Deep Reinforcement Learning Assisted Genetic Programming Ensemble Hyper-Heuristics for Dynamic Scheduling of Container Port Trucks

Xinan Chen, *Student Member, IEEE,* Ruibin Bai, *Senior Member, IEEE,* Rong Qu, *Senior Member, IEEE,* Jing Dong, and Yaochu Jin, *Fellow, IEEE*

***Abstract*—Efficient truck dispatching is crucial for optimizing container terminal operations within dynamic and complex scenarios. Despite good progress being made recently with more advanced uncertainty-handling techniques, existing approaches still have generalization issues and require considerable expertise and manual interventions in algorithm design. In this work, we present deep reinforcement learning-assisted genetic programming hyper-heuristics (DRL-GPHH) and their ensemble variant (DRL-GPEHH). These frameworks utilize a reinforcement learning agent to orchestrate a set of auto-generated genetic programming (GP) low-level heuristics, leveraging the collective intelligence, ensuring advanced robustness and an increased level of automation of the algorithm development. DRL-GPEHH, notably, excels through its concurrent integration of a GP heuristic ensemble, achieving enhanced adaptability and performance in complex, dynamic optimization tasks. This method effectively navigates traditional convergence issues of deep reinforcement learning (DRL) in sparse reward and vast action spaces, while avoiding the reliance on expert-designed heuristics. It also addresses the inadequate performance of the single GP individual in varying and complex environments and preserves the inherent interpretability of the GP approach. Evaluations across various real port operational instances highlight the adaptability and efficacy of our frameworks. Essentially, innovations in DRL-GPHH and DRL-GPEHH reveal the synergistic potential of reinforcement learning and GP in dynamic truck dispatching, yielding transformative impacts on algorithm design and significantly advancing solutions to complex real-world optimization problems.**

***Index Terms*—automatic truck dispatching, dynamic task scheduling, genetic programming, reinforcement learning**

I. Introduction

MARITIME container shipping constitutes a cornerstone of global trade, handling over 90% of international commerce. The rapid escalation of containerized trade—from 102 million metric tons in 1980 to approximately 1.83 billion metric tons in 2017—has compelled stakeholders to adopt new strategies for efficiency [1]. One prevalent strategy has been to augment the capacity of container ships, resulting in a deadweight tonnage increase from 11 million metric tons in 1980 to 275 million metric tons by 2020 [2]. While such ships have bolstered container throughput, they have also prolonged vessel turnaround times, thereby attenuating port efficiency. Though ports can theoretically mitigate this impact by expanding their infrastructure, deep-water berths suitable for larger vessels remain scarce and highly strategic resources [3]. Consequently, optimizing berth loading and unloading processes has become a priority for maintaining port efficiency.

Therefore, the Ningbo-Zhoushan Port company has partially implemented and deployed a dynamic container truck dispatching algorithm based on a manually devised heuristic [4] to improve operational efficiency over the past few years. Unfortunately, this traditional expert heuristic consumes a significant amount of time and labor to build and adjust and can still not cope with diverse scenarios. Thus, auto-generated heuristics based on real-life data were proposed to adapt to complex scenarios in real-world operating environments. Genetic programming (GP) and reinforcement learning (RL) are considered capable of automatically learning parameter settings and dispatching methods for different scenarios based on historical data, thus achieving better scenario adaptability. However, it is disappointing to observe the poor performance of these two algorithms when directly applied to the dynamic container port truck dispatching problem [5].

Consequently, we pivoted to a hyper-heuristics framework, distinguished by its enhanced robustness and generalization capabilities [6]. Based on the proven efficacy of deep reinforcement learning-based hyper-heuristics (DRL-HH) [7] and genetic programming hyper-heuristics (GPHH) [8] in simplistic training environments, we aim to extrapolate this effectiveness to unseen, more complex real-world testing environments. Considering the port's need for interpretability of dispatch methods, we introduced a framework that employs high-level heuristics to select explainable low-level GP heuristics training in different environments. This approach aggregates the intelligence of multiple GP heuristics, thereby dramatically improving the performance of DRL-HH and GPHH in complex real-world environments while eliminating

This work is supported by the National Natural Science Foundation of China (Grant No.72071116) and Ningbo Municipal Bureau of Science and Technology (Grant No. 2023Z237). (Corresponding author: Ruibin Bai.)

Xinan Chen is with the International Business School Suzhou, Xi'an Jiaotong-Liverpool University, Suzhou, 215123, China (email: xinanchen.pt@xjtlu.edu.cn).

Ruibin Bai is with the Digital Port Technologies Lab, School of Computer Science, University of Nottingham Ningbo China, Ningbo 315100, China (email: ruibin.bai@nottingham.edu.cn).

Rong Qu is with the School of Computer Science, University of Nottingham, Nottingham NG72RD, UK (email: rong.qu@nottingham.ac.uk).

Jing Dong is with the Department of Engineering, University of Cambridge, Cambridge CB21TN, UK (email: jd704@cam.ac.uk).

Yaochu Jin is with the School of Engineering, Westlake University, Hangzhou 310030, China (email: jinyaochu@westlake.edu.cn).

the dependence on expert-designed heuristics.

