Automated Coordination Strategy Design using Genetic Programming for Dynamic Multi-Point Dynamic Aggregation

Guanqiang Gao, Yi Mei, Senior Member, IEEE, Bin Xin, Member, IEEE, Ya-hui Jia, Member, IEEE, and Will N. Browne, Member, IEEE

Abstract—The multi-point dynamic aggregation (MPDA) problem of the multi-robot system is of great significance for its real-world applications like bush fire elimination. The problem is to design the optimal plan for a set of heterogeneous robots to complete some geographically distributed tasks collaboratively. In this paper, we consider the dynamic version of the problem where new tasks keep appearing after the robots are dispatched from the depot. The dynamic MPDA problem is a complicated optimization problem due to several characteristics, such as the collaboration of robots, the accumulative task demand, the relationships among robots and tasks, and the unpredictable task arrivals. In this paper, a new model of the problem considering these characteristics is proposed. To solve the problem, we develop a new genetic programming hyper-heuristic (GPHH) method to evolve reactive coordination strategies (RCS) which can guide the robots to make decisions in real-time. The proposed GPHH method contains a newly designed effective RCS heuristic template to generate the execution plan for the robots according to a GP tree. A new terminal set of features related to both robots and tasks, and a cluster filter which assigns the robots to urgent tasks are designed. The experimental results show that the proposed GPHH significantly outperformed the state-of-the-art methods. Through further analysis, useful insights such as how to distribute and coordinate robots to execute different types of tasks are discovered.

Index Terms—Genetic programming, hyper-heuristic, multi-robot system, multi-point dynamic aggregation, real-time decision-making

I. INTRODUCTION

The Multi-Point Dynamic Aggregation (MPDA) problem is an important optimization problem that has many real-world applications [1]–[3]. The objective of MPDA is to design a routing plan for a set of robots to complete some distributed tasks in a collaborative way so that the makespan (the completion time of the last task) is minimised. A typical example is the robot fire-fighting planning problem. In the past years, much attention has been given to the fire-fighting problem (e.g. the bush fire 2019 in Australia and the bush fire 2020 in California) [4], [5] since fire causes enormous damage to the economy and natural environments. Because of good mobility and safety [6], multi-robot systems (e.g. unmanned aerial/ground vehicles) have been considered to execute fire-fighting missions [1], [7]–[10]. In addition to the fire-fighting application, MPDA also has other real-world applications, such as intelligent security [11], search and rescue [12], and environment monitoring [13].

So far, most studies for MPDA have been focused on static environments, in which all the information regarding the robots and tasks is known in advance [2], [3], [14]. However, in the real world, the environment is dynamic and the information about the new tasks is unknown until the tasks are detected. For example, Fig. 1 shows a dynamic scenario of a multi-robot fire-fighting mission. Three robots are assigned to execute tasks 1, 2 and 3. Tasks 4 and 5 are yet not known to the robots, as they have not been detected/arrived at yet. In this paper, we will focus on solving the Dynamic MPDA (DMPDA) problem with dynamic task arrivals.

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1308000; in part by the National Natural Science Foundation of China under Grant 61720106011; in part by the Beijing Advanced Innovation Center for Intelligent Robots and Systems; and in part by the Peng Cheng Laboratory. (Corresponding author: Bin Xin.)

Guanqiang Gao and Bin Xin are with the School of Automation, Beijing Institute of Technology, Beijing 100081, China; Yi Mei, Ya-hui Jia, and Will N. Browne are with School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand.
cumulated demands of the remaining tasks. This characteristic of the problem makes the search space very rugged. A slight change of the solution could lead to a very different makespan. Third, allowing collaboration between robots (each task can be visited by multiple robots) generates a much larger search space than that of the non-collaborative path-finding problems such as vehicle routing problems [16], [17].

Reactive coordination strategies (RCSs) have been shown as a promising heuristic for the multi-robot system [6], [18], [19]. RCS is used in the execution process of the mission, to guide the robots’ activities (e.g., decide the next task for a robot to execute when it completes the current task) based on the analysis and synthesis of the environment and task information [20], [21]. Since RCS can decide the next task for the robot to execute in a very short time, they are very suitable for solving the DMPDA problem. There have been a variety of manually designed RCSs for MPDA [1], [22], [23], which have achieved promising results.

Designing RCSs for the DMPDA problem needs to consider various factors related to both robots and tasks. Not only are high expertise and domain knowledge required, but different RCSs should be designed for different scenarios. Thus, manually designing RCSs is very difficult.

Genetic programming Hyper-heuristic (GPHH) methods have been used to automatically generate rules for many dynamic problems [24]–[27], such as dynamic job shop scheduling and dynamic vehicle routing. With its ability to evolve sophisticated rules and flexible representations [28], [29], GPHH is a potential method for automatically generating better RCSs than the manually designed RCSs. To the best of our knowledge, no GPHH method has been designed for evolving RCSs for the DMPDA problem.

Before investigating the DMPDA problem, there are some other difficulties. First, there is no proper model that can characterise the problem so far. Second, there is no existing benchmark test set that can be used to evaluate the performance of an algorithm. Third, the existing execution process for robots to complete the tasks is not effective enough to deal with the incremental demands of the tasks. Especially, a little delay on one task leads to a much larger demand of the other tasks. This issue needs to be addressed in the execution process. Finally, an effective RCS needs to be designed and embedded in the execution process. To address these issues and also solve the DMPDA problem effectively, we establish a DMPDA model, design a DMPDA benchmark set, and propose a novel GPHH method. Specifically, the paper has the following contributions.

1) A DMPDA problem model is proposed and investigated:
   • To describe the DMPDA problem comprehensively, a novel model is proposed in this paper. The proposed model considers the time-varying task demands, the collaboration between tasks, and the dynamic environment where the tasks cannot be known until they are detected.
   • To test different algorithms, a new benchmark set is designed. The benchmark set contains a wide range of instances with different characteristics, such as the number of robots and tasks, robot abilities, and detection time of the dynamic tasks.

2) An effective GPHH algorithm is designed:
   • We design a new heuristic template of robots to effectively handle the complex relationship among the robots and tasks, and the dynamic task arrivals. The heuristic template can be considered as the simulation of the DMPDA problem.
   • A new filter is designed for the candidate task pool to adaptively switch between distributing and grouping the robots during the RCS heuristic template processing. The new filter can prevent a task demand from growing too large to be completed in time.
   • To evolve RCS, we design a terminal set including features of the robots and tasks for the GPHH approach.
   • We analyse the structure of the rules and solutions obtained by the proposed GPHH algorithm to gain insights from the useful patterns in the promising DMPDA solutions.

