.\ \boldsymbol{x}_k \in \Omega_k, k \in \{1, 2, ..., K\} \quad (1)$$

where K is the number of tasks involved, $\boldsymbol{x}_k = (x_{k,1}, x_{k,2}, \ldots, x_{k,D^k})$ represents a feasible solution for the *k-th* task f_k, D_k denotes its dimensionality, and Ω_k is the search space.

Compared to multi-objective optimization, there are two major distinctive features in MFO. Firstly, there exists no dependence between tasks, i.e. no prior knowledge on the relationship among the K tasks needs to be identified beforehand, while tasks in multi-objective optimization conflict explicitly with each other. Secondly, the variable spaces of K tasks are heterogeneous, namely, each task is evaluated in its own design space.

To compare individuals in the population and clarify relationships between individuals and tasks, the following definitions are proposed in [18] for EMO:

Factorial Rank: The factorial rank r_k^i of individual p_i for the *k-th* task is the index of p_i sorted in ascending order by the function value f_k of the *k-th* task.

Skill Factor: The skill factor τ_i of individual p_i indicates the task that p_i is most skillful in, i.e. $\tau_i = argmin_k\{r_k^i\}$, $k \in \{1, 2, \ldots, K\}$

Scalar Fitness: The scalar fitness ϕ_i of individual p_i is calculated as $\phi_i = 1/min\{r_k^i\}$, $k \in \{1, 2, \ldots, K\}$.

C. Motivations

Although hyper-heuristics have good generalities by searching in heuristics space, the current search paradigms of them still focus on solving isolated problems or isolated problem domains independently. However, as stated in [20], *the real-world problems seldom exist in isolation*, which unveils the fact that the potential of hyper-heuristics might be underestimated. EMO has shown to perform well in solving MFO problems based on the unified representation and the knowledge transfer mechanism [18]–[29]. It also shows good generality in handling multiple optimization tasks simultaneously.

In addition, the direct solutions of different problems usually have no common structure or connection with each other, e.g. the two-dimensional timetable compared to the one-dimensional coloring of a graph. However, the solutions of these two problems in the heuristic space may share similar patterns. For example, in [34], it is found that LWD rather than SD at the early stage of solution construction for ETTPs tends to generate better solutions.

Given above facts, this paper aims to propose an efficient hyper-heuristic framework by building the evolutionary multitasking in the heuristic space. To be specific, on the one hand, this research combines evolutionary multitasking with hyper-heuristics to raise the generality of both hyper-heuristics and evolutionary multitasking to a higher level; i.e., endow hyper-heuristics the ability to handle multitasking problems and extend the concepts of unification scheme in evolutionary multitasking to heuristic space. On the other hand, the proposed framework aims to investigate the underlying similar patterns among different tasks in the heuristic space based on the concept of evolutionary multitasking and knowledge transfer to facilitate effective convergence of optimization.

III. The Evolutionary Multitasking Hyper-heuristic Framework

A. Framework of EMHH

In the unified framework of EMHH shown in Algorithm 1, each individual is associated with one of the K tasks in the population to address K tasks simultaneously. In the initial population on the low-level heuristic space, every individual is evaluated on all tasks at first according to lines 2-7 in Algorithm 1. Then in line 8, the factorial ranks and skill factors are calculated and assigned to all individuals. In the following generations, individual p_i will be only evaluated on the task indicated by its skill factor τ_i. Fitness of all other unevaluated tasks is set to infinite values (i.e. a large enough number). The offspring in the unified representation thus inherit the skill factors from their parents, and the genetic materials of one task can be transferred to address other tasks during the evolution. The workflow of EMHH is presented as Fig.1.

Algorithm 1 EMHH

```
Input:
    $N$: Population size.
    $K$: The number of tasks.
    $H$ : A set of low-level construction heuristics.
Output:
    The best solutions of all tasks.
 1: Initialize population $\boldsymbol{P}$ with randomly generated $N$ low-level heuristic sequences from $H$.
 2: for $i$ = 1 to $N$ do
 3:     for $k$ = 1 to $K$ do
 4:         Construct solution $s_{i,k}$ by heuristic sequence $p_i$ for task $k$.
 5:         Evaluate $s_{i,k}$ for task $k$.
 6:     end for
 7: end for
 8: Initialize skill factor $\tau_i$ of $p_i$, $i \in \{1,2,\ldots,N\}$. // see Section II.B
 9: while termination criteria are not satisfied do
10:     Generate offspring population $\boldsymbol{P}'$ using the cross-task mating operator on $\boldsymbol{P}$. // see Algorithm 2
11:     Evaluate every individual in $\boldsymbol{P}'$ on the task indicated by its skill factor.
12:     $\boldsymbol{R} = \boldsymbol{P} \cup \boldsymbol{P}'$.
13:     Update scalar fitness of every individual in $\boldsymbol{R}$.
14:     Select the $N$ fittest individuals from $\boldsymbol{R}$ according to the scalar fitness as the next generation $\boldsymbol{P}$.
15: end while
16: Output the best solution obtained.
```

In EMHH, a cross-task mating operator is used to generate the offspring as shown in Algorithm 2. This follows the same paradigm as used in [27]. In Algorithm 2, if the selected parents have identical skill factors such as case 1 in Fig.2, the offspring generated by crossover and mutation operators inherent the same skill factors as those of their parents. Otherwise, the transfer of genetic materials between parents depending on a predefined probability *rmp* as shown in lines 5-11 in Algorithm 2 is conducted. As can be seen from Fig.2, the knowledge transfer among tasks occurs in case 3. Since the selected parents have different skill factors, each of their offspring has two alternative skill factors to inherit. This eventually results in four possible combinations of offspring as shown in the dashed rectangle of case 3, where all except the second one have genetic materials transferred. For example, assume o_1 and o_2 are derived from p_a and p_b, respectively, then in the first combination, o_2 inherits p_a's skill factor rather than p_b's, and it will be evaluated on the task indicated by the skill factor of p_a. Consequently, the genetic materials learned in optimizing p_b's task is transferred to p_a, which will likely be helpful in transferring the meaningful building-blocks from one task to the others.

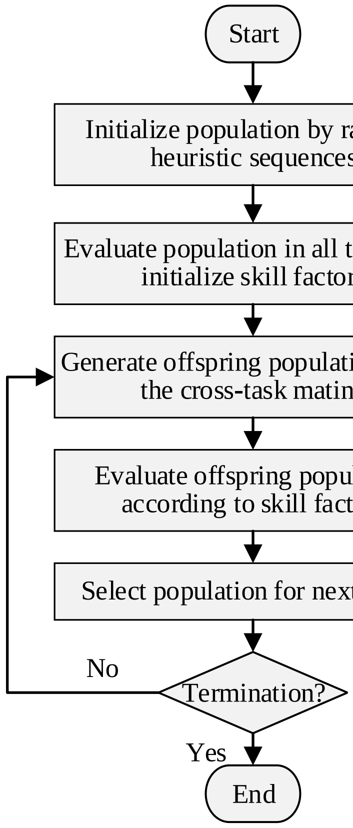

Start
Initialize population by random heuristic sequences
Evaluate population in all tasks and initialize skill factors
Generate offspring population using the cross-task mating
Evaluate offspring population according to skill factors
Select population for next round
No
Termination?
Yes
End

Fig. 1. The workflow of EMHH.

Algorithm 2 Cross-task mating

```
Input:
    $p_a$, $p_b$: Randomly selected parents.
    $rmp$: Predefined random mating probability.
Output:
    offspring $o_1$, $o_2$.
 1: if $\tau_a = \tau_b$ then
 2:     Crossover and mutation are applied sequentially on $p_a$ and $p_b$ to generate $o_1$ and $o_2$.
 3:     $\tau_1 = \tau_2 = \tau_a$(or $\tau_b$).
 4: else
 5:     if $rand < rmp$ then
 6:         Crossover and mutation are applied sequentially on $p_a$ and $p_b$ to generate $o_1$ and $o_2$.
 7:         Randomly assign $\tau_a$ or $\tau_b$ to $\tau_1$ and $\tau_2$.
 8:     else
 9:         Mutate $p_a$ to generate $o_1$, $\tau_1$ = $\tau_a$.
10:         Mutate $p_b$ to generate $o_2$, $\tau_2$ = $\tau_b$.
11:     end if
12: end if
```

B. Solution Representation and Evaluation

The chromosome of the individual is represented as a sequence of heuristics. Recall that the dimensionalities of the direct solutions for the K tasks could be different. Denote $D^{max} = max\{D^k\}$, $k \in \{1,2,\ldots K\}$, to be the length of individual heuristic sequences. Fig.3 shows an example of how the chromosome of two tasks f_1 and f_2 with the