In the realm of high-level heuristics in this framework, both GP and RL emerge as strong contenders due to their adaptability and data-driven capabilities. However, Chen et al.'s experiments demonstrated that using GP as a high-level heuristic to select low-level GP heuristics [9] yielded promising results in small, generated datasets but faltered with more extensive, real-world datasets. To overcome this limitation, we propose a reinforcement learning-assisted genetic programming hyper-heuristic (DRL-GPHH). This new approach incorporates a set of GP-generated low-level heuristics and employs a reinforcement learning agent to act as the selector among these heuristics for various scenarios. Comparative evaluations indicate that DRL-GPHH outperforms its counterparts in simulated real-world conditions.

Nonetheless, it is noteworthy that the performance enhancement observed in DRL-GPHH was not remarkably superior to the results achieved by the best GP individual. This is because in DRL-GPHH, multiple GP evolved heuristics participate in the process, but during each decision point, only one GP individual is chosen to calculate the dispatching preference. Such an approach wastes the knowledge embedded in other unselected GP individuals, weakening the algorithm's performance and generalization capabilities. With this hypothesis, we propose a reinforcement learning-assisted genetic programming ensemble hyper-heuristics (DRL-GPEHH) method, which uses a reinforcement learning agent to assign different weights to different GP individuals during each dispatch and subsequently combines the results of all GP heuristics according to their weights to produce the final dispatching solution.

DRL-GPEHH, with its innovative integration of auto-evolved GP heuristics, demonstrates excellent proficiency in complex dynamic optimization tasks, as evidenced in container port truck dispatching. Its robust performance during both training and testing underscores its versatility for diverse real-world dynamic optimization scenarios. This method, distinct from traditional DRL, DRL-HH, or GPHH, represents a significant advancement in generative, automated optimization techniques, offering promising solutions for various dynamic scheduling challenges beyond container ports. The primary contributions of this paper are as follows:

- We propose DRL-GPHH, an approach that employs DRL to select GP-generated low-level heuristics. This approach effectively increases the adaptability of the algorithm, eliminating the dependence on expertly designed low-level heuristics inherent in DRL-HH, resulting in a much higher level of automation in algorithm design.
- We further propose a novel GP ensemble hyper-heuristics framework DRL-GPEHH, which leverages the knowledge of multiple auto-evolved GP heuristics simultaneously during scheduling, leading to significantly improved performance and robustness compared to DRL-HH and DRL-GPHH.
- We design a multi-action proximal policy optimization (PPO) agent with a novel reward to effectively adjust the weights of several lower-level heuristics within an action, promoting collaboration and performance improvement.
- We conduct comprehensive experiments and ablation studies on benchmark instances, covering diverse scenarios and terminal configurations, to fully test the performance of DRL-GPEHH in comparison with its main variants, DRL-HH, DRL-GPHH, and a deep reinforcement learning-assisted ensemble hyper-heuristics (DRL-EHH). The results demonstrate the superiority of DRL-GPEHH in the dynamic truck dispatching problem.
- We analyze the differences between manual heuristic and GP-generated heuristic ensemble methods and explain why a continuous weight adjustment is necessary for GP ensembles to achieve higher performance, further highlighting the compatibility of DRL-GPEHH with GPHH integration and establishing a novel approach for RL and GP cooperation.

The rest of this paper is organized as follows. Section II reviews related work and provides background information on the maritime container terminal truck dispatching problem. Section III introduces and formulates this dynamic truck dispatching problem. The proposed DRL-GPHH and DRL-GPEHH methods are delineated in Section IV. Section V delineates the experimental outcomes and provides ablation & sensitivity analysis of DRL-GPEHH. Finally, conclusions are drawn in Section VI.

II. Background

A. Dynamic truck dispatching in maritime container terminals

Fig. 1 illustrates a typical maritime container terminal with four major parts: the ship berths, quay cranes (QCs), yard blocks, and yard cranes (YCs). These components collaboratively transfer containers between the yard and the container ships via trucks. The QCs and YCs, positioned at the ship berth and the yard blocks, respectively, are primarily responsible for container operations. Both QCs and YCs can move left and right, enabling them to cover a wide range of operations, while they cannot cross each other. Moreover, each QC and YC can only operate containers on one truck at a time, inevitably leading to potential queuing and waiting at the operation nodes. The efficiency of the entire port operation is consequently influenced by the careful orchestration of these three key elements: the QCs, the YCs, and the trucks.

Research on YCs and QCs has demonstrated that optimizing their workflow can improve port efficiency to a certain extent [10]. However, numerous previous studies have overlooked the impact of truck availability on the scheduling of QCs and YCs. Inadequate truck support compromises the efficiency of these cranes [11]. Consequently, optimizing container truck dispatching has gained increasing attention as a critical issue for enhancing port efficiency.

Various techniques have been explored to address the challenge of container truck dispatching, such as mixed integer programming [12], min-max nonlinear integer programming [13], greedy algorithms [14], and genetic algorithms [15]. Many of these studies have reported favorable outcomes regarding reduced ship dock time, decreased empty-truck travel distance, or overall improvement in truck travel distance.