The rest of this paper is organised as follows. Section II presents the mathematical model for the DMPDA problem and reviews the related work. Afterwards, the proposed GPHH is described in Section III. Sections IV–VI present the experimental studies, including empirical comparisons with manually designed heuristics, the demonstration of the efficacy of the proposed strategies, and the insights of the evolved RCSs. Finally, conclusions are drawn in Section VII.

II. BACKGROUND

This section first briefly describes the problem definition of the DMPDA problem, then introduces the related work about MPDA and GPHH.

A. Problem Definition

An undirected graph $G(V, E)$ is used to define the DMPDA problem. In the undirected graph, $V = \{v_0, v_1, \ldots, v_N\}$ represents the set of vertices and $E$ is the set of edges. $v_0$ indicates the depot. Each vertex $v_j \in V$ is a task and is associated with three characteristics: the detection/arrival time $t_{0,j}$, the inherent increment rate $\gamma_j$, and the initial demand $q_{0,j}$ when it is detected. The accumulated demand $q_i(t)$ of $v_i$ at time $t$ is shown as follows.

$$q_i(t) = q_{0,i} + \gamma_i \times (t - t_{0,i}). \quad (1)$$

Each edge $e_{ij} = (v_i, v_j) \in E$ is associated with a travel time $t_{ij}$ from $v_i$ to $v_j$.

There is a set of robots $R = \{rob_1, \ldots, rob_M\}$ to execute these tasks. The robots are initially located at the depot $v_0$. Each robot $rob_k$ has an ability $u_k$, indicating the amount of demand that the robot can reduce per time unit.

Fig. 2 gives an example to show how the robots reduce the accumulated demand of a task over time. From Fig. 2, the task is detected at time 2, its inherent increment rate $\gamma$ is 1, and its initial demand is 2. Its accumulated demand increases with its inherent increment rate from time 2 to 4, and reaches 4 at time 4. $rob_1$ with the ability of 0.5 arrives at the task at time 4 and starts to execute it, i.e., reduce its demand. The demand increment rate becomes 1-0.5=0.5. At time 7, the accumulated demand reaches 5.5, and $rob_2$ with the ability of 1.5 arrives
at the task to execute it. The demand increment rate becomes 0.5-1.5=1, i.e., the demand starts to decrease by 1 per time unit. Finally, the demand is decreased to 0 at time 12.5, and the task is completed by these two robots.

The problem is to design the execution plan of robots to complete all the tasks subject to the following constraints.

- A task is unseen by the planning system until it is detected.
- The number of incoming paths equals the number of outgoing paths for each robot at each task.

\[
\sum_{j=0}^{N} x_{ij} = \sum_{j=0}^{N} x_{ji}, \forall i = 1, \ldots, N, \forall k = 1, \ldots, M \tag{2}
\]

where the binary decision variable \( x_{ij} \) takes 1 if there is a path for \( rob_k \) to go from \( v_i \) to \( v_j \), and 0 otherwise.

- Each task is executed by each robot at most once.

\[
\sum_{i=0}^{N} x_{ij} \leq 1, \forall j = 1, \ldots, N, \forall k = 1, \ldots, M \tag{4}
\]

- A task cannot be executed before it is detected.

\[
st_{jk} \geq t_{0,j}, \forall j = 1, \ldots, N, \forall k = 1, \ldots, M \tag{5}
\]

where \( st_{jk} \) represents the start time of \( rob_k \) at \( v_j \).

- The demand of each task must decrease to zero to be completed.

\[
st_{jk} = \sum_{i=0}^{N} (ct_i + t_{ij}) x_{ij}^k, \forall j = 1, \ldots, N, \forall k = 1, \ldots, M \tag{6}
\]

\[
q_{0,j} + \gamma_j (ct_j - t_{0,j}) = \sum_{i=0}^{N} \sum_{k=1}^{M} x_{ij}^k \mu_k (ct_j - st_{jk}), \forall j = 1, \ldots, N \tag{7}
\]

where \( ct_j \) represents the completion time of \( v_j \). (6) implies that a robot immediately goes to its next task after it completed its current task. (7) implies that the accumulated demand at \( ct_j \) (when the task is completed) equals the total demand fulfilled by the robots executing the task till \( ct_j \). (8) implies that for every task, the total ability of the robots executing the task must be greater than the inherent increment rate of the task so that the task can be completed.

- The start time and completion time of all the robots at the depot are 0.

\[
st_{0k} = ct_0 = 0, \forall k = 1, \ldots, M \tag{9}
\]

Given an instance \( \xi \) and its solution \( X_{\xi} \) (all the \( x_{ij}^k \) values), the makespan (completion time of the last task) is:

\[
C(X_{\xi}, \xi) = \max_{i \in \{1, \ldots, N\}} ct_i \tag{10}
\]

where \( ct_i \) is calculated from (7)-(9) based on \( \xi \) and \( X_{\xi} \).

The objective of the DMPDA problem is to minimize the average performance on a set of instances, which is shown as follows:

\[
\min E(C(X_{\xi}, \xi)) | \xi \in \Xi, \tag{11}
\]

where \( \Xi \) represents the set of instances.

To better understand the DMPDA problem, an example of a solution on an instance with two robots is shown in Fig. 3. At time \( t_0 \), there are 4 tasks known to the robots, and the robots are planned with two execution paths: (Depot \( \rightarrow A \rightarrow C \rightarrow \text{Depot} \)) and (Depot \( \rightarrow B \rightarrow C \rightarrow \text{Depot} \)). In this execution plan, the two robots collaborate to complete tasks C and D. At time \( t_1 \), two new tasks E and F are detected. Then, the two robots re-plan their paths to complete all the tasks: (Depot \( \rightarrow A \rightarrow E \rightarrow G \rightarrow F \rightarrow \text{Depot} \)) and (Depot \( \rightarrow B \rightarrow C \rightarrow D \rightarrow G \rightarrow F \rightarrow \text{Depot} \)). At the final time \( t_2 \), the tasks are all completed, and the two robots return to the depot. It should be noticed that although the return of robots is shown in the example, the returning time is not considered in the model since in practice, compared with the time used to execute the tasks, the returning time is not important.