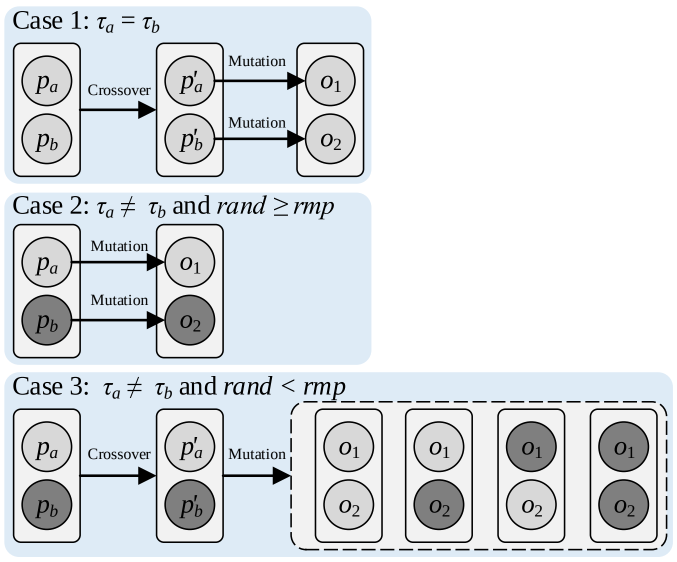

Case 1: $\tau_a = \tau_b$
p_a
p_b
Crossover
p'_a
p'_b
Mutation
Mutation
o_1
o_2
Case 2: $\tau_a \neq \tau_b$ and $rand \geq rmp$
Mutation
Mutation
Case 3: $\tau_a \neq \tau_b$ and $rand < rmp$
Crossover
Mutation

Fig. 2. The process of cross-task mating. The light and dark greyed cycles represent different skill factors.

dimensionality of D_1 and D_2, where $D_1 < D_2$, respectively, are encoded. Each bit in Fig.3 represents a low-level heuristic. If the individual p_i in Fig.3 is associated with task f_1; that is, the skill factor of p_i is $\tau_i = 1$, then, D_1 heuristics selected from the beginning of p_i are employed to construct the solution for f_1, while the remaining $D_2 - D_1$ heuristics are discarded. Otherwise, if $\tau_i = 2$, all heuristics are employed to construct the solution for f_2. The representation for problems with more than two tasks follows the same scheme. In cases where the prior experience is unavailable, the representation for different tasks all starts from the beginning of the chromosome by default [27].

The heuristic sequence p_i itself cannot be evaluated directly without associating it with a specific task or problem; that is, the fitness of a heuristic sequence depends on the quality of the solution constructed by p_i. For example, in the exam timetabling problem, every heuristic in the heuristic sequence is used to schedule one or several exams. The heuristic is scanned one by one in the heuristic sequence p_i. At each construction step, the first unused heuristic h_i in the heuristic sequence p_i is employed to select and then schedule the exams. After a complete solution s is constructed, its soft constraint penalty cost $\phi(s)$ is calculated and is used as the objective value of the heuristic sequence p_i.

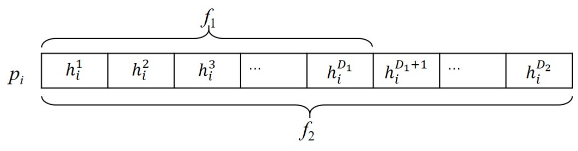


Fig. 3. An example of solution representation.

IV. A Study on Exam Timetabling and Graph Coloring Problems

In this section, the performance of EMHH is evaluated on two to five-task intra and cross-domain MFO problems generated by 16 Carter exam timetabling benchmarks and their corresponding graph coloring problem variants. The effectiveness, efficiency, and generality of EMHH on solving both intra-domain and cross-domain problems are analyzed. Note that a task in EMHH represents an instance of ETTPs or GCPs in the following context unless otherwise stated.

A. Test problems

ETTPs involve assigning a given number of examinations to a set of predefined timeslots subject to certain constraints [7]. The constraints can be classified into two categories, namely, hard and soft constraints. Solutions that satisfy all hard constraints are called feasible solutions. Soft constraints are generally used as criteria in evaluating the quality of feasible solutions. In real cases, both hard and soft constraints vary from institutions to institutions. The most common constraints in ETTPs are listed as follows:

(1) Student conflict: students cannot have more than one exam in one timeslot (hard constraint).
(2) Room capacity: the total number of students assigned cannot exceed the room capacity in one timeslot or session (hard constraint).
(3) Exam distribution: exams of one student should be as spread as possible in the timetable (soft constraint).

Three variants of ETTPs, namely, the graph coloring problem, uncapacitated exam timetabling problems (uETTPs), and capacitated exam timetabling problems (cETTPs), have been studied in the existing literature.

- GCPs aim to find the smallest number of colors for all vertices without conflicts; that is, adjacent vertices are assigned different colors. An ETTP degrades to a graph coloring problem if only constraint (1) is taken into considerations [7], [35].
- In uETTPs, the room capacity constraint is relaxed, and the objective is to minimize the violation of constraint (3). A penalty $w_t = 2^{5-t}, t \in [1, 5]$ occurs if two exams of a student are assigned t timeslots apart [7]. The objective value is calculated as $\phi = Total\_penalty/Number\_of\_students$ where the *total_penalty* is the summation of penalty caused by all students. This objective represents a preference to timetables where each student's exams distribute as sparse as possible.
- In contrast to uETTPs, in cETTPs, restrictions of room capacities apply per timeslot [36], [37] or per session [35]. The objective of cETTPs is to minimize the number of students sitting two exams consecutively during one day [36], and it can also be extended to overnight cases [35], [37].

In this paper, we employ version I, II and IIc (the corrected version II) Carter benchmarks summarized in [7] as the test sets of uETTPs and GCPs. A brief summary of the properties of these benchmarks can be found at http://www.cs.nott.ac.uk/~pszrq/data.htm and Table SI in the Supplementary Material. A modified penalty w'_t of uETTPs as shown in Equation (2) is adapted, where a large penalty occurs for each pair of conflicting exams. The resulting search space of the proposed algorithm thus contains infeasible solutions; however, the fitness landscape becomes connected. The greedy strategy in [5] and [34] is used to choose timeslots for the selected exams.

$$w'_t = \begin{cases} 2^{5-t}, & t \in [1,5] \\ 10000, & t = 0 \end{cases} \tag{2}$$

For the GCPs, the direct use of the number of colors in the objective function is very likely to form a fitness landscape with large plateaus, thus, it is difficult to distinguish different solutions with the same number of colors. In this paper, we modified the evaluation function in [5] to include two measures on the coloring, namely, the coloring sum as used in [5] and the cube of the number of colors used. Coloring sum is calculated as the sum of the product of the size and color index of each color class that comprises of a set of vertices with the same color. Thus, this new evaluation function not only considers the number of colors but also the size of color classes.

B. Genetic Operators

In [34] and [38], it is shown that the heuristics at the later stage of solution construction tend to make less difference in the quality of solutions. To dynamically allocate the computational resource in EMHH, an adaptive one-point mutation operator shown in Algorithm 3 is used to firstly modify early parts of heuristic sequences and then extend to later parts along with the evolution. $|p|$ represents the length of individual p. As a result, the earlier heuristics in the heuristic sequences have more chances to evolve towards proper hybridizations. The uniform crossover [39] is used in EMHH.

Algorithm 3 Adaptive one-point mutation

```
Input:
    $p$: Parent heuristic sequence.
    $gen$: Current generation.
    $G$: Total number of generations.
Output:
    $o$: Offspring.
 1: $mLength = max(1, |p| \cdot gen/G)$. // Decide the mutation
    part in $p$, i.e. $p(1) \sim p(mLength)$.
 2: $pm = 1/mLength$. // Probability of mutation.
 3: $o = p$.
 4: for each position $i = 1$ to $mLength$ do
 5:     if $rand < pm$ then
 6:         //$rand \in (0, 1)$ is a random real number.
 7:         randomly change $o(i)$ to another heuristic.
 8:     end if
 9: end for
10: return $o$.
```

C. Experimental Setup

In this paper, only Saturation Degree (SD) and Largest Weighted Degree (LWD) are employed as the candidate heuristics in EMHH. According to [31], [33], [40], SD outperforms other graph heuristics when being used alone, and showed to be effective in [34] when hybridized with LWD. Each of the graph heuristics will schedule two events as in [33] to construct the solutions.

In EMHH, the population size is set to 30 and the total number of generations is set to 100. The single-tasking optimization form, named SOHH, is developed with every setting identical with EMHH, except that only one task is optimized.

The predefined probability rmp controls how often the evolved knowledge is transferred among different tasks in EMHH. As suggested in [25], rmp should be close to 1 if prior knowledge that handling tasks are correlated exists; otherwise, a small rmp value is proper. In the literature, there exists no effective measurement between uETTP instances due to the difficulty in assessing their similarity. In a set of preliminary experiments, the range of rmp is set to [0.1, 1] with an increasing step of 0.1 to examine the impact of rmp on the performance of the EMHH for solving three MFO problems (hec92, hec92 II), (hec92, sta83), and (hec92, tre92). The average results from 30 independent runs as plotted in Fig. 4 show that EMHH achieved relatively good performance on both tasks in (hec92, hec92 II) and (hec92, tre92) with rmp = 0.8, and the performance on (hec92, sta83) is less sensitive to the settings of rmp. We therefore set rmp to 0.8 in the remaining experiments.