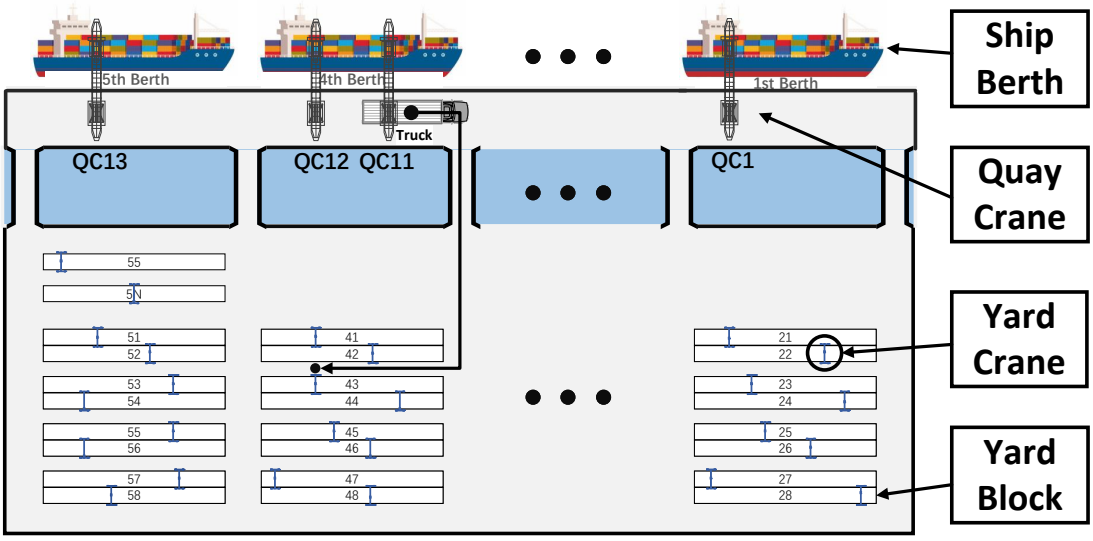

Fig. 1. Sample Map of Maritime Container Terminal at Ningbo Meishan Port

These findings indicate that optimizing container truck dispatching not only enhances local equipment efficiency in the berth or yard area but also benefits the overall efficiency of the port.

In actual container port operations, the cranes' operating time and container trucks' transit time are subject to uncertainty. This variability renders optimization solutions generated by offline algorithms, such as matheuristics, metaheuristics, and integer programming, often infeasible in real-world scenarios. Consequently, heuristic-based dynamic real-time dispatching methods are commonly employed in actual port operations [4]. Although heuristics do not guarantee optimal solutions [16], they can rapidly generate high-quality solutions. In dynamic dispatching systems, operators can also participate in the dispatching solution adjusting based on their experience. Due to their robustness, comprehensibility, and rapid response, heuristic-based dynamic dispatching methods are currently the preferred choice for real-life ports [17].

B. Hyper-heuristic

Hyper-heuristics are an innovative and burgeoning research paradigm in the field of metaheuristic optimization, which aims to advance the development of automated problem-solving methodologies. This paradigm encompasses high-level methodologies that intelligently select, combine, generate, or adapt low-level heuristics to solve a diverse array of complex combinatorial optimization problems [18]. By transcending the reliance on problem-specific knowledge and fine-tuning, hyper-heuristics boast enhanced generality and adaptability, thereby fostering a more robust optimization process to cope with larger scale optimization problems like job-shop scheduling [19], educational timetabling [20], packing[21], combinatorial optimization [22], among others.

The underlying assumption of hyper-heuristics is that the heuristic space is less problem-dependent than the solution space is. Thus, a more general search method can be developed by searching the solution space indirectly through the heuristic space. As depicted in Fig. 2, hyper-heuristics employs a two-tiered architecture, encompassing a high-level heuristic and an array of low-level heuristics that manipulate solutions. The high-level heuristic abstains from direct problem interaction; rather, it either adaptively selects from a predetermined repertoire of heuristics to address the given problem or learns to generate novel heuristics tailored to the problem at hand. The high-level heuristic typically harnesses domain knowledge and problem features amassed throughout the problem-solving process (either online, offline, or both) to facilitate the selection or generation of pertinent low-level heuristics.

In this study, we employed two types of hyper-heuristics. More specifically, GP serves as a hyper-heuristic that generates low-level heuristics, while reinforcement learning functions as a hyper-heuristic for selecting and combining GP individuals (i.e. low-level heuristics). Contrary to a standard hyper-heuristics approach, in which a high-level heuristic typically aims to identify a single low-level heuristic that aligns with the prominent features of the problem, we propose a heuristic ensemble framework that employs reinforcement learning to utilize multiple low-level heuristics that collaboratively ad-

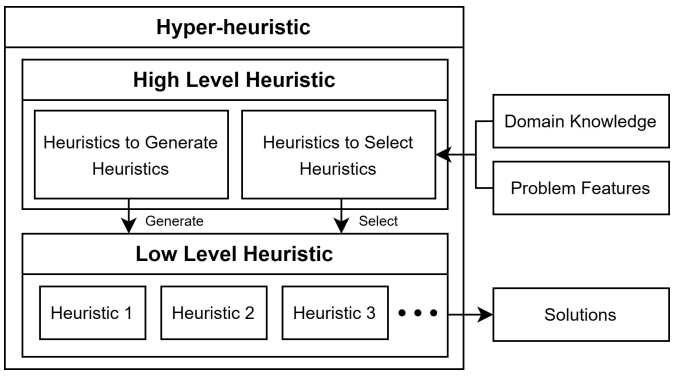

Fig. 2. Hyper-heuristic Framework

dress problems, representing a key advancement over the previous single heuristic methods. This approach enables us to tackle complex multi-scenario challenges characterized by abundant interferences, slow learning, difficult generalization, and suboptimal performance. The principles underlying this approach will be discussed in greater detail in the subsequent sections.