### B. Related work

1) Multi-Point Dynamic Aggregation: MPDA originates from the multi-robot system, and it determines how to allocate robots to tasks to optimize the performance of the multi-robot system [1], [11]–[13]. Due to the complex relationships among robots and tasks caused by the time-varying demand and coordinated execution, MPDA is a very challenging and interesting problem. Over the past few years, researchers proposed several methods to address the problem [2], [22], [30]. These methods

### TABLE I

<table>
<thead>
<tr>
<th>Notation</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( i, j )</td>
<td>index</td>
<td>the indexes of tasks</td>
</tr>
<tr>
<td>( k )</td>
<td>index</td>
<td>the index of robots</td>
</tr>
<tr>
<td>( N )</td>
<td>value</td>
<td>the number of tasks</td>
</tr>
<tr>
<td>( M )</td>
<td>value</td>
<td>the number of robots</td>
</tr>
<tr>
<td>( q_i(t) )</td>
<td>value</td>
<td>the demand of ( v_i ) accumulated at time ( t )</td>
</tr>
<tr>
<td>( t_{0,i} )</td>
<td>value</td>
<td>the detection time ( v_i )</td>
</tr>
<tr>
<td>( \gamma_i )</td>
<td>value</td>
<td>the inherent increment rate of ( v_i )</td>
</tr>
<tr>
<td>( q_0,i )</td>
<td>value</td>
<td>the initial demand</td>
</tr>
<tr>
<td>( t_{x,i} )</td>
<td>value</td>
<td>travel time from ( v_i ) to ( v_j )</td>
</tr>
<tr>
<td>( \mu_k )</td>
<td>value</td>
<td>the ability of ( rob_k )</td>
</tr>
<tr>
<td>( v_i )</td>
<td>indicator</td>
<td>the task with index ( i )</td>
</tr>
<tr>
<td>( rob_k )</td>
<td>indicator</td>
<td>the robot with index ( k )</td>
</tr>
<tr>
<td>( x_{ij}^k )</td>
<td>decision variable</td>
<td>1 if ( rob_k ) goes from ( v_i ) to ( v_j ), and 0 otherwise</td>
</tr>
<tr>
<td>( st_{jk} )</td>
<td>auxiliary variable</td>
<td>the start time of ( rob_k ) at ( v_j )</td>
</tr>
<tr>
<td>( ct_j )</td>
<td>auxiliary variable</td>
<td>the completion time of ( v_j )</td>
</tr>
</tbody>
</table>
can be categorised into two kinds: manually designed reactive coordination strategy and meta-heuristic.

Manually designed reactive coordination strategy: Teng et al. [30] proposed an efficient market-based coordination strategy to assign multiple robots to extinguish a bushfire with multiple fire fronts. Xin et al. [1] proposed a distributed coordination strategy to overcome the path conflict during the MPDA mission. By using the designed coordination strategy [1], multiple robots coordinate to execute a task simultaneously, and they can avoid the obstacles and conflicts. A recruitment coordination strategy to address the MPDA problem was proposed in [22]. In the designed strategy, if the increment rate of a task is larger than the ability of the robot (meaning the robot cannot complete the task alone), more robots will be recruited to complete the task together. Chen et al. [23] proposed an algorithm that can provide a theoretically optimal task plan for small static MPDA instances when the travel cost is ignored, and proposed a task assignment approach to balance between the completion time of all the tasks.

The above manually designed coordination strategies can generate a plan quickly. However, they do not consider the dynamic task arrivals during the execution process, and may become ineffective in the DMPDA problem. Furthermore, the quality of the generated plans by the above manually RCSs are not satisfactory enough.

Meta-heuristic: Poggenpohl et al. [31] designed the task planning approach hybridised with a greedy method and a simulated annealing algorithm. The experimental results showed that the hybrid approach can lead to a better makespan and a better balance among the workloads of the robots. Another hybrid method combining differential evolution and estimation of distribution algorithm was proposed to address the MPDA planning problem by [32]. Experiments showed that this method [32] can obtain a good solution with stable convergence speed. A multi-model estimation of distribution algorithm (EDA) was designed in [15]. Compared with a genetic algorithm and a random search method, EDA [15] achieved better performance. Recently, we also proposed an adaptive coordination ant colony optimisation with an individual learning strategies to address the MPDA problem [3]. Experiments showed that the proposed ant colony optimisation algorithm significantly outperformed the previous algorithms in terms of both effectiveness and efficiency.

The above meta-heuristic approaches for the MPDA problem can obtain good performance by the evolution process. However, similar to the limitations of the manually designed coordination strategy, the aforementioned meta-heuristic methods did not consider the dynamic arrivals. It is difficult to apply them to the DMPDA problem to adjust the solution online due to their high computational cost.

2) Genetic programming hyper-heuristic: The field of automatic heuristic design using hyper-heuristics has attracted much attention recently [33]–[35]. The hyper-heuristic approaches search in the heuristic space rather than the solution space [36], [37]. The obtained heuristics can then be applied to unseen problem instances or environments.

GP is a type of evolutionary algorithm popularised by Koza in the 1990s [38]. With a flexible representation, GP is able to generate a good performance heuristic by some blocks of domain knowledge given by the experts. In recent years, GP has been the dominating approach in the scope of generative hyper-heuristics [27], [39]–[41], which is known as GPHH. Every individual in the GP population represents a candidate heuristic to the problem, which is usually defined as a priority function with the symbolic representation. The evolutionary process to generate these heuristics also follows three steps: evaluation, selection, and genetic operations (including reproduction, crossover, and mutation) like genetic algorithms. However, since the symbolic representation is different from the regular real value representations, the crossover and mutation operations are elaborately designed which can be found in [42]. Whenever a decision is made, the priorities of candidates are calculated by the priority function, and the candidate with the highest priority is chosen. For example, in dynamic job shop scheduling [43]–[45], the evolved heuristic is used to calculate the priorities of jobs to machines. In uncertain routing problems [24], [46], [47], it is used to calculate the priorities of tasks to vehicles. The other works have also demonstrated the effectiveness of GPHH [48], [49].

When designing GPHH to evolve RCSs for the DMPDA problem, there are two main components: 1) an effective heuristic template that can generate a feasible solution (robot execution plan) based on the given DMPDA instance and RCS, and 2) an efficient GP algorithm (e.g. terminal and function sets, crossover mechanism, and selection mechanism) to evolve the RCS. In this paper, we develop a new GPHH algorithm containing these two components.

III. GENETIC PROGRAMMING HYPER-HEURISTIC FOR THE DMDPA PROBLEM

This section describes the newly proposed GPHH algorithm which evolves RCSs for DMDPA. In this section, the representation of a RCS is presented first, followed by the designed heuristic template that applies a RCS to generate an execution plan for a given DMDPA instance. Then, the evolutionary process of the proposed GPHH method is described, including the framework, fitness evaluation, and GP operators.