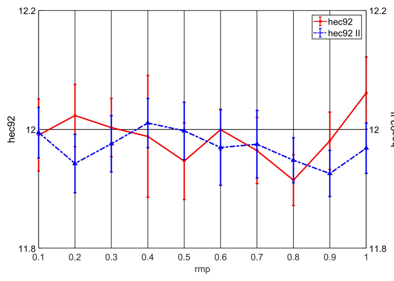
All algorithms were implemented in C++ using Visual Studio 2017. Experiments were conducted on a desktop with Intel Core i7-3820 CPU (3.60GHz) 16.0 GB memory and 64-bit Windows 10. Average results are obtained over 30 independent runs of the algorithms.

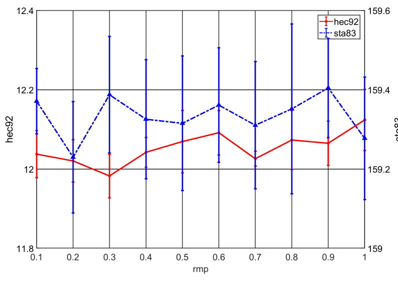
D. Experimental Results

An additional five sets of experiments have been conducted to evaluate the EMHH framework. The first three experiments demonstrate the effectiveness, efficiency and generality of asynchronous optimization of EMHH for intra-domain problems. The fourth set of experiments presents the generality of EMHH in addressing cross-domain optimization problems. The generality of EMHH is further examined on problems with over two tasks in the fifth set of experiment. Finally, comparisons between EMHH and other existing hyper-heuristics are presented.

1) Comparison between EMHH and SOHH on uETTPs:

At first, we set K to 2, thus in total 120 MFO problems are created from the 16 Carter exam timetabling benchmarks. The average, best fitness values, and standard deviations obtained by EMHH and SOHH are presented in Table I, where F1 and F2 represent the objective value of the two tasks (f_1, f_2) in an MFO problem, respectively.For example, in the first MFO problem (car91, car92), F1 and F2 denotes the objective value for task f_1 (car91) and f_2 (car92), respectively. Unless otherwise stated, Fx represents the objective value of the task f_x, $x = 1, ..., K$, in the following context. The Wilcoxon rank-sum test with 95% confidence level is conducted between results of EMHH and SOHH. In Table I, the significantly better results are highlighted in grey, and better results are in light


(a) (hec92,hec92 II)


(b) (hec92,sta83)

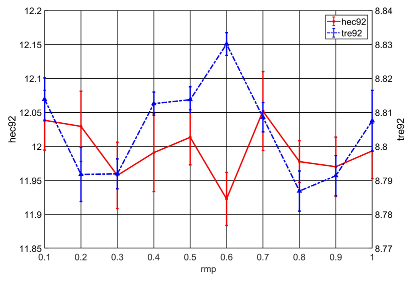

(c) (hec92,tre92)

Fig. 4. The impact of different probabilities rmp in EMHH.

grey. The detailed p and h values are provided in Table SIII in the Supplementary Material, where $h = 1$ or $p < 0.05$ denotes a significantly better performance at the confidence level of 95%. Table II compares the results of EMHH and SOHH listed in Table I using three indicators defined as follows:

- All-win: if the results on both tasks, i.e. F1 and F2 in an MFO problem, obtained by approach A are all significantly better than approach B, the count of all-wins for A is increased by one.
- One-win: if approach A achieves a significantly better result on one of the two tasks, and better, worse or equal results on another task than B, then the count of one-wins for A is increased by one.
- Tie: otherwise, increase the tie count by one.

The comparison results show that EMHH is highly effective against SOHH on uETTPs. It outperformed SOHH on 43 MFO problems, achieved significantly better results for one of the two tasks than SOHH on 63 MFO problems, and performed similarly on 13 MFO problems. SOHH only won on one MFO problem in terms of the one-win indicator. Moreover, the results demonstrate that multitasking works effectively in the low-level heuristic space. In summary, EMHH is shown to be an effective hyper-heuristic framework in solving uETTPs.

2) Comparisons between EMHH and SOHH on Computational Costs: Compared with SOHH under the same parameter settings, the number of evaluations in EMHH can be reduced to almost $1/K$ of that in SOHH. In EMHH, each individual is only evaluated on one of the K tasks indicated by its skill factor, which eventually leads to the reduction of computational costs. As the baseline, Table III presents the average time used by SOHH for solving the 16 uETTP instances. To demonstrate the efficiency of EMHH, the average computational costs for solving all MFO problems comprising of the instance lse91 and other 15 instances are presented in Fig.5. For comparison, SOHH is used to solve both single tasks sequentially in these MFO problems (i.e. the average time consumed by the first and second tasks is denoted as SO-F1 and SO-F2 in Fig.5, respectively). Instance lse91 consumes median average time according to Table III; thus, the performance of EMHH on multitasking lse91 with other instances consuming longer and shorter time can be clearly presented.

In most of the cases shown in Fig.5, the time consumed by

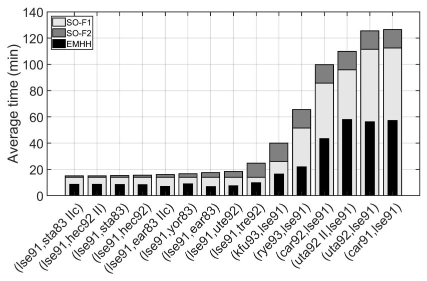


Fig. 5. Time comparison between SOHH and EMHH for MFO problems.

EMHH is around half of that by SOHH. Moreover, EMHH significantly reduces the time consumed by the first task, namely, instance lse91, which represents the expensive computational task in an MFO. Note that the orders of tasks in an MFO do not affect the performance of EMHH, here we subject the expensive computational task to f_1 for demonstration purposes. We can conclude that EMHH works more efficiently in the low-level heuristic space than SOHH in solving uETTPs. The same conclusion can be drawn from the time comparisons of other instances shown in Fig.S1 of the Supplementary Material.

3) Generality on Asynchronous Optimization: Results in Table II indicate that in the low-level heuristic space, the solutions for different instances may share some commonly evolved knowledge. Although the direct solutions of timetables for different instances could be highly distinctive, the low-level heuristic sequences used to construct them may possess similar structures, which subsequently infers that the heuristic sequences obtained in solving one task could be reused for solving other tasks. To verify this, another set of experiments is conducted, where at first a single task is optimized by SOHH, then after a number of iterations the second task is inserted into the evolution, thus the paradigm turns into EMHH for the remaining generations. Based on the average time shown in Table III, the instance hec92 consumes relatively less time compared with other instance, and therefore is processed first as task f_1 to save computational resources. All parameters are kept unchanged and the second task is serted after half of the total generations.

As can be seen from Table IV, compared with SOHH, EMHH achieved significantly better results on f_2 in ten, better results

TABLE I
THE MINIMUM, MEAN AND STANDARD DEVIATIONS OBTAINED BY EMHH AND SOHH ON THE uETTPs.