C. Genetic Programming

GP is an evolutionary computation technique that evolves a population of programs, often represented as GP trees, through selection, crossover, mutation, and replacement [23]. Over the years, GP has been successfully applied to various engineering and optimization problems, demonstrating its versatility and robustness in diverse contexts [24], [25].

Compared to other methods, such as decision trees, logistic regression, support vector machines, and artificial neural networks, GP offers three key advantages for solving optimization problems. First, GP, in essence, is a generative method. It allows for flexible representations, enabling the encoding of complex problem structures [26]. Second, its powerful search mechanisms facilitate the exploration of large and intricate solution spaces [27]. Finally, the heuristics generated by GP are partially interpretable and highly efficient in execution, thereby enhancing their practical value [28].

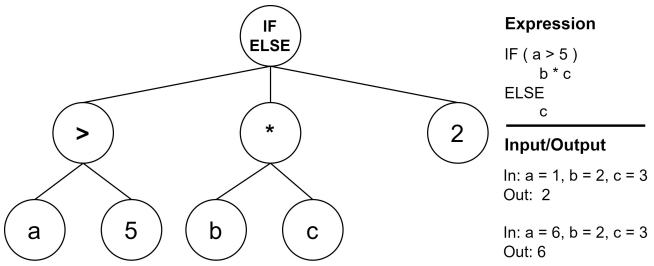

Fig. 3. Example Genetic Programming Tree Structure

In this study, we chose the widely adopted tree-based GP structure (example in Fig. 3) for several key reasons. The tree-based structure offers visualization and interpretability and can be easily expressed through mathematical expressions. It is also flexible, adaptable, and scalable, allowing it to accommodate a range of heuristics tailored to the problem at hand and handle problems of varying sizes and complexities. Furthermore, it is compatible with genetic operators like crossover and mutation, essential for the evolutionary process. With these advantages, tree-based GP is an ideal choice for generating low-level heuristics in our DRL-GPHH and DRL-GPEHH frameworks.

When employing tree-based GP to solve problems, as the example depicted in Fig. 3, GP generates a parameterized tree-structured expression. Depending on the input parameters across various environments, it outputs different solutions. In contrast to traditional evolutionary algorithms like genetic algorithms, which search directly within the solution space [6], tree-based GP operates at a higher dimension, seeking heuristics that produce solutions and can thus be categorized as Genetic Programming hyper-heuristics (GPHH). In comparison to manually designed heuristics requiring substantial expert inputs, GPHH provides a more flexible and adaptable approach to tackling complex tasks and can generate distinct heuristics for a wide range of problems without extensive domain-specific knowledge [29]. Due to its favorable adaptability and performance, the proposed DRL-GPEHH algorithm in this study is not solely confined to addressing port dispatching issues; it also possesses the potential to tackle numerous other complex optimization problems in real-world scenarios.

D. Ensemble Methods

Ensemble methods, also known as multi-expert models, have attracted significant attention in recent years due to their ability to address complex problems by combining the strengths of several individual models or heuristics. These models are based on the premise that integrating the outputs of multiple experts can result in a more robust and accurate decision-making process, as opposed to relying on a single expert or model.

Ensemble methods have been widely employed in machine learning and pattern recognition tasks, where several base classifiers are combined to improve classification accuracy [30]. These techniques aim to create diverse and complementary classifiers by manipulating the training data or learning algorithms. Research has shown that ensembles can significantly improve algorithm performance, particularly when each algorithm exhibits different strengths and weaknesses [31].

In optimization, ensemble methods have been employed to solve complex problems more effectively. For instance, the cooperative co-evolutionary algorithm [32] divides a problem into smaller subproblems, which individual experts solve. The solutions are then combined to form a global solution. Similarly, the evolutionary forest [33] and particle swarm optimization [34] apply multiple search strategies simultaneously, allowing the exploration of the solution space more effectively.