A. Representation

In the proposed GPHH method, each RCS (individual) is represented as a tree-based priority function. Whenever a task
Every event contains four elements, i.e.,

\[ \text{event}_i = (\text{rob}_i, v_i, t_i, C_i) \]  

where \( \text{rob}_i \) is the robot involved in the event, \( v_i \) is the task of the event, \( t_i \) is the occurrence time of the event, and \( C_i \) stands for the type of the event. Specifically, \( C_i = 1 \) means that \( \text{rob}_i \) arrives at task \( v_i \) at time \( t_i \), while \( C_i = 0 \) means that \( \text{rob}_i \) completes and departs from \( v_i \) at time \( t_i \). The heuristic template proposed in this paper is shown in Algorithms 1 and 2. It contains three stages: initialization stage (lines 1-2 in Algorithm 1), decision making stage (lines 4-16 in Algorithm 1), and execution stage (Algorithm 2).

1) **Initialization stage:** All the robots are at the depot and active, and their predicted events are set as \( (\text{rob}_k, 0, 0, 1) \). The predicted event of a robot implies the task that the robot is going to execute. All tasks are not yet completed, and the execution plan is an empty set. In addition, the simulation time \( t \) is set as 0, and the archive \( \mathcal{A} \) used to distinguish the urgent tasks is empty.

2) **Decision making stage:** Here we consider the execution period of a task for making better decisions. The (estimation) execution period \( ep_k \) of a task equals to the (estimation) completion time of the task minus the start time of first robot executing the task.

\[ ep_k = ct_k - \min_{k \in \{1, \ldots, M\}} st_{ik}, \forall k = 1, \ldots, N \]  

At the beginning of the decision making stage, a set of candidate tasks \( \Omega \) are selected from all the uncompleted tasks by a filter scheme, which is shown in Algorithm 3. There are two situations considered in the filter scheme:

- Urgent tasks are assigned to active robots mandatorily (lines 2-11 in Algorithm 3): Due to the accumulated demands of tasks in the DMPDA problem, if some tasks cannot be completed in time, it will lead to a long

![Fig. 4. An example of a reactive coordination strategy.](image-url)
Algorithm 3 The filter algorithm

**INPUT:** rob_k, uncompleted tasks \( T^{(t)} \), archive \( A \) for recording execution periods, and parameters \( \omega, \phi \).

**OUTPUT:** A set of candidate tasks \( \Omega \).

1: \( \Omega \leftarrow \emptyset \).
2: if \( |A| = \phi \) then
3: \hspace{1cm} Calculate maximal execution period \( ep^* \) from \( A \).
4: \hspace{1cm} Get uncompleted tasks \( T^{(t)}_c \) that have been executed from \( T^{(t)} \).
5: \hspace{1cm} for \( v_i \in T^{(t)}_c \) do
6: \hspace{2cm} Calculate the estimation execution period \( ep_i \) of \( v_i \).
7: \hspace{2cm} if \( ep_i > \omega \times ep^* \) then
8: \hspace{3cm} return \( \{v_i\} \)
9: \hspace{2cm} end if
10: \hspace{1cm} end for
11: end if
12: for \( v_j \in T^{(st)} \) do
13: \hspace{1cm} if \( ct(v_j) > ct(rob_k) + t_{ij} \) then
14: \hspace{2cm} \( \Omega \leftarrow \Omega \cup v_j \).
15: \hspace{1cm} end if
16: end for
17: return \( \Omega \).

makespan. Thus, it is important to design a mechanism that allocates more robots to the urgent tasks. We maintain an archive \( A \) of historical information about previous robot executions to identify urgent tasks. The archive is initially empty, and is updated by adding each robot execution period \( ep \) during its execution process. The identification of the urgent task is started after the size of the archive reaches \( \phi \), when we consider enough information has been gathered for the identification. Specifically, for each uncompleted task in \( T^{(t)}_c \), if its estimation execution period is larger than \( \omega \) times the maximal execution period \( ep^* \) in the archive, this task is considered as an urgent task, seen Section IV for sensitive analysis of parameters. Once an urgent task is detected, the filter scheme returns only the detected urgent task to force all the robots to execute the urgent task. The estimation execution period is calculated according to Eqs. (7)-(9) and (14).

- If there is no urgent task, the filter scheme removes \( v_j \) from the candidate tasks of \( rob_k \) if \( v_j \) will be completed by the current robots before \( rob_k \) arrives at \( v_j \) (lines 12-16 in Algorithm 3).

Then, during the decision making process, every active robot calculates the priorities of the candidate tasks in the set \( \Omega \) according to the given RCS \( h(\cdot) \), and selects the task with the highest priority to execute next.

It should be noticed that, in some cases, multiple robots will be active at the same time. The generated execution plan can be affected by the order of the decisions made for the active robots. In our proposed algorithm, we make the decision for the robot with the smallest ability first.

3) Execution stage: In this stage, the robots will perform the executions according to the decisions made in the previous decision making stage. The states of the robots including their current positions and predicted events, and the states of the tasks including their current demands, current changing rates, and predicted completion time will be updated (shown in Algorithm 2).

In the execution stage, the predicted events of all the active robots are updated first. Then, the designed RCS heuristic template runs until one task has been completed. The earliest predicted event \( event^*_e \) of all robots will be selected as the next event, and the simulation time \( t \) goes to the time of the next event. Then, two situations are distinguished.

The first situation is that the earliest event \( event^*_e \) is an arrival event (i.e. \( Ct = 1 \)), which represents \( rob^*_k \) beginning to execute the task \( v_e \). In this situation, the predicted event type of the corresponding robot \( rob_k \) is updated as the departure event (\( Ct = 0 \)) representing that the robot is going to complete and depart from the corresponding task. To estimate the occurrence time of the predicted event, there are three rules to follow.

1) If \( rob^*_k \) departs from its current executing task \( v_e \) immediately when \( v_e \) has been completed by other robots.

2) If the sum of abilities of robots executing the \( v_e \) is not greater than the inherent increment rate of \( v_e \), representing that \( v_e \) cannot be completed, the occurrence time of the updated predicted event is \( \infty \).

3) Otherwise, if the sum of abilities of robots executing the \( v_e \) is greater than the inherent increment rate of \( v_e \), \( v_e \) can be completed in a finite time according to the current execution plan. The time of the updated predicted event of all the robots executing \( v_e \) is updated according to (7).