MFOs	EMHH		SOHH		MFOs	EMHH		SOHH		MFOs	EMHH		SOHH		MFOs	EMHH		SOHH	
	F1	F2	F1	F2		F1	F2	F1	F2		F1	F2	F1	F2		F1	F2	F1	F2
car91 car92	5.14	4.27	5.18	4.31	ear83 hec92	35.88	11.68	36.62	11.69	hec92 tre92	11.65	8.55	11.69	8.78	lse91 ute92	11.39	27.31	11.65	28.31
	5.22	4.39	5.26	4.41		36.75	11.99	37.12	12.12		11.98	8.79	12.12	8.88		11.68	28.31	11.85	28.75
	0.04	0.04	0.04	0.04		0.54	0.16	0.25	0.27		0.18	0.08	0.27	0.06		0.16	0.31	0.16	0.14
car91 ear83	5.15	36.07	5.18	36.62	ear83 hec92 II	35.79	11.63	36.62	11.67	hec92 uta92	11.5	3.38	11.69	3.39	lse91 yor83	11.18	39.65	11.65	41.05
	5.22	36.98	5.26	37.12		36.64	12.01	37.12	12.1		11.93	3.43	12.12	3.43		11.68	41.8	11.85	42.87
	0.03	0.51	0.04	0.25		0.4	0.2	0.25	0.29		0.28	0.02	0.27	0.02		0.19	0.71	0.16	2.91
car91 ear83 IIc	5.15	39.21	5.18	39.51	ear83 kfu93	35.9	15.09	36.62	15.37	hec92 uta92 II	11.63	3.37	11.69	3.35	rye93 sta83	9.32	158.83	9.34	169.77
	5.23	40.74	5.26	41.13		36.75	15.57	37.12	15.68		11.99	3.41	12.12	3.41		9.57	159.36	9.55	169.86
	0.03	0.93	0.04	1		0.5	0.16	0.25	0.15		0.23	0.02	0.27	0.03		0.09	0.38	0.08	0.09
car91 hec92	5.19	11.66	5.18	11.69	ear83 lse91	35.49	11.07	36.62	11.65	hec92 ute92	11.55	27.9	11.69	28.31	rye93 sta83 IIc	9.44	34.25	9.34	34.26
	5.22	11.95	5.26	12.12		36.84	11.7	37.12	11.85		12.02	28.4	12.12	28.75		9.6	34.65	9.55	34.99
	0.03	0.15	0.04	0.267		0.57	0.18	0.25	0.16		0.26	0.24	0.27	0.14		0.08	0.18	0.08	0.22
car91 hec92 II	5.11	11.62	5.18	11.67	ear83 rye93	35.68	9.43	36.62	9.34	hec92 yor83	11.6	40.21	11.69	41.05	rye93 tre92	9.3	8.61	9.34	8.78
	5.22	11.97	5.26	12.1		36.8	9.57	37.12	9.55		12.01	41.74	12.12	42.87		9.59	8.78	9.55	8.88
	0.04	0.19	0.04	0.29		0.51	0.08	0.25	0.08		0.22	0.55	0.27	2.91		0.09	0.08	0.08	0.06
car91 kfu93	5.16	15.07	5.18	15.37	ear83 sta83	36.01	158.56	36.62	169.77	hec92 II kfu93	11.58	15.22	11.67	15.37	rye93 uta92	9.43	3.4	9.34	3.39
	5.23	15.52	5.26	15.68		36.91	159.5	37.12	169.86		12.02	15.6	12.1	15.68		9.57	3.43	9.55	3.43
	0.03	0.21	0.04	0.15		0.54	0.49	0.25	0.09		0.21	0.16	0.29	0.15		0.08	0.01	0.08	0.02
car91 lse91	5.14	11.23	5.18	11.65	ear83 sta83 IIc	35.9	34.06	36.62	34.26	hec92 II lse91	11.62	11.18	11.67	11.65	rye93 uta92 II	9.45	3.37	9.34	3.35
	5.23	11.69	5.26	11.85		36.84	34.64	37.12	34.99		11.98	11.57	12.1	11.85		9.6	3.41	9.55	3.41
	0.03	0.23	0.04	0.16		0.44	0.23	0.25	0.22		0.21	0.2	0.29	0.16		0.08	0.02	0.08	0.03
car91 rye93	5.18	9.4	5.18	9.34	ear83 tre92	35.31	8.62	36.62	8.78	hec92 II rye93	11.5	9.35	11.67	9.34	rye93 ute92	9.37	27.8	9.34	28.31
	5.23	9.57	5.26	9.55		36.78	8.81	37.12	8.88		11.99	9.59	12.1	9.55		9.59	28.42	9.55	28.75
	0.03	0.08	0.04	0.08		0.48	0.08	0.25	0.06		0.23	0.09	0.29	0.08		0.09	0.24	0.08	0.14
car91 sta83	5.19	158.66	5.18	169.77	ear83 uta92	35.64	3.38	36.62	3.39	hec92 II sta83	11.63	158.57	11.67	169.77	rye93 yor83	9.44	41.03	9.34	41.05
	5.23	159.31	5.26	169.86		36.81	3.42	37.12	3.43		11.95	159.33	12.1	169.86		9.58	42.33	9.55	42.87
	0.03	0.48	0.04	0.09		0.38	0.02	0.25	0.02		0.19	0.42	0.29	0.09		0.07	2.04	0.08	2.91
car91 sta83 IIc	5.16	34.24	5.18	34.26	ear83 uta92 II	35.71	3.38	36.62	3.35	hec92 II sta83 IIc	11.63	33.81	11.67	34.26	sta83 sta83 IIc	158.68	34.37	169.77	34.26
	5.23	34.65	5.26	34.99		36.79	3.41	37.12	3.41		12.06	34.61	12.1	34.99		159.3	34.74	169.86	34.99
	0.03	0.19	0.04	0.22		0.52	0.02	0.25	0.03		0.23	0.26	0.29	0.22		0.29	0.17	0.09	0.22
car91 tre92	5.17	8.61	5.18	8.78	ear83 ute92	36.19	27.23	36.62	28.31	hec92 II tre92	11.53	8.62	11.67	8.78	sta83 tre92	158.5	8.62	169.77	8.78
	5.24	8.8	5.26	8.88		36.77	28.31	37.12	28.75		11.98	8.83	12.1	8.88		159.24	8.8	169.86	8.88
	0.03	0.09	0.04	0.06		0.33	0.35	0.25	0.14		0.17	0.07	0.29	0.06		0.39	0.06	0.09	0.06
car91 uta92	5.12	3.39	5.18	3.39	ear83 yor83	35.82	40.65	36.62	41.05	hec92 II uta92	11.38	3.39	11.67	3.39	sta83 uta92	158.59	3.4	169.77	3.39
	5.22	3.42	5.26	3.43		36.65	41.78	37.12	42.87		11.99	3.43	12.1	3.43		159.36	3.43	169.86	3.43
	0.04	0.02	0.04	0.02		0.49	0.62	0.25	2.91		0.24	0.02	0.29	0.02		0.34	0.02	0.09	0.02
car91 uta92 II	5.17	3.38	5.18	3.35	ear83 IIc hec92	39.58	11.56	39.51	11.69	hec92 II uta92 II	11.6	3.38	11.67	3.35	sta83 uta92 II	158.77	3.37	169.77	3.35
	5.23	3.41	5.26	3.41		40.71	11.96	41.13	12.12		11.95	3.42	12.1	3.41		159.26	3.42	169.86	3.41
	0.03	0.02	0.04	0.03		0.78	0.21	1	0.27		0.2	0.02	0.29	0.03		0.4	0.02	0.09	0.03
car91 ute92	5.16	27.66	5.18	28.31	ear83 IIc hec92 II	39.41	11.63	39.51	11.67	hec92 II ute92	11.48	27.42	11.67	28.31	sta83 ute92	158.6	27.26	169.77	28.31
	5.22	28.25	5.26	28.75		40.47	11.99	41.13	12.1		11.99	28.32	12.1	28.75		159.17	28.15	169.86	28.75
	0.04	0.3	0.04	0.14		0.77	0.22	1	0.29		0.21	0.34	0.29	0.14		0.38	0.35	0.09	0.14
car91 yor83	5.18	41.06	5.18	41.05	ear83 IIc kfu93	39.41	14.99	39.51	15.37	hec92 II yor83	11.58	40.52	11.67	41.05	sta83 yor83	158.73	40.71	169.77	41.05
	5.23	41.98	5.26	42.87		40.42	15.58	41.13	15.68		11.96	41.85	12.1	42.87		159.42	41.93	169.86	42.87
	0.03	0.43	0.04	2.91		0.67	0.18	1	0.15		0.16	0.5	0.29	2.91		0.41	0.65	0.09	2.91
car92 ear83	4.32	35.5	4.31	36.62	ear83 IIc lse91	39.36	11.4	39.51	11.65	kfu93 lse91	14.79	11.24	15.37	11.65	sta83 IIc tre92	34.05	8.66	34.26	8.78
	4.4	36.62	4.41	37.12		40.87	11.68	41.13	11.85		15.54	11.62	15.68	11.85		34.64	8.79	34.99	8.88
	0.03	0.48	0.04	0.25		2.65	0.15	1	0.16		0.24	0.17	0.15	0.16		0.23	0.08	0.22	0.06
car92 ear83 IIc	4.3	38.99	5.18	39.51	ear83 IIc rye93	39.37	9.31	39.51	9.34	kfu93 rye93	15.1	9.44	15.37	9.34	sta83 IIc uta92	33.74	3.4	34.26	3.39
	4.38	40.74	5.26	41.13		40.65	9.56	41.13	9.55		15.58	9.59	15.68	9.55		34.65	3.43	34.99	3.43
	0.04	1.25	0.04	1		1.17	0.09	1	0.08		0.19	0.07	0.15	0.08		0.25	0.02	0.22	0.02
car92 hec92	4.33	11.57	4.31	11.69	ear83 IIc sta83	39.51	158.71	39.51	169.77	kfu93 sta83	15.23	158.53	15.37	169.77	sta83 IIc uta92 II	34.28	3.38	34.26	3.35
	4.39	11.94	4.41	12.12		41.31	159.41	41.13	169.86		15.57	159.38	15.68	169.86		34.66	3.42	34.99	3.41
	0.03	0.22	0.04	0.27		2.44	0.38	1	0.09		0.18	0.32	0.15	0.09		0.19	0.01	0.22	0.03
car92 hec92 II	4.35	11.63	4.31	11.67	ear83 IIc sta83 IIc	39.25	34.43	39.51	34.26	kfu93 sta83 IIc	15.31	34.2	15.37	34.26	sta83 IIc ute92	34.08	27.32	34.26	28.31
	4.39	11.93	4.41	12.1		40.16	34.7	41.13	34.99		15.58	34.63	15.68	34.99		34.64	28.2	34.99	28.75
	0.03	0.2	0.04	0.29		0.61	0.17	1	0.22		0.16	0.2	0.15	0.22		0.27	0.35	0.22	0.14
car92 kfu93	4.31	15.2	4.31	15.37	ear83 IIc tre92	39.29	8.69	39.51	8.78	kfu93 tre92	15.37	8.6	15.37	8.78	sta83 IIc yor83	34.21	40.72	34.26	41.05
	4.39	15.6	4.41	15.68		40.78	8.8	41.13	8.88		15.6	8.81	15.68	8.88		34.69	41.95	34.99	42.87
	0.04	0.16	0.04	0.15		1.04	0.06	1	0.06		0.14	0.1	0.15	0.06		0.22	0.57	0.22	2.91
car92 lse91	4.34	11.3	4.31	11.65	ear83 IIc uta92	38.94	3.39	39.51	3.39	kfu93 uta92	15.13	3.39	15.37	3.39	tre92 uta92	8.6	3.38	8.78	3.39
	4.39	11.62	4.41	11.85		40.22	3.43	41.13	3.43		15.59	3.42	15.68	3.43		8.8	3.43	8.88	3.43
	0.03	0.13	0.04	0.16		0.71	0.02	1	0.02		0.17	0.02	0.15	0.02		0.07	0.02	0.06	0.02
car92 rye93	4.35	9.43	4.31	9.34	ear83 IIc uta92 II	39.52	3.36	39.51	3.35	kfu93 uta92 II	15.26	3.37	15.37	3.35	tre92 uta92 II	8.58	3.39	8.78	3.35
	4.39	9.61	4.41	9.55		40.4	3.41	41.13	3.41		15.59	3.41	15.68	3.41		8.81	3.41	8.88	3.41
	0.03	0.08	0.04	0.08		0.68	0.02	1	0.03		0.14	0.02	0.15	0.03		0.1	0.02	0.06	0.03
car92 sta83	4.35	158.56	4.31	169.77	ear83 IIc ute92	39.03	27.64	39.51	28.31	kfu93 ute92	15.23	27.74	15.37	28.31	tre92 ute92	8.66	27.23	8.78	28.31
	4.4	159.33	4.41	169.86		40.47	28.37	41.13	28.75		15.6	28.41	15.68	28.75		8.8	28.24	8.88	28.75
	0.03	0.35	0.04	0.09		0.71	0.3	1	0.14		0.15	0.31	0.15	0.14		0.09	0.36	0.06	0.14
car92 sta83 IIc	4.32	34.3	4.31	34.26	ear83 IIc yor83	39.39	40.16	39.51	41.05	kfu93 yor83	15.08	41.11	15.37	41.05	tre92 yor83	8.67	40.66	8.78	41.05
	4.38	34.65	4.41	34.99		40.46	41.78	41.13	42.87		15.57	41.85	15.68	42.87		8.8	41.69	8.88	42.87
	0.03	0.17	0.04	0.22		0.59	0.6	1	2.91		0.19	0.45	0.15	2.91		0.07	0.48	0.06	2.91
car92 tre92	4.29	8.63	4.31	8.78	hec92 hec92 II	11.5	11.51	11.69	11.67	lse91 rye93	11.3	9.44	11.65	9.34	uta92 uta92 II	3.38	3.37	3.39	3.35
	4.39	8.8	4.41	8.88		11.91	11.95	12.12	12.1		11.63	9.57	11.85	9.55		3.43	3.41	3.43	3.41
	0.04	0.08	0.04	0.06		0.21	0.19	0.27	0.29		0.2	0.08	0.16	0.08		0.02	0.02	0.02	0.03
car92 uta92	4.32	3.39	4.31	3.39	hec92 kfu93	11.69	15.05	11.69	15.37	lse91 sta83	11.2	158.52	11.65	169.77	uta92 ute92	3.39	27.88	3.39	28.31
	4.38	3.43	4.41	3.43		11.99	15.54	12.12	15.68		11.56	159.29	11.85	169.86		3.43	28.26	3.43	28.75
	0.03	0.02	0.04	0.02		0.2	0.17	0.27	0.15		0.17	0.39	0.16	0.09		0.02	0.24	0.02	0.14
car92 uta92 II	4.32	3.37	4.31	3.35	hec92 lse91	11.68	11.29	11.69	11.65	lse91 sta83 IIc	11.21	34.14	11.65	34.26	uta92 yor83	3.39	39.59	3.39	41.05
	4.38	3.41	4.41	3.41		12.05	11.61	12.12	11.85		11.65	34.68	11.85	34.99		3.43	42.13	3.43	42.87
	0.04	0.02	0.04	0.03		0.22	0.16	0.27	0.16		0.18	0.18	0.16	0.22		0.02	2.22	0.02	2.91
car92 ute92	4.33	27.69	4.31	28.31	hec92 rye93	11.59	9.36	11.69	9.34	lse91 tre92	11.19	8.64	11.65	8.78	uta92 II ute92	3.37	27.5	3.35	28.31
	4.39	28.29	4.41	28.75		11.99	9.59	12.12	9.55		11.06	8.08	11.85	8.88		3.42	28.28	3.41	28.75
	0.04	0.34	0.04	0.14		0.22	0.1	0.27	0.08		0.16	0.07	0.16	0.06		0.01	0.31	0.03	0.14
car92 yor83	4.33	40.99	4.31	41.05	hec92 sta83	11.49	158.61	11.69	169.77	lse91 uta92	11.33	3.38	11.65	3.39	uta92 II yor83	3.37	40.62	3.35	41.05
	4.39	41.86	4.41	42.87		12.07	159.35	12.12	169.86		11.66	3.43	11.85	3.43		3.41	41.7	3.41	42.87
	0.03	0.43	0.04	2.91		0.27	0.46	0.27	0.09		0.2	0.03	0.16	0.03		0.02	0.41	0.03	2.91
ear83 ear83 IIc	35.44	39.38	36.62	39.51	hec92 sta83 IIc	11.45	34.28	11.69	34.26	lse91 uta92 II	11.31	3.34	11.65	3.35	ute92 yor83	27.2	40.45	28.31	41.05
	36.58	40.11	37.12	41.13		12.05	34.72	12.12	34.99		11.67	3.41	11.85	3.41		28.27	41.95	28.75	42.87
	0.56	0.62	0.25	1		0.21	0.2	0.27	0.22		0.2	0.03	0.16	0.03		0.36	0.7	0.14	2.91