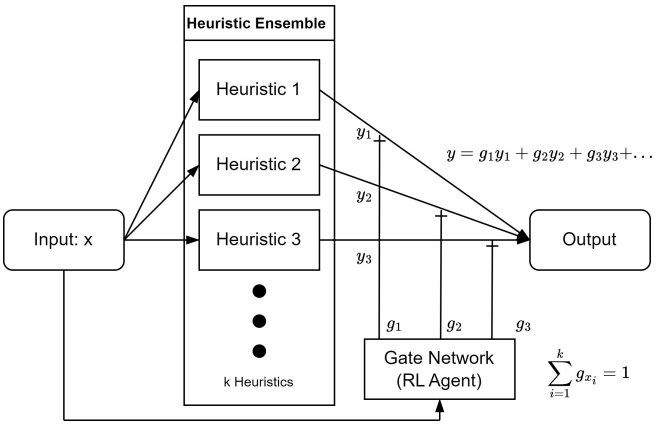

Fig. 4. Learning-Assisted Heuristic Ensemble Framework

Leveraging the benefits of both ensemble methods and hyper-heuristics models, ensemble-based hyper-heuristics demonstrate enhanced performance [35], [36]. Inspired by

their capabilities, this paper introduces a novel, learning-assisted heuristic ensemble model. This model proposes using reinforcement learning as a gate network for selecting/combining multiple low-level heuristics to address intricate multi-scenario problems. The learning-assisted ensemble structure in this paper is depicted in Fig. 4. The heuristics compete to generate output, while the gating network orchestrates this contest. For each input x, the gating network acquires information about the performance of all heuristics $(y_1, y_2, y_3, ...)$ involved in addressing the task, and the output of each heuristic is compared with the target output y. The gating weights of heuristics $(g_1, g_2, g_3, ...)$ are adjusted based on the relative performance of that heuristic, compared to the other heuristics, for the specific input pattern. Within this framework, each heuristic is required to solve the problem solely within its designated area of focus, thus mitigating the model's complexity and training costs while enhancing the robustness of the generated solutions.

We posit that the learning-assisted ensemble hyper-heuristics model can deliver enhanced performance characterized by rapid convergence and superior generalization capabilities. Employing a weighted sum in this model facilitates the effective integration of individual heuristics' strengths, culminating in a more robust and powerful decision-making process. Moreover, the tree-based GP and manual heuristics low-level heuristics in this model offer traceable decisions despite reduced comprehensibility from ensemble and weight adaptations. This transparency is a huge advantage over the opaque nature of deep learning methods. This model is examined and analyzed on a complex multi-scene port truck dispatching problem, which involves uncertainty, in Section V.

E. Reinforcement Learning

Reinforcement Learning is a concept initially introduced by Minsky in 1954 [37], characterized by the fundamental notion of learning through the administration of rewards and punishments. Following a hiatus, DeepMind introduced the concept of deep Q-learning (DQN) in 2013 [38], ushering in a new epoch for reinforcement learning. Since then, deep reinforcement learning has been employed in an extensive array of other domains, including operation research [39], resource management [40], and self-driving [41], among many others.

However, the innovations in port operations using the latest artificial intelligence (AI) technologies have been largely constrained to computer vision and its applications. The application of deep learning in dynamic port dispatching problems remains relatively rare. The primary reasons can be categorized as follows: 1) the infeasibility of training, as reinforcement learning algorithms are challenging to train in real-life port environments; 2) convergence difficulties, as utilizing simulators and historical data for training may prove arduous due to the abundance of states and actions, potentially impeding convergence; 3) sparse rewards, since the dynamic dispatching problem constitutes a sparse reward problem, it is difficult to compute the reward without completing a certain number of tasks, and traditional methods struggle to receive timely feedback after each action; 4) delays, as the real-time requirement for port dispatching is stringent, and large reinforcement learning models face difficulty in making decisions within a limited time frame [42].

Consequently, a deep reinforcement learning-based hyper-heuristics (DRL-HH) [7], which integrates reinforcement learning and heuristic algorithms, is proposed to tackle the challenges associated with reinforcement learning for the dynamic truck dispatching problem, yielding some success. However, employing parameter-derived variants of a single expert-designed heuristic for truck assignment exhibits inadequate generalization and restricted performance. DRL-GPHH and DRL-GPEHH, in conjunction with multiple GP-generated heuristics proposed in this study, more effectively address these concerns and further enhance the performance of reinforcement learning approaches in the dynamic truck dispatching problem in container terminals.

III. Problem description and formulation

This study aims to optimize the efficiency of port operations by developing an internal container truck dispatch algorithm that addresses the multi-constrained vehicle routing problem specific to port logistics. By ensuring an adequate supply of trucks for QCs and Yard Cranes (YCs), the algorithm aims to minimize idle and waiting times, thereby expediting the container handling process crucial to ship turnaround efficiency. Containers, arriving primarily in two standard sizes measured in twenty-foot equivalent units (TEUs), give rise to three types of dispatch tasks: a single large container, two bound small containers, and single small containers requiring dynamic pairing. While QCs can handle two small or one large container, YCs are limited to one container per operation. Load balancing requirements and stacking location constraints dictate a predetermined container handling order, with congestion typically arising from crane occupancy or delayed truck arrivals. The dispatch problem is thus modeled as a complex, multi-constrained vehicle routing problem involving queuing and waiting on a directed graph.