The second situation is that the earliest event \( event^*_e \) is a departure event (i.e. \( Ct = 1 \)), which represents that \( rob^*_k \) has completed its current task \( v_e \) and is preparing to visit its next task. In some cases of this situation, multiple robots will simultaneously complete one task. These robots will be selected as a set of active robots together to participate in the next round’s decision making stage. Meanwhile, the execution period \( ep_e \) of \( v_e \) is calculated and pushed into the archive \( A \) for recording the execution periods of completed tasks.

To better understand the heuristic template, an example is shown in Fig. 5. First, at time \( t = 0 \), all the robots are active, and they calculate priorities of candidate tasks according to the given RCS. The highest priority task of each active robot is added to the execution sequence \( \Pi \). The execution sequence \( \Pi \) of all robots is \{\([rob_1, v_1], [rob_2, v_2], [rob_3, v_3]\)\}, and the predicted events of the robots are \([rob_1, v_1, 3, 1] \), \([rob_2, v_2, 2, 1]\), and \([rob_3, v_2, 2, 1]\). With the current execution sequence \( \Pi \), \( rob_2 \) and \( rob_3 \) will execute \( v_2 \) collaboratively. The earliest predicted event \( [rob_2, v_2, 2, 1] \) of \( rob_2 \) is selected as the next event. The designed RCS heuristic template advances to the time \( t = 2 \), and status of \( v_2 \) and \( rob_2 \) are updated according to the departure output part in Algorithms 1 and 2. Then, \( [rob_3, v_2, 2, 1] \) and \( [rob_1, v_1, 3, 1] \) are selected as the next event in order, and the status of corresponding tasks and robots are updated. When the earliest event is the departure event of \( rob_2 \), the designed RCS heuristic template advances to the time \( t = 6 \). At the time \( t = 6 \), new tasks \( v_5 \) and \( v_6 \) are detected, and \( rob_2 \) and \( rob_3 \) are selected as the active robots to make decisions. The whole heuristic template repeats the aforementioned process until all the tasks have been completed.

C. Evolutionary Process

1) Framework: The overall framework of GPHH for the DMPDA problem is shown in Fig. 6. It has two processes, a
training process and a testing process. The GPHH algorithm generates/trains the RCS using a set of training instances, and then tests the best RCS on a set of unseen test instances. First, a population containing a set of individuals (each representing a RCS [50], [51]) is randomly initialized by the ramp-half-and-half method [38]. In each generation of the training process, the training instances are sampled and the individuals are evaluated according to the training instances and the simulator. After evaluating all the individuals, a new population is generated by the selection, crossover, mutation, and reproduction operators. For generating new individuals, one or two individuals are first selected to be the parents by the tournament selection. Then, the tree-based crossover, one or two individuals are first selected to be the parents and reproduction operators. For generating new individuals, the evolutionary process continues until the stopping criteria are met, and the best individual of the final population is returned as the final RCS $h(\cdot)$.

2) Mini-batch training process: A mini-batch training process is adopted in this paper to improve the generalization [24], [51]. The training instances $\Xi_{train}$ are split into 20 mini-batches $\Xi_{train}^{gen}$. After the splitting process, the mini-batches instances $\Xi_{train}^{gen}$ are used in the GPHH algorithm in different generations (e.g. mini-batch 1 in generation 1, 21, 41.).

3) Fitness Evaluation: The pseudo-code of the fitness evaluation of RCS $h(\cdot)$ on the subset of training instance $\Xi_{train}^{gen}$ is described in the Algorithm 4. The evaluated RCS will be applied to each instance of $\Xi_{train}^{gen}$ in Algorithm 1, and the objective value are recorded in a set of objective values $\mathcal{O}$. Then the average value of the makespans in $\mathcal{O}$ is calculated as the fitness of $h(\cdot)$.

4) Genetic Operators: The proposed GPHH uses the subtree crossover [38], in which the subtrees of two parents are randomly sampled and swapped to generate new children for the next generation. The traditional subtree-based mutation operator [38] is also employed in the proposed GPHH. In the mutation operator, a randomly sampled subtree of the mutated individual is replaced by a new randomly generated subtree. The reproduction operator selects an individual from the population by tournament selection, and directly copies it into the next generation [38].

5) Testing RCS performance: After a final RCS $h(\cdot)^*$ is obtained, its performance will be evaluated on a set of unseen test instances $\Xi_{test}$. The testing fitness of a RCS $h(\cdot)$ is defined as $f_{TE}(h(\cdot)) = \text{Evaluate}(\Xi_{test}, h(\cdot))$ (see in Algorithm 4).

IV. DESIGN OF EXPERIMENTS

A. Data Sets

Previous studies on the MPDA only focus on the static scenario in which all the tasks are known in advance. There is no existing benchmark set that contains dynamic scenarios in the DMPDA problem. To facilitate the study of the DMPDA problem, we design a new DMPDA benchmark set with a wide range of instances. The new benchmark set contains instances that are different in the number of robots and tasks, the abilities of robots, the inherent increment rates of tasks, and the detection time of tasks. The location of the depot is set as the centre of the workspace. The task locations are generated randomly in a 100*100 2D plane. The number of robots is chosen from $\{10, 15, 20\}$. The number of tasks is chosen from $\{200, 300, 500\}$. Following the conventional settings in [2], [3], the abilities of robots are sampled from the two normal distributions. The first distribution $N(0.035, 0.035/2)$ has a large variance and the second distribution $N(0.035, 0.035/10)$ has a small variance. Similarly for the mean inherent increment rates $\gamma_i$ of tasks, they are generated uniformly within an interval, which is chosen from $\{0.01, 0.02, 0.05, 0.08\}$. Then, the inherent increment rates of the tasks are sampled according to the normal distribution (i.e. $\gamma_i \sim N(\overline{\gamma_i}, \overline{\gamma_i}/10)$) following the conventional settings in [25], [51]. For all the tasks, their initial
demands are set to a relatively large value 2.30 to make the problem complicated enough to evaluate the approaches.

The detection time is also a key characteristic for the DMPDA problem. If the inter-detection time (the gap between the detection times of adjacent tasks) is very small, the abilities of robots are not enough to effectively handle all the tasks. If the inter-detection time is too large, the robots will always be idle so that the scenario is not challenging. To design an appropriate inter-detection time, we consider the travel cost, abilities of robots, the number of robots, and the increment rate of tasks. In the new benchmark set, there are ten initial tasks with detection time of 0. The inter-detection time of the other tasks follows the exponential distribution, and its rate parameter $\lambda$ is calculated as

$$\lambda = \left(\frac{\eta_0 + \tau \times \overline{\tau}}{\sum_{k=1}^{M} \mu_k - \overline{\tau}} + \tau\right) \times \rho \quad (15)$$

where $\tau$ and $\overline{\tau}$ represent the average value of travel cost among the tasks and inherent increment rates of tasks respectively, and $\rho$ means that the number of 1/$\rho$ tasks will be detected during the average period of all robots completing a new detected task together.