TABLE II
STATISTIC COMPARISONS BETWEEN EMHH AND SOHH ON RESULTS IN TABLE I.

All-win		One-win		Tie
EMHH	SOHH	EMHH	SOHH	13
43	0	63	1	

in three, and equal result in one test case, respectively; while SOHH obtained a better result in only one test case. Although the second task is introduced halfway in the evolution, EMHH still outperformed SOHH. This demonstrates that the low-level heuristic sequences learned from optimizing hec92 are applicable to other instances, i.e. the solutions of different uETTP instances in the low-level heuristic space may share some common structures or knowledge. Moreover, EMHH is capable of retaining such knowledge in the low-level heuristic space via the knowledge transfer scheme, showing a higher level of generality on asynchronous optimization.

4) Generality on Synchronous Cross-domain Optimization: To examine the generality of EMHH on synchronous cross-domain optimization, 16 MFO problems are created, with each consisting of a uETTP task and its graph coloring problem variant. All parameters used in this subsection are consistent with the above experiments. The comparison results between EMHH and SOHH on solving these cross-domain MFO problems and their single-objective optimization variants, respectively, are presented in Table V.

As can be seen from Table V, for nine instances, significantly better results on uETTPs are obtained by solving the MFO problems by EMHH than solving tasks independently by SOHH. For the rest of the instances, EMHH achieved better results in five of them and equal results with the SOHH in the two other instances. For all the graph coloring variants, both approaches obtained the same best results except for instance yor83, where the performance of EMHH is slightly more robust than that of SOHH. To conclude, the results in Table V demonstrate a high generality of EMHH on synchronous cross-domain optimization. Moreover, in the low-level heuristic space, like the graph heuristic space here, the solutions for different problems, such as uETTPs and GCPs, may share some common structures evolved by solving the companion tasks in MFO. Therefore, the knowledge in the solutions of heuristic space obtained from optimizing in one domain could be utilized by other domains, which could reduce the computational cost significantly, and more importantly, may accelerate the convergency of optimization problems in every domain.

5) Generality on problems with over two tasks: In order to further evaluate the generality of EMHH, experiments on problems with over two tasks are conducted in this subsection. The test set includes a) six three-task problems and two five-task problems, with all uETTP tasks; and b) two four-task cross-domain problems with two uETTP tasks and two GCP tasks. Tables SIV, SV, and SVI in the Supplementary Material present the three-task, four-task, and five-task test sets, respectively. The design of these test sets is to cover problems with tasks of various scales, i.e., dimensions. For example, the scales of tasks in Set 1 are close with each other, while the middle-scale task tre92 in Set 2 and large-scale instances task car91 in Set 3 are of even larger differences among tasks in these two sets. We keep all parameters consistent with the above experiments, and 30 independent runs are conducted for each test set. The experimental results on Set 1-6, Set 7-8, and Set 9-10 are shown in Table SVII, SVIII, and SIX, respectively, in the Supplementary Material.

As can be seen from Tables SVII, the proposed EMHH outperforms SOHH in all three-task test sets, except that in Set 5 the SOHH obtains better average objective value in f_1. The results shown in Tables SIX also demonstrate that EMHH can still perform better than SOHH on five-task test sets. Moreover, from Table SVIII we can see that EMHH performs equally with SOHH on GCPs but slightly better than SOHH on uETTP tasks. Given that the total generations used in EMHH and SOHH are identical, the average computational resource allocated to each task in three, four, and five-task problems in EMHH is only one-third, a quarter, and one-fifth, respectively, of that used in SOHH. Based on these observations, we can conclude that the proposed EMHH still has good generality on problems with more than two tasks.