The problem can be formally delineated as follows. An abstract container terminal is depicted as a directed graph, denoted by $G = (A, C)$, where $C = Q \cup Y$ constitutes the nodes representing the work operation points for all tasks. The sets Q and Y encompass all QCs and YCs, respectively. The set A consists of direct driving connections between distinct nodes. The truck depot, represented by d, is the point from which all trucks depart at the commencement of the operation and return upon completion of all tasks. The set $V = v_1, v_2, v_3, \ldots, v_m$ signifies the collection of m available trucks for allocation. A function $\tau(x, y)$ maps two disparate operation points, $x \in C$ and $y \in C$, to the time required to traverse from one point to the other, reflecting the actual terminal road network. The work instruction list encompasses all n transport tasks in $T = t_1, t_2, t_3, \ldots, t_n$. The container size for each task t_i is denoted by z_i. The source and destination nodes for a given t_i are represented by a_i and b_i, respectively, with $a_i, b_i \in C$. Based on the diverse types of

source and destination nodes, y_i is defined as the type of task i. $y_i = 1$ signifies an unloading task, while $y_i = 0$ corresponds to a loading task.

Within our problem framework, tasks are confined to transportation journeys exclusively between QCs and YCs. Consequently, a_i and b_i pertain to distinct crane-type node sets, either QCs or YCs. The maximum difference in task serial numbers, denoted as q, indicates the acceptable swapping order of unloading tasks (in this paper, $q = 3$, considering the practicalities). The start time of service for t_i at its source node is represented by s_i, while its completion time at the destination node is symbolized by e_i, where $s_i \in S = \{s_1, s_2, s_3, \ldots, s_n\}$ and $e_i \in E = \{e_1, e_2, e_3, \ldots, e_n\}$. Since a crane is required to either load or unload the container at the beginning and end of a task, the parameters d_i and h_i depict the operating time of t_i at the source and destination nodes, respectively, and their sum is r_i. The operation times at QCs and YCs are assumed to be stochastic and extracted from historical data.

To model the problem formally, the assignments of tasks to trucks are defined by the following binary variable in (1):

$$\alpha_{ij} = \begin{cases} 1 & t_j \text{ is assigned to } v_i \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

The following auxiliary variable indicates whether t_k is serviced immediately after task t_j by truck v_i.

$$\beta_{ijk} = \begin{cases} 1 & t_k \text{ is served right after } t_j \text{ by } v_i \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The order of tasks belonging to a crane $c_i \in C$ is described by (3).

$$\gamma_{ijk} = \begin{cases} 1 & t_k \text{ is followed by } t_j \text{ in } c_i \\ 0 & \text{otherwise.} \end{cases} \tag{3}$$

The primary objective in truck dispatching problems for container terminals involves enhancing the port company's profitability by increasing turnover and minimizing the waiting time of ships. To evaluate the extent to which this objective is accomplished, various metrics can be employed. In this study, we adopt the objective of *TEU per hour (TEU/h)*, which computes the number of TEUs processed per hour by all QCs under consideration. Port companies commonly use this metric as the principal indicator for comparing operational efficiency against competitors. It is noteworthy that the TEU/h metric is analogous to the makespan employed in numerous scheduling problems when the task set remains constant. Consequently, our truck dispatching problem can be modeled as follows:

$$\max\left(\frac{\sum_{i=1}^{n} z_i}{\max(E) - \min(S)}\right) \tag{4}$$

$$\sum_{i=1}^{m} \alpha_{ij} = 1 \quad \forall t_j \in T \tag{5}$$

$$\sum_{i=1}^{m} \sum_{k=1}^{n} \beta_{ijk} \leq 1 \forall t_j \in T \tag{6}$$

$$\sum_{j=l}^{n} \sum_{k=1}^{l} \gamma_{ijk} \leq q \cdot y_i \quad l \in [1, n] \tag{7}$$

$$s_i = \max \begin{cases} \sum_{j=1}^{n} \sum_{k=1}^{m} \beta_{kji} \cdot (\tau(b_j, a_i) + e_j) \\ \tau(d, a_i) \cdot (1 - \sum_{j=1}^{n} \sum_{k=1}^{m} \beta_{kji}) \end{cases} \tag{8}$$

$$e_i = \max \begin{cases} \max(s_i, \sum_{j=1}^{n} \sum_{k=1}^{m} \gamma_{kji} \cdot e_j) + \tau(a_i, b_i) + r_i \\ \sum_{j=1}^{n} \sum_{k=1}^{m} \gamma_{kji} \cdot e_j + d_i \end{cases} \tag{9}$$

The objective delineated in (4) represents the average production rate per unit of time (hour), where $\max(E)$ and $\min(S)$ correspond to the completion time of the final task and the start time of the first task, respectively. The constraint articulated in (5) guarantees that each task is assigned exclusively to one truck, while the constraint in (6) ascertains that each task is succeeded by a maximum of one other task or none if it represents the truck's final task. For each crane, constraint (7), following container terminal transportation rules, ensure that tasks involving the same crane cannot commence until the preceding task is concluded, except for the unloading tasks in QCs where the operational sequence can be interchanged between $q = 3$ neighboring tasks. Constraints (8) and (9) calculate the start and end times of tasks, verifying that tasks initiate crane operation only after completing preceding task operations.