Considering these factors, 45 representative scenarios are designed in the benchmark set. The designed scenarios are named based on their number of robots, number of tasks, distribution of the abilities of robots, and distribution of the inherent incremental rates of tasks. For example, r10t200_L_S represents that there are 10 robots whose abilities are sampled from the distribution with a large variance ($N(0.035, 0.035/2)$) and 200 tasks whose inherent incremental rates are generated from a small uniform distribution ([0.01, 0.02]). The details of these scenarios are shown in Table SI of the supplemental document. Then, for each DMPDA scenario, 150 sampled instances are generated by sampling the random variables independently using different random seeds. After that, the 150 sampled instances are divided into 100 training instances and 50 test instances. For each DMPDA scenario, RCSs are trained on the 100 training instances, and then the final obtained RCS is tested on the 50 test instances. The 100 training instances are further split into 20 mini-batches, each containing 5 instances.

### B. Parameter Settings

The GPHH algorithm for learning RCSs is implemented based on a C++ GP research system named GPC++ [52]. Specifically, the terminal set is given in Table II. The parameter settings of the GPHH algorithm used in the rest of this paper follow the conventional settings [24], [28], [46]. The population size is set to 1000, 50 generations are given as the training stopping criterion. The crossover rate is 0.8, the mutation rate is 0.15, and the reproduction rate is 0.05. The maximal depth of a GP tree is 8. The size of the tournament selection is set to 7. For each scenario, the GPHH algorithm was run 20 times independently and their results compared using the Wilcoxon rank-sum test with a significance level of 0.05. All experiments are conducted on computers with Core(TM) i7-6700 3.40GHz CPU.

### Table II

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC</td>
<td>travel cost from the current position to the candidate task</td>
</tr>
<tr>
<td>CD</td>
<td>current demand of the candidate task</td>
</tr>
<tr>
<td>DAM</td>
<td>demand of the candidate task at the arrival moment</td>
</tr>
<tr>
<td>CR</td>
<td>current changing rate of the candidate task</td>
</tr>
<tr>
<td>FRT</td>
<td>fraction of the remaining (uncompleted) tasks</td>
</tr>
<tr>
<td>FUT</td>
<td>fraction of the unassigned tasks</td>
</tr>
<tr>
<td>AB</td>
<td>ability of the robot</td>
</tr>
<tr>
<td>ICR</td>
<td>inherent increment rate of the tasks</td>
</tr>
<tr>
<td>NRT</td>
<td>number of the robots assigned to the candidate task</td>
</tr>
<tr>
<td>HAB</td>
<td>total ability of robots assigned to the candidate task</td>
</tr>
<tr>
<td>HRT</td>
<td>predicted changing rate of the candidate task</td>
</tr>
<tr>
<td>TDT</td>
<td>detection time of the candidate task</td>
</tr>
<tr>
<td>LCT</td>
<td>last start time of robots executing the candidate task</td>
</tr>
<tr>
<td>FEP</td>
<td>first start time of robots executing the candidate task</td>
</tr>
<tr>
<td>RCMP</td>
<td>the reduction of the completion time of the candidate task (with/out a robot executing it)</td>
</tr>
</tbody>
</table>

### Table III

MEAN AND STANDARD DEVIATION OF THE 20 RUNS OF THE GPHH WITH DIFFERENT VALUES OF THE PARAMETER $\phi$ AND $\omega$.

<table>
<thead>
<tr>
<th>((\phi), (\omega))</th>
<th>r10t300_L_S</th>
<th>r15t500_L_M</th>
<th>r20t300_S_L</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5, 10)</td>
<td>4.06E+4</td>
<td>3.94E+4</td>
<td>2.47E+4</td>
</tr>
<tr>
<td>(5, 20)</td>
<td>4.06E+4</td>
<td>3.94E+4</td>
<td>2.47E+4</td>
</tr>
<tr>
<td>(10, 10)</td>
<td>4.06E+4</td>
<td>3.94E+4</td>
<td>2.47E+4</td>
</tr>
<tr>
<td>(10, 20)</td>
<td>4.06E+4</td>
<td>3.94E+4</td>
<td>2.47E+4</td>
</tr>
<tr>
<td>(10, 100)</td>
<td>4.06E+4</td>
<td>3.94E+4</td>
<td>2.47E+4</td>
</tr>
<tr>
<td>(10, 1000)</td>
<td>4.06E+4</td>
<td>3.94E+4</td>
<td>2.47E+4</td>
</tr>
</tbody>
</table>

* represents that the obtained RCSs by the GPHH method cannot generate feasible solutions in all of the tests for all 20 runs.

The size of the archive for recording the historical execution periods and the urgent threshold $\omega$ in the heuristic template are two new parameters for the proposed GPHH algorithms. Theoretically, if they are set to very small values, the algorithm will focus on gathering robots together to execute each task one by one. If their values are large, the algorithm has a poor ability to deal with urgent tasks. Hence, to investigate the influences of these parameters on the algorithm performance, the following experiment about the parameter sensitivity analysis is carried out. Here, three scenarios with different scales and characteristics, r10t300_L_S, r15t500_L_M, and r20t300_S_L, are used to find how these parameters affect the performance. Four candidate values \(\{5, 10, 20, 100\}\) are set for $\phi$, and four candidate values \(\{10, 20, 100, 1000\}\) are set for $\omega$. The other settings are kept unchanged. Each parameter setting on each instance is run 20 times independently. The mean and standard deviation of the objective values with the 20 independent runs are shown...
in Table III. The calculation method of the objective value is based on Eq. (11).

From the table, it can be found that:

- The performance of GPHH tends to be worse with the increase of $\phi$. In the designed urgent task handling mechanism, the opportunity of triggering the handling mechanism decreases with the increase of $\phi$. The reason is that a large size of the archive $\phi$ leads to a late time for triggering the handling mechanism and creates a large threshold, which is used to distinguish the urgent tasks and normal tasks. Thus, a large $\phi$ is not suitable for the proposed GPHH.

- The performance of GPHH improves with the increase of $\omega$ from 10 to 20. However, when $\omega$ increases to 100 and 1000, the performance becomes worse. When $\omega$ is a very small value, the threshold used for distinguishing the urgent and normal tasks is so small that majority of tasks are classified into urgent tasks. Robots are forced to gather together to execute the majority of tasks. When $\omega$ is very large, the threshold is so large that urgent tasks are difficult to identify. The accumulated demands of the urgent tasks lead to the poor performance of the whole multi-robot system.