6) Comparisons of EMHH with other hyper-heuristics: The above comparisons can be regarded as the comparison between the conventional evolutionary algorithm (EA)-based and the multitask EA-based hyper-heuristics. In this subsection, we compare our approach with other state-of-the-art hyper-heuristics in the literature on both uETTPs and GCPs. The chosen comparing algorithms include selection constructive hyper-heuristics that work on graph heuristics and were evaluated on Carter benchmarks. The comparisons on uETTPs and GCPs are provided in Tables VI and VII, respectively. Note that this comparison is just to provide an overall performance evaluation on the proposed method from another point of view. There might be other indicators or criteria which may provide comparisons on different aspects of the algorithms.

Table VI presents the comparisons between EMHH and the selected algorithms. For a better understanding the performance of EMHH, three indicators are used in the last three columns of Table VI. $f_{i-min-min}$ refers to the best of the minimum objective value obtained by EMHH in solving intra-domain MFO problems that consist of pure uETTP tasks shown in Table I for each instance. $f_{i-avg-min}$ represents the average of the minimum objective value of each instance in solving these MFO problems. The minimum objective value produced by EMHH in solving the cross-domain MFO problems, i.e. problems that comprise of uETTP and GCP tasks are denoted as f_{c-min}. In the last two rows of Table VI, average ranks over all instances (INS (16)) and instances excluding the ear83 IIc, hec92 II, sta83 IIc, and uta92 II (INS (12)) for all compared algorithms based on their best performance are provided. Due to lacks of competitive algorithms in the existing literature, the INS(16) ranks in Table VI are provided particularly for the convenience of future comparisons. Details of the ranks for each compared algorithm on each instance can be found in Table SII in the Supplementary Material.

From Table VI, we can see that EMHH achieves the 2^{nd}

TABLE III
AVERAGE TIME CONSUMED BY SOHH IN SOLVING CARTER BENCHMARKS (IN MINUTES).

Instance	sta83 IIc	hec92 II	sta83	hec92	ear83 IIc	yor83	ear83	ute92
Av. Time	1.03	1.09	1.36	1.62	2.04	2.64	3.48	4.42
No. of exams	138	80	139	81	189	181	190	184
Instance	tre92	lse91	kfu93	rye93	car92	uta92 II	uta92	car91
Av. time	10.77	13.99	26.08	51.52	85.69	95.86	111.49	112.46
No. of exams	261	381	461	486	543	638	622	682

TABLE IV
COMPARSIONS BETWEEN EMHH AND SOHH, WHERE IN EMHH THE SECOND TASK IS TRIGGERED IN LATE GENERATIONS.

MFOs	EMHH	SOHH	p-value	MFOs	EMHH	SOHH	p-value	MFOs	EMHH	SOHH	p-value
	F2	F2			F2	F2			F2	F2	
hec92	5.19	5.18		hec92	15.05	15.37		hec92	8.55	8.78	
car91	5.22	5.26	1.49E-04	kfu93	15.54	15.68	2.38E-03	tre92	8.79	8.88	2.58E-06
	0.03	0.04			0.17	0.15			0.08	0.06	
hec92	4.33	4.31		hec92	11.29	11.65		hec92	3.38	3.39	
car92	4.39	4.41	8.13E-03	lse91	11.61	11.85	5.46E-06	uta92	3.42	3.43	5.35E-01
	0.03	0.04			0.16	0.16			0.02	0.02	
hec92	35.88	36.62		hec92	9.36	9.34		hec92	3.37	3.35	
ear83	36.75	37.12	2.21E-03	rye93	9.59	9.55	6.57E-02	uta92 II	3.41	3.41	8.48E-01
	0.54	0.25			0.1	0.08			0.02	0.03	
hec92	39.58	39.51		hec92	158.61	169.77		hec92	27.9	28.31	
ear83 IIc	40.71	41.13	9.19E-02	sta83	159.35	169.86	2.95E-11	ute92	28.4	28.75	5.53E-08
	0.7	1			0.46	0.09			0.24	0.14	
hec92	11.51	11.67		hec92	34.28	34.26		hec92	40.21	41.05	
hec92 II	11.95	12.1	7.98E-02	sta83 IIc	34.72	34.99	7.73E-06	yor83	41.74	42.87	7.29E-03
	0.19	0.29			0.2	0.22			0.55	2.91	

TABLE V
THE MINIMUM, AVERAGE AND STANDARD DEVIATIONS FOR CROSS-DOMAIN MFO PROBLEMS.

MFOs	EMHH		SOHH		MFOs	EMHH		SOHH	
	F1	F2	F1	F2		F1	F2	F1	F2
car91	5.17	30	5.18	30	rye93	9.45	21	9.34	21
car91	5.23	31	5.26	31	rye93	9.59	22	9.55	22
	0.03		0.04			0.08		0.08	
car92	4.32	29	4.31	29	sta83	158.54	13	169.77	13
car92	4.38	30	4.41	30	sta83	159.35	13	169.86	13
	0.03		0.04			0.35		0.09	
ear83	35.96	22	36.62	22	sta83 IIc	34.16	35	34.26	35
ear83	36.91	23	37.12	23	sta83 IIc	34.64	35	34.99	35
	0.71		0.25			0.22		0.22	
ear83 IIc	39.02	23	39.51	23	tre92	8.61	20	8.78	20
ear83 IIc	40.19	24	41.13	24	tre93	8.81	20	8.88	20
	0.61		1			0.07		0.06	
hec92	11.52	17	11.69	17	uta92	3.4	31	3.39	31
hec92	12.05	18	12.12	18	uta92	3.43	31	3.43	31
	0.26		0.27			0.02		0.02	
hec92 II	11.43	17	11.67	17	uta92 II	3.36	30	3.348	30
hec92 II	11.93	18	12.1	18	uta92 II	3.41	31	3.41	31
	0.23		0.29			0.02		0.03	
kfu93	15.15	19	15.37	19	ute92	27.59	10	28.31	10
kfu93	15.58	19	15.68	19	ute92	28.17	10	28.75	10
	0.16		0.15			0.29		0.14	
lse91	11.26	17	11.65	17	yor83	40.7	20	41.05	20
lse91	11.67	17	11.85	17	yor83	42.04	20	42.87	21
	0.21		0.155			0.67		2.91	

and 6th place out of 14 competitors based on $f_{i-min-min}$ and $f_{i-avg-min}$ regarding the INS (12) ranks, respectively. This indicates that the EMHH attains competitive results on the intra-domain MFO problems. Particularly, given that EMHH optimizes multiple instances in a single run, the results strongly demonstrate a high generality of EMHH. Besides, EMHH ranks the 9th place based on f_{c-min} in Table VI, and from Table VII one can verify that it obtains competitive results on GCPs as well. Thus, the proposed EMHH has better generality than existing hyper-heuristics given the competitive results and its capability of tackling instances from different problem domains simultaneously, which has not been addressed by other hyper-heuristics in the literature.

E. *Preliminary Discussions*

To analyze the reason for the success of the EMHH, the landscape of the heuristics search space consists of sequences of graph heuristics for the tested problems should be analyzed first. In previous work [34], the trends of hybridizing LWD with SD in the obtained best heuristic sequences for both uEPPTs and GCPs were statistically analyzed. The visualized results indicated that the best heuristic sequences vary significantly among different instances whether from the same or different problem domain. However, the overall trend is that they employ more LWD in the early stages of the solution construction than in the later stages. This phenomenon can be treated as a similar pattern of different problems in the heuristic space. Thus, one of our conjecture of the reason for the success of the EMHH is that the knowledge transfer mechanism in it can promote the exchange of the learned patterns among tasks, which eventually facilitate the convergence of all tasks.
Furthermore, according to [38], the landscape of the heuristics search space for uETTPs has following features: *big valley structure*, *large number of local optima*, *high ruggedness*, *wide plateaus*, *shallow valleys*, and *positional bias*. Based on these features, it can be inferred that search methodologies

TABLE VI
COMPARISONS BETWEEN EMHH AND A SELECTION OF ALGORITHMS ON UETTPS.