The truck dispatching challenge in maritime container terminals is deemed NP-hard, as it can be reduced to the vehicle routing problem [43]. This classification means finding the optimal solution becomes computationally intensive as the problem increases. While past studies utilized metaheuristics assuming crane operation times r_i for tasks in T were predictably constant, these times are notably uncertain in practice, making full predictions impractical. Additionally, our efforts to address this using mixed-integer programming (MIP) solver based on the formulations faced difficulties due to the problem's NP-hard complexity and parameters' uncertainty, hindering the identification of feasible solutions.

As a result, we have characterized this issue as an online optimization problem, which we address utilizing a GP ensemble method. Details will expounded in the next section.

IV. Methodologies

This section delineates the dynamic dispatch system at container ports and outlines the process for training and testing the dynamic dispatch algorithm using historical data and simulators. Moreover, we will expound upon the five heuristics utilized in the experiments. Specifically, the manual heuristics, DRL-HH, DRL-GPHH, DRL-EHH, and DRL-GPEHH, are introduced in this paper.

To adapt to the evolving external environment and to generate a dispatch solution in sub-real time, the majority of container ports currently employ a dynamic truck dispatching system to assign tasks. The dynamic dispatching system gathers real-time environmental parameters, vehicular congestion and sequence, QC working and waiting conditions, current task queues, and other data from the terminal operation system (TOS). Utilizing this information, a dynamic dispatching algorithm can allocate idle trucks to the most suitable tasks. The dynamic dispatching algorithm plays a pivotal role in this process, and the algorithm's logic for truck dispatching can significantly influence the efficiency of port operations.

So far, most ports still rely on expert heuristics as dynamic dispatching algorithms for truck dispatching. Building on expert heuristics, a recent study proposed a DRL-HH method [7] that employs RL to select different expert heuristics, achieving superior performance over individual manual heuristics and GP-generated heuristics. This paper further proposes the DRL-GPHH method, which combines the strengths of RL and GP in different search spaces by using RL to select different GP-generated low-level heuristics, enhancing the algorithm's versatility and performance. The DRL-GPHH algorithm does not require the design of various manual expert heuristics but generates heuristics through GP training, making it easily transferable to other complex real-world optimization problems beyond port dispatching.

We observed that both DRL-GPHH and DRL-HH underutilize the diverse knowledge encapsulated in unselected heuristics, thereby compromising their generalization capabilities and performance. In light of these insights, we introduced DRL-GPEHH, which leverages an RL agent to dynamically fine-tune the weights of GP-generated heuristics in response to varying environmental states. This innovation enhances the algorithm's stability, generalization, and performance while also facilitating greater automation in algorithmic design. To further delineate the contrasts between GP and manual heuristic ensembles, we also introduced DRL-EHH as a control, employing RL to dynamically modulate the weights of manual heuristics.

A. Dynamic Truck Dispatching and Port Simulator

In traditional ports, loading and unloading tasks are assigned to individual QCs, while container trucks are designated to specific QC task pools. A designated truck carries out tasks within its assigned pool, waiting for new tasks if none are available. Port operators continually adjust the number of trucks in each pool based on remaining tasks, QC availability, operational speed, vessel berthing, personal experience, and other parameters to minimize waiting time for both QCs and container trucks. Although this scheduling method has shown positive results, it relies heavily on the dispatcher's experience, and their workload can be substantial. Inefficient assignment of idle trucks can negatively impact overall port operations efficiency.

Given the challenges in port operations, dynamic truck dispatch algorithms have been introduced to dispatch all trucks dynamically, allowing for the decoupling of trucks from specific QCs. This approach enhances port efficiency by enabling trucks to undertake tasks from all QCs.

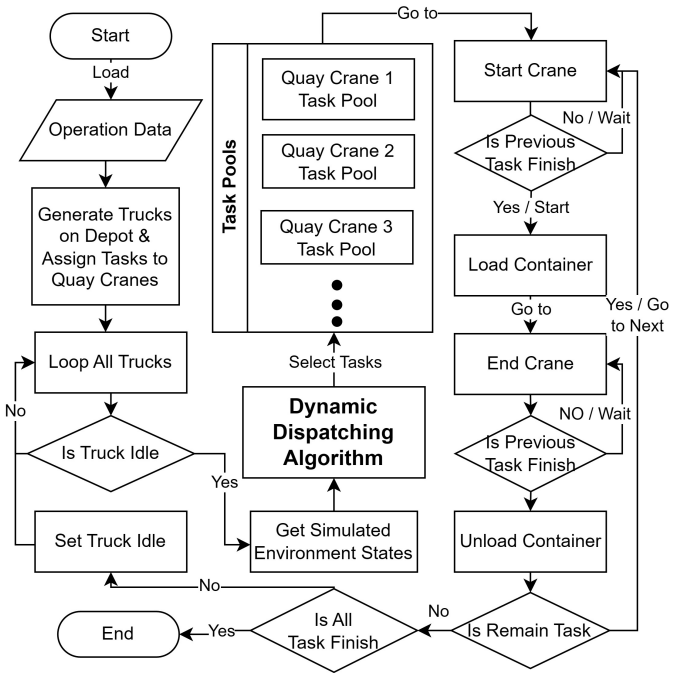

Fig. 5. Event-based Port Simulator Flow Chart

To facilitate the testing and training of dynamic truck dispatch algorithms, we developed an event-driven simulator, as illustrated in Fig. 5, using historical port data to emulate the port operation process. Upon container truck availability, the system assigns either one 2TEU task or two 1TEU tasks. The truck proceeds to the crane, and loading begins once the prerequisite task is completed, with loading time influenced by random factors. Afterward, the truck moves for unloading, and upon completion, either returns for remaining containers or awaits new assignments. This process highlights the complexity of truck dispatching in maritime container terminals and motivates the proposal of dynamic heuristic scheduling algorithms.