Besides, when $\phi$ and $\omega$ are too small or too large, some RCSs evolved by the GPHH method cannot generate feasible solutions for all tested instances in the 20 runs. Hence, it was found that when the increment rates of tasks are small, the performance of GPHH is not sensitive to the parameter of $\phi$ and $\omega$. In summary, $\phi = 20$ and $\omega = 20$, which maintain a good performance for different scenarios, are adopted.

C. Comparison Design

Due to the dynamic process of discovering tasks in the DMPDA problem, existing meta-heuristic methods cannot be applied directly. Thus, we take the following seven manually designed RCSs as benchmark comparisons in this paper. First, four RCSs are manually designed to cater to the main characteristics of the DMPDA problem including travel cost, inherent increment rate, demand, and robot abilities.

- NNT: select the task with the nearest distance.
- MaxR: select the task with the maximum inherent increment rate.
- MinD: select the task with the minimum current demand.
- aveAbi: select the task with the maximal ICR-TAB.

Then, three state-of-the-art manually designed RCSs for the static MPDA problem are utilized: auction [22], Dynamic Deployment (DD) [23], and auction [23] are transferred to the DMPDA problem for benchmark comparisons.

V. RESULTS AND DISCUSSIONS

A. Test Performance of the evolved RCS

The computational results of the seven methods are shown in Table IV. For every compared method, $(+)/(\approx)/(-)$ represents that it is significantly better than, statistically comparable with, and significantly worse than the proposed GPHH method based on the Wilcoxon rank-sum test with a significance level of 0.05 and Bonferroni correction. Meanwhile, the shadow grid in GPHH indicates that GPHH significantly outperformed all the other compared algorithms. An additional row at the bottom of Table IV represents the number of instances on which the corresponding compared algorithm achieves significantly better, statistically comparable, and significantly worse results than the proposed GPHH method.

First, from Table IV, we can conclude that the proposed GPHH method apparently outperformed all the compared algorithms in all the test scenarios. There are various factors related to both robots and tasks in the complex DMPDA problem that affect the performance of the RCS. Hence, it is challenging to manually design effective RCS for a given complex DMPDA problem scenario. In contrast, since GPHH has a good ability to evolve sophisticated and flexible rules, the proposed GPHH algorithm can effectively learn the useful knowledge of the DMPDA problem that is hard to be identified by human from the historical data. Thus, GPHH can automatically design RCSs that are suitable for different scenarios with different characteristics.

Second, the statistical results are analysed in terms of the different scenarios. When the number of tasks $N$ and robots $M$ are the same, the performance of GPHH in the scenarios in which the abilities of the robots are sampled from the small variance distribution always obtained smaller objective values than the situations with large variance distribution. The performance gap between GPHH and the other manually designed algorithms in the scenarios with small inherent increment rates is smaller than the performance gap on the scenarios with large inherent increment rates. The reason is that the tasks with small increase rates usually will not reach very large and uncontrolled demands. However, the tasks with large inherent increment rates easily reach very large demands if they are not executed in time. Several tasks with very large demands lead to a very large makespan. Thus, the impact of execution plans in scenarios with small increase rates is smaller than the impact of execution plans in scenarios with medium and large increase rates.

Thirdly, the statistical results are analysed in terms of the different algorithms. For the auction and DD algorithms [23], they obtain good performances on the scenarios with small increment rates (e.g. $r10t200$). However, they cannot generate feasible solutions in the tested process for all 20 runs on scenarios with large increment rates (e.g. $r10t200$). The reason is that the auction and DD algorithms try to ensure each task can be completed sequentially. However, in some scenarios, the total task increment rate is much greater than total robot ability so that a task cannot be completed at one decision making stage. In contrast, the recruitment algorithm [22] ensures that at least one task can be completed at each decision making stage. Thus, the recruitment algorithm can always generate a feasible solution. From the table, it can also be found that the manually designed RCSs that gather robots together to execute tasks are suitable for the scenarios with large increment rates. On the contrary, the manually designed RCSs that scatter robots are suitable for the scenarios with small increment rates.

Overall, it can be concluded that GPHH is an effective...
algorithm to address the DMPDA problem.

B. Training Curve

Fig. 7 shows the convergence curves of the objective values of GPHH in the training process on four representative scenarios. The x-axis represents the generation. Due to the huge gap of different objective values among different solutions, the y-axis stands for the natural logarithm of the average objective value of the best solution fitness over 20 runs in each generation. From Fig. 7, it can be observed that objective values obtained by the proposed GPHH method gradually decrease during the training process. In addition, the convergence curves in the figure fluctuates due to the rotation of the mini-batches instances in different generations. However, if we compare the performance of two generations using the same mini-batches instance (e.g., generations 10 and 30), the performance in the latter generation is improved.

C. Training Time

Fig. 8 shows the training time of the proposed GPHH method across different numbers of tasks according to the results of 18 different representative scenarios with different characteristics. The x-axis represents the number of tasks and the y-axis stands for the average training time of 20 runs for each scenario measured by seconds. From Fig. 8, the following observations can be obtained.

- Comparing Fig. 8 (a) and Fig. 8 (b), we can find that the training time of the proposed GPHH is not sensitive to the number of robots and the variance of robot abilities.
- The training time of the proposed GPHH increases as the inherent increment rates of tasks increase. The reason is
that the collaboration behaviours in the scenarios with large inherent increment rates are more intense than the scenarios with small inherent increment rates. Thus, there are more events in the entire execution plan in the scenarios with large inherent increment rates, leading to a larger training time.

- The training time of the proposed GPHH increases by the square of the number of tasks. The reason is that the greater the number of tasks leads to the greater times of calculating priorities in the RCS execution simulation process.

VI. FURTHER ANALYSIS

The comparison results show that the RCSs generated by the proposed GPHH algorithm can achieve good performances. In this section, the RCSs generated by GPHH will be further investigated to determine how they affect the DMPDA execution process.