Instance	Asm04	Asm07	Asm09	Bur07	Pil07	Pil09a	Pil09b	Qu09a	Qu09b	Sab12	Qu15	EMHH		
						f_{min}						$f_{i-min-min}$	$f_{i-avg-min}$	f_{c-min}
car91	5.29	5.19	5.29	5.36	–	4.97	–	5.3	5.11	5.14	4.95	5.11	5.15	5.17
car92	4.56	4.32	4.54	4.53	–	4.28	–	4.7	4.32	4.7	4.09	4.27	4.32	4.32
ear83	37.02	36.16	37.02	37.92	36.74	36.86	37.39	35.54	35.56	37.86	34.97	35.31	35.75	35.96
ear83 IIc	–	–	–	–	–	–	–	–	39.38	–	–	38.94	39.31	39.02
hec92	11.78	11.6	11.78	12.25	11.55	11.85	11.43	11.78	11.62	11.9	11.11	11.45	11.59	11.51
hec92 II	–	–	–	–	–	–	–	–	11.5	–	–	11.39	11.57	11.43
kfu93	15.81	15.03	15.8	15.2	14.22	14.62	–	15.09	15.18	15.3	14.09	14.79	15.14	15.15
lse91	12.09	11.35	12.09	11.33	10.9	11.14	–	12.71	11.32	12.33	10.71	11.07	11.26	11.26
rye93	10.35	9.75	10.38	–	9.35	9.65	–	–	–	10.71	9.2	9.3	9.39	9.45
sta83	160.42	158.64	160.42	158.19	158.22	158.33	158.38	159.2	158.88	160.12	157.64	158.5	158.62	158.54
sta83 IIc	–	–	–	–	–	–	–	–	34.32	–	–	33.74	34.16	34.15
tre92	8.67	8.47	8.67	8.75	8.48	8.48	–	8.67	8.52	8.32	8.27	8.55	8.62	8.61
uta92	3.57	3.52	3.57	3.88	–	3.4	–	3.32	3.21	3.88	3.33	3.38	3.39	3.4
uta92 II	–	–	–	–	–	–	–	–	3.45	–	–	3.34	3.37	3.36
ute92	27.28	27.55	28.07	28.01	26.65	28.88	27.31	30	28	32.67	26.18	27.2	27.52	27.59
yor83	40.66	39.25	39.8	41.37	41.57	40.74	39.96	40.24	40.71	40.53	37.88	39.59	40.54	40.7
INS (12)	9.5	6.3	10.2	10.5	5.1	6.5	5.8	9	6.6	10.3	1.2	3.6	6.3	6.9
INS (16)	–	–	–	–	–	–	–	–	3.8	–	–	1	3.3	2.3

"–" indicates that the corresponding instances are not tested or the calculation of that value is nonsense. The solutions that are compared against our approach on uETTPs include:
Asm04: Fuzzy combinations of two ordering criteria [41].
Asm07: Fuzzy combinations of multiple ordering criteria [42].
Asm09: Asm04 with turning [43].
Bur07: Tabu search [33].
Pil07: A genetic programming-based hyper-heuristic [44].
Pil09a: Four hierarchical combination of heuristics [45].
Pil09b: A genetic programming hyper-heuristic to evolve functions of low-level heuristic and logical operators [46].
Qu09a: Four local search [47].
Qu09b: An adaptive hybridization of heuristics [34].
Sab12: A graph coloring constructive hyper-heuristics where the hybridizations of four heuristic sequence are utilized to order exams [48].
Qu15: EDA-based hyper-heuristic [5].

TABLE VII
THE MINIMUM NUMBER OF COLORS FOUND BY EMHH AND THE SELECTED ALGORITHMS.

	car91	car92	ear83	ear83 IIc	hec92	hec92 II	kfu93	lse91
Qu09	30	29	22	–	17	–	19	17
Qu15	**28**	**27**	22	–	17	–	19	17
EMHH	30	29	22	23	17	17	19	17
	rye93	sta83	sta83 IIc	tre92	uta92	uta92 II	ute92	yor83
Qu09	–	13	–	20	31	–	10	19
Qu15	21	13	–	20	**29**	–	10	18
EMHH	21	13	35	20	31	30	10	20

"–" indicates that instances are not tested. Bold values represent the best solutions among all competitors, and the optimal results are underlined. The solutions that are compared against our approach on GCPs include:
Qu09b: An adaptive hybridization of heuristics [34].
Qu15: EDA-based hyper-heuristic [5].

that work in the search space under discussion may easily get stuck in local optima, especially considering that there are wide plateaus in the search space. Therefore, we suspect part of the superiority of EMHH is that it can promote the diversity of the heuristic sequences via knowledge transfer mechanism, thus improving the possibility of jumping out of the local optima. To verify this conjecture, we have observed the best 10 heuristic sequences for all the instances based on the results of EMHH and SOHH from Table I. The MFO problem (ear83, sta83) is selected as the representative due to the significant distinction between their best 10 heuristic sequences. The selected heuristic sequences of ear83 and sta83 are shown as Fig.S2. From Fig.S2 (a) and (b) we can see that the left 3 heuristics in selected heuristic sequences of ear83 and sta83 are obviously different. Therefore, EMHH intuitively would suffer from negative transfer. However, Table I shows that EMHH achieves significantly better results than SOHH in both tasks. Furthermore, from Fig.S2 we can see that results obtained by EMHH are clearly closer to the best 10 heuristic sequences than that obtained by SOHH. Namely, SOHH is stuck in local optima while EMHH successfully escaped benefiting from the knowledge transfer mechanism. Note that in [26] the similar conjecture has been made in the permutation-based encoding search space, but the verification was not provided. Last but not the least, there may be negative transfers [18] in the process of knowledge transfer, as in the cases of MFO problems that including rye93, where SOHH achieved better results than EMHH on rye93.

F. *Potential Applications and Future Research Challenges*

Other educational timetabling problems, besides examination timetabling studied in this paper, are possible applications that require multitask hyper-heuristics. Another scenario is the problem of cloud services, where the server is likely to face concurrent service requests from multiple customers. The optimization of these requests might have different scales or even belong to different domains. In this regard, the multitask hyper-heuristics present likely an appropriate approach that can provide good services with a reduced computational cost from the multitasking and knowledge transfer schemes. In addition, the applications of hyper-heuristics as presented in [10], such as vehicle routing problems, nurse rostering problems and packing problems might also be potential applications.

As mentioned in the Introduction, hyper-heuristic can be classified into four categories, namely, selection constructive/perturbative, generation constructive/perturbative. This paper concerns the selection constructive hyper-heuristic as the precursor of extending the hyper-heuristics into multitasking scheme. Thus, extensions of other traditional hyper-heuristics remain to be studied. Here, we point out several challenges that one may encounter in these extensions. First, the design of the unified representation. Although so-called low-level heuristics constitute the search space of hyper-heuristics, they are domain-specific; that is, the graph heuristics used to solve educational timetabling problems might not be suitable to solve other combinational problems. Thus, careful attention should be paid for the design of the unified representation when handling problems with very distinct heuristics. Moreover, much effort is needed on multitasking hyper-heuristics that combine different categories of them; for example, selection and generation constructive hyper-heuristics. Second, the design of efficient knowledge transfer schemes is challenging as well.

V. Conclusion

In this paper, a unified framework of evolutionary multitasking graph-based hyper-heuristic (EMHH) is proposed, where the concept of evolutionary multitasking and graph heuristics are used as the high-level search methodology and low-level heuristics, respectively. The EMHH has been evaluated on uncapacitated exam timetabling and graph coloring problems. Given that the purpose of this paper is to propose a new hyper-heuristic framework instead of competing with certain algorithms on specific problems, we found the obtained results encouraging.

In conclusion, the superiorities of the proposed EMHH comparing with the conventional single-tasking hyper-heuristic are two-fold: (1) It raises the generality of both hyper-heuristic and evolutionary multitasking to a higher level; i.e., extend the generality of hyper-heuristics in addressing multiple optimization problems and the scope of unification in evolutionary multitasking. (2) The EMHH is more effective and efficient. To be specific, on one hand, it can take advantages of the commonalities among tasks to facilitate the convergence of the algorithm. On the other hand, the search biases provided by different tasks via the knowledge transfer mechanism can promote the diversity in the heuristic space, thus improving the global search ability of the EMHH.

In the future, we will investigate the performance of EMHH on other problem domains. The properties of common structures of solutions in high-level space need to be further examined. The design of more effective mechanisms to adapt the reusable knowledge from one domain to other domains could be another interesting future research direction.

References

[1] K. Hussain, M. N. Mohd Salleh, S. Cheng, and Y. Shi, "Metaheuristic research: a comprehensive survey," *Artificial Intelligence Review*, pp. 1–43, 2018. [Online]. Available: https://doi.org/10.1007/s10462-017-9605-z

[2] M. Gendreau and J.-Y. Potvin, *Handbook of metaheuristics*. Springer, New York, 2010.

[3] K. Chakhlevitch and P. Cowling, "Hyperheuristics : Recent Developments," in *Adaptive and Multilevel Metaheuristics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 3–29.

[4] P. Cowling, G. Kendall, and E. Soubeiga, "A Hyperheuristic Approach to Scheduling a Sales Summit," in *International Conference on the Practice and Theory of Automated Timetabling*, E. Burke and W. Erben, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 176–190.

[5] R. Qu, N. Pham, R. Bai, and G. Kendall, "Hybridising heuristics within an estimation distribution algorithm for examination timetabling," *Applied Intelligence*, vol. 42, no. 4, pp. 679–693, 2015.

[6] E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg, *Hyper-Heuristics: An Emerging Direction in Modern Search Technology*. Boston, MA: Springer US, 2003, pp. 457–474. [Online]. Available: https://doi.org/10.1007/0-306-48056-5{_}16

[7] R. Qu, E. K. Burke, B. McCollum, L. T. Merlot, and S. Y. Lee, "A survey of search methodologies and automated system development for examination timetabling," *Journal of Scheduling*, vol. 12, no. 1, pp. 55–89, 2009.

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

[9] N. Pillay, "A review of hyper-heuristics for educational timetabling," *Annals of Operations Research*, vol. 239, no. 1, pp. 3–38, 2016.

[10] N. Pillay and R. Qu, *Hyper- Heuristics : Theory and Applications*. Springer International Publishing, 2018.

[11] M. Bader-El-Den and R. Poli, "Generating SAT Local-Search Heuristics Using a GP Hyper-Heuristic Framework," in *International Conference on Artificial Evolution (Evolution Artificielle)*, N. Monmarché, E.-G. Talbi, P. Collet, M. Schoenauer, and E. Lutton, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 37–49.

[12] S. A. Bittle and M. S. Fox, "Learning and Using Hyper-heuristics for Variable and Value Ordering in Constraint Satisfaction Problems," in *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, ser. GECCO '09. New York, NY, USA: ACM, 2009, pp. 2209–2212. [Online]. Available: http://doi.acm.org/10.1145/1570256.1570304

[13] P. Garrido and M. C. Riff, "DVRP: A hard dynamic combinatorial optimisation problem tackled by an evolutionary hyper-heuristic," *Journal of Heuristics*, vol. 16, no. 6, pp. 795–834, 2010.

[14] D. Meignan, A. Koukam, and J. C. Créput, "Coalition-based metaheuristic: A self-adaptive metaheuristic using reinforcement learning and mimetism," *Journal of Heuristics*, vol. 16, no. 6, pp. 859–879, 2010.

[15] E. K. Burke, M. R. Hyde, G. Kendall, and J. Woodward, "Automatic Heuristic Generation with Genetic Programming: Evolving a Jack-of-all-trades or a Master of One," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '07. New York, NY, USA: ACM, 2007, pp. 1559–1565. [Online]. Available: http://doi.acm.org/10.1145/1276958.1277273

[16] E. K. Burke, M. R. Hyde, and G. Kendall, "Grammatical evolution of local search heuristics," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 3, pp. 406–417, 2012.

[17] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "A Classification of Hyper-heuristic Approaches," in *Handbook of Metaheuristics*. Springer, Boston, MA, 2010, pp. 449—-468. [Online]. Available: http://link.springer.com/10.1007/b101874

[18] 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, 2016.

[19] A. Gupta, J. Mańdziuk, and Y.-S. Ong, "Evolutionary multitasking in bi-level optimization," *Complex & Intelligent Systems*, vol. 1, no. 1, pp. 83–95, Dec 2015. [Online]. Available: https://doi.org/10.1007/s40747-016-0011-y

[20] 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, 2018. [Online]. Available: http://ieeexplore.ieee.org/document/8114198/

[21] R. Chandra, A. Gupta, Y. S. Ong, and C. K. Goh, "Evolutionary Multi-task Learning for Modular Knowledge Representation in Neural Networks," *Neural Processing Letters*, vol. 47, no. 3, pp. 993–1009, 2018.

[22] L. Zhou, L. Feng, Jinghui Zhong, Y. Ong, Z. Zhu, and E. Sha, "Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, Dec 2016, pp. 1–8.

[23] Y. Yuan, Y. S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems:

Realization with TSP, QAP, LOP, and JSP," in *IEEE Region 10 Annual International Conference, Proceedings/TENCON*, 2017, pp. 3157–3164.

[24] 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, Feb 2019.

[25] A. Gupta, Y. S. 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.

[26] Y. Yuan, Y.-S. Ong, A. Gupta, P. S. Tan, and H. Xu, "Evolutionary multitasking in permutation-based combinatorial optimization problems: Realization with tsp, qap, lop, and jsp," in *2016 IEEE Region 10 Conference (TENCON)*. IEEE, 2016, pp. 3157–3164.

[27] H. Li, Y. Ong, M. Gong, and Z. Wang, "Evolutionary multitasking sparse reconstruction: Framework and case study," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2018. [Online]. Available: https://ieeexplore.ieee.org/document/8540026

[28] L. Feng, L. Zhou, J. Zhong, A. Gupta, Y. Ong, K. Tan, and A. K. Qin, "Evolutionary multitasking via explicit autoencoding," *IEEE Transactions on Cybernetics*, vol. 49, no. 9, pp. 3457–3470, Sep. 2019.

[29] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary Multitasking with Dynamic Resource Allocating Strategy," *IEEE Transactions on Evolutionary Computation*, pp. 1–1, 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8616832/

[30] N. Pillay, "A review of hyper-heuristics for educational timetabling," *Annals of Operations Research*, vol. 239, no. 1, pp. 3–38, 2016.

[31] M. W. Carter, G. Laporte, and S. Y. Lee, "Examination Timetabling: Algorithmic Strategies and Applications," *Journal of the Operational Research Society*, vol. 47, no. 3, pp. 373–383, 1996. [Online]. Available: https://doi.org/10.1057/jors.1996.37

[32] D. J. A. Welsh and M. B. Powell, "An upper bound for the chromatic number of a graph and its application to timetabling problems," *The Computer Journal*, vol. 10, no. 1, pp. 85–86, 1967. [Online]. Available: https://dx.doi.org/10.1093/comjnl/10.1.85

[33] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu, "A graph-based hyper-heuristic for educational timetabling problems," *European Journal of Operational Research*, vol. 176, no. 1, pp. 177–192, 2007.

[34] R. Qu, E. K. Burke, and B. McCollum, "Adaptive automated construction of hybrid heuristics for exam timetabling and graph colouring problems," *European Journal of Operational Research*, vol. 198, no. 2, pp. 392–404, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.ejor.2008.10.001

[35] L. Merlot, N. Boland, B. Hughes, and P. Stuckey, "A Hybrid Algorithm for the Examination Timetabling Problem," in *International Conference on the Practice and Theory of Automated Timetabling*. Springer, Berlin, Heidelberg, 2003, pp. 207–231.

[36] E. K. Burke, J. R. Newall, and R. E. Weare, "A memetic algorithm for university exam timetabling," in *international conference on the practice and theory of automated timetabling*. Springer, Berlin, Heidelberg, 1996, pp. 241–250.

[37] E. K. Burke, R. F. Weare, and J. P. Newall, "Initialization strategies and diversity in evolutionary timetabling," *Evolutionary Computation*, vol. 6, no. 1, pp. 81–103, 1998.

[38] G. Ochoa, R. Qu, and E. K. Burke, "Analyzing the landscape of a graph based hyper-heuristic for timetabling problems," in *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*. ACM, 2009, pp. 341—-348. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1569901.1569949

[39] P. Kora and P. Yadlapalli, "Crossover Operators in Genetic Algorithms: A Review," *International Journal of Computer Applications*, vol. 162, no. 10, pp. 34–36, 2017. [Online]. Available: http://www.ijcaonline.org/archives/volume162/number10/kora-2017-ijca-913370.pdf

[40] E. K. Burke and J. P. Newall, "Solving Examination Timetabling Problems through Adaption of Heuristic Orderings," *Annals of Operations Research*, vol. 129, pp. 107–134, 2004.

[41] H. Asmuni, E. K. Burke, J. M. Garibaldi, and B. Mccollum, "Fuzzy Multiple Ordering Criteria for Examination Timetabling," in *International Conference on the Practice and Theory of Automated Timetabling*, 2004, pp. 147–160.

[42] H. Asmuni and E. Burke, "Determining rules in fuzzy multiple heuristic orderings for construction examination timetables," in *Proceedings of the 3rd multidisciplinary international conference on scheduling: theory and application*, 2007, pp. 59–66. [Online]. Available: http://www.mistaconference.org/2007/papers/DeterminingRulesinFuzzyMultipleHeuristicOrderings.pdf

[43] H. Asmuni, E. K. Burke, J. M. Garibaldi, B. McCollum, and A. J. Parkes, "An investigation of fuzzy multiple heuristic orderings in the construction of university examination timetables," *Computers and Operations Research*, vol. 36, no. 4, pp. 981–1001, 2009.

[44] N. Pillay and W. Banzhaf, "A Genetic Programming Approach to the Generation of Hyper-Heuristics for the Uncapacitated Examination Timetabling Problem," in *Portuguese Conference on Artificial Intelligence*. Springer Berlin Heidelberg, 2007, pp. 223–234.

[45] ——, "A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem," *European Journal of Operational Research*, vol. 197, no. 2, pp. 482–491, 2009. [Online]. Available: http://dx.doi.org/10.1016/j.ejor.2008.07.023

[46] N. Pillay, "Evolving hyper-heuristics for the uncapacitated examination timetabling problem," *Journal of the Operational Research Society*, 2012.

[47] R. Qu and E. K. Burke, "Hybridizations within a graph-based hyper-heuristic framework for university timetabling problems," *Journal of the Operational Research Society*, vol. 60, no. 9, pp. 1273–1285, 2009.

[48] N. R. Sabar, M. Ayob, R. Qu, and G. Kendall, "A graph coloring constructive hyper-heuristic for examination timetabling problems," *Applied Intelligence*, vol. 37, no. 1, pp. 1–11, 2012.