A. Effectiveness of the designed filter

To verify the effectiveness of the designed cluster filter, we compare it with the GP without filter (GPHH-WF) and with a baseline filter in which only the priorities of the top ten tasks with minimal traveling cost will be calculated by a given GP tree [45]. We use the r10t200_S_M, r15t500_L_S, r15t500_S_L, and r20t500_S_M with different robot abilities and scales as the test scenarios. Fig. 9 shows the results obtained by GPHH, GPHH-WF, and GPHH-DF. From Fig. 9, we can see that GPHH, GPHH-WF, and GPHH-DF obtained obviously different distributions of the makespan. Especially, on the three scenarios (r10t200_S_M, r15t500_S_L, and r20t500_S_M) where the inherent increment rates of the tasks are medium or large, the final results obtained by GPHH-WF and GPHH-DF have apparent outliers. GPHH has clear superiority against GPHH-WF and GPHH-DF in terms of both consistency and effectiveness.

Some tasks with medium or large inherent increment rates are easy to grow to become urgent tasks. If there are several urgent tasks, the time to complete these tasks in turn will increase exponentially. In the designed filter, these urgent tasks can be identified during the designed RCS heuristic template, and robots are allocated to execute these tasks when their demands are not very large. Therefore, the designed filter is very useful.

B. Effectiveness of the decision order

Decision order is a key issue in the multi-robot system since different decision orders can cause different decision qualities. Thus, computational experiments are performed to validate the influences of decision orders on the proposed GPHH algorithm. In the experiments, a GPHH algorithm in which robots with large abilities make decisions first, a GPHH algorithm in which robots make decisions in a random order, and a GPHH algorithm in which robots make decisions according to a pre-determined order are designed as the control group, denoted as GPHH-LF, GPHH-R, and GPHH-P respectively. The computational results of the GPHH methods with different decision orders are shown in Table V, and (+)/(-)/(-) represents the method is better/equal/worse than GPHH. An additional row at the bottom of this table represents the number of instances on which the compared algorithm has achieved a better result than the GPHH methods under the Wilcoxon rank-sum test with Bonferroni correction.
First, the Friedman test is carried out to detect differences between the GPHH and compared algorithms across seven scenarios. The p-value of the Friedman test show that there are not significant differences on r10t500_S_S, r15t200_S_M, and r20r500_S_S. From Table V, we can also see that the results are insensitive to the decision order on the three scenarios. The reason may be that the automated designed RCSs always gather robots together to execute tasks one by one. Compared with GPHH-LF, GPHH obtains better performance except for r15t200_S_M.

Compared with GPHH-R and GPHH-P, GPHH is significantly better on r10t500_L_M and r15t300_L_L in which the abilities of robots have large variance. We can also find that the decision order in scenarios with large robot ability variance is more sensitive than the decision order in scenarios with small robot ability variance. In summary, the designed decision order in which robots with small abilities make decisions first is more effective than the other strategies.

C. Feature Analysis

Fig. 10 shows the number of times of the features appearing in the best evolved RCSs in the 20 runs in four different scenarios, respectively. The x-axis represents the features and the y-axis indicates the number of appearances of the corresponding features in the final best performance RCSs. From Fig. 10, the following observations can be obtained:

- TC and DAM are the most frequently used features among the 14 features, representing that the travelling cost and the predicted demand at the arrival moment are important for the DMPDA problem.
- TD and FEP are the most infrequently used features among the 14 features, representing that the detection time of the candidate task and the first start time of robots executing the candidate task may not be the key features for the DMPDA problem.
- RCMP is more frequently used for the scenarios with a small variance of robot abilities and does not make a significant difference in the scenarios with small inherent increment rates.
- NRT is more frequently used in scenarios with a large number of robots, indicating that the number of the robots assigned to the candidate task is important for the task planning of a large number of robots.
- ITAB is frequently used for the scenarios with small inherent increment rates. This indicates that balancing the abilities of robots is important for the scenarios with small inherent increment rates.

D. RCS semantic analysis

To gain a deeper understanding about the GP-evolved RCSs, we take the best RCS achieved by the proposed GPHH algorithm as an example since r15t500_S_L is the most difficult to be optimised than other three example scenarios. After the semantic simplification, e.g. \( a - a = 0, a/a = 1 \), this RCS is shown in (16).

\[
h(\cdot) = \text{TAB} - 2\text{DAM} + 2\text{RCMP} + \text{CD}(\text{TC} - 1) - \text{TC} \quad (16)
\]

The example RCS is quite small and easy to understand after the simplification. From (16), the following observations can be obtained:

- The RCS prefers the selection of a candidate task if a large total ability of robots have been assigned to it (i.e. TAB), and the selection can lead to a large reduction of the completion time (i.e. RCMP). It suggests that the robots should complete some tasks as soon as possible, so that the future demand increment can be reduced.
- Since the travel cost (i.e. TC) is always larger than one, the candidate task with a large current demand (i.e. CD) is preferred to be executed first. The reason is that a task with a large CD can be regarded as a urgent task, which needs to be executed as soon as possible.
- To reduce the travel cost of the robots, the candidate task with a small travel cost (i.e. TC) is preferred to be executed first.
- TAB and TC are relatively small compared with RCMP, DAM, and CD as the execution process goes. Thus, DAM, RCMP, and CD play a major role in the decision making stage.

VII. CONCLUSIONS

In summary, this is the first paper to investigate the DMPDA problem. Considering the three characteristics of the problem, i.e. time-varying demands, dynamic arrival/detection time, and collaborative behaviour among robots, a new model is proposed in this paper. To address the DMPDA problem, a new GPHH method is developed in this paper. The new GPHH algorithm contains a new problem-specific heuristic template for fitness evaluation. Also, a new terminal set is designed considering the information that can be obtained during the mission execution. The heuristic template also features a novel filter scheme to handle the urgent tasks. The experimental results showed that the proposed GPHH method significantly outperforms the state-of-the-art manually designed rules in solving the DMPDA problem. Also, the proposed GPHH method is very flexible, so it can be easily transferred to other...
real-world applications (e.g. search and rescue). The further analysis and discussions showed when and how to distribute and group robots to execute different types of tasks efficiently in the DMPDA problem. For future studies, there are still some problems that need to be further investigated. This paper has shown that the GPHH-evolved RCS can provide favourable decisions during the decision making stage. However, the decision making stage in this paper is based on the full-connected and no-delay network environment, which is hard to implement in practice. It would be very useful to investigate the performances of the decision making in different network environments, such as communication delay networks. Besides, the DMPDA problem is a naturally multi-objective problem which needs to balance the number of robots and the efficiency of executing all the tasks. It would be important to design a multi-objective method to complete the tasks as soon as possible balanced with a minimised number of robots. Finally, even though the proposed DMPDA model has considered the dynamic task arrivals, the uncertainty characteristics (i.e. travel cost) which exists in real applications are not considered in this paper. It would be worthwhile paying attention to extend the DMPDA model with these uncertainty characteristics.

REFERENCES