B. Manual Heuristic

The manually designed heuristic, developed by experts and currently serving as the prevalent method employed in ports [4], serves as the baseline for comparison in this study. This algorithm devises a series of environment parameters, such as the optimal number of trucks for a given QC, the maximum number of trucks allowed for a QC, and the priority assigned to the QC based on the operator's experience. The algorithm automatically scores each task, considering the parameters, the busyness of the QC to which the task is assigned, and the time required for the truck to reach the task's starting location. Ultimately, the task with the best score is selected as the output task and dispatched to the idle truck.

This manual heuristic algorithm has been successfully implemented for years at Ningbo port. With the support of this

manual heuristic dynamic dispatching algorithm, port operators no longer need to dispatch trucks continually; instead, they only need to adjust the vehicle dispatching plan as necessary, yielding improved results compared to previous methods. Although the manual heuristic exhibits limited performance, it offers several advantages, including no need for training, strong generalization, and consistent performance across different data sets. Consequently, it serves as the baseline in this paper.

C. DRL-GPHH & DRL-HH

Reinforcement learning methods have demonstrated considerable advantages in addressing complex optimization problems. However, in applications applied to real-life scenarios, it is possible to make unpredictable decisions when encountering unknown scenarios due to its black-box characteristics. Such a situation is usually unacceptable in a real-life operation scenario. To ensure the safety and order of the operation, all decisions must be traceable, adaptable, and understandable.

Therefore, a DRL-GPHH approach is proposed, wherein a reinforcement learning network selects various GP-generated heuristics for truck dispatching. Owing to the white-box nature of the heuristics, even when a heuristic is chosen by reinforcement learning, the final decision is derived through the selected heuristic, making it relatively interpretable. This method effectively amalgamates the distinct advantages of RL and GP heuristics and is readily applicable in real-world port operation environments.

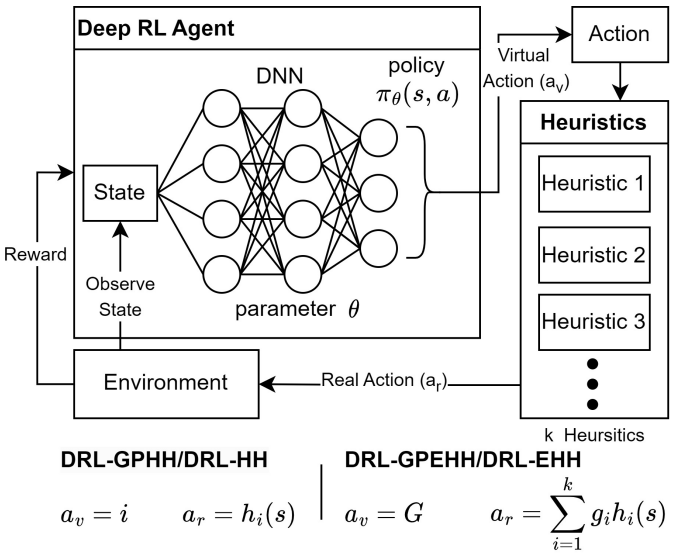

Fig. 6. DRL-HH/DRL-GPHH & DRL-GPEHH/DRL-EHH Framework

As illustrated in Fig. 6, in the context of reinforcement learning genetic programming hyper-heuristics, given the `state` of an environment, the algorithm selects an appropriate `virtual action` according to a specific `policy`. This action subsequently picks a heuristic. Upon execution of the heuristic, the actual action alters the environment, leading to a transition to a new state, denoted as S'. With each action execution, the algorithm receives a `reward` value and the estimated quality of the new state. The algorithm then adjusts its policy based on the magnitude of the reward value, ultimately maximizing the sum of rewards obtained when all steps are completed and the state reaches the `terminal` state.

In contrast to DRL-HH, which relies heavily on experts' domain knowledge for task scheduling and compromises its performance and generalization, DRL-GPHH replaces expert manual heuristics with GP-generated heuristics. In experiments, DRL-GPHH outperformed DRL-HH, highlighting the potential for collaborative problem-solving between RL and GP. Furthermore, given that GP-generated heuristics do not require the input of experts with extensive knowledge of the problem, they present a scalable solution for a range of similar complex optimization problems.

Specifically, DRL-GPHH and DRL-HH in this paper follow all the DRL settings in previous work [7], which adopts the double deep Q-learning (DDQN) algorithm [44] as a high-level heuristic and uses a four-layer DRL network. The number of neurons in each layer is set as follows: 80, 100, 180, and 10, respectively. Rectified linear unit (ReLU) functions serve as the activation functions for each hidden layer, and the learning rate is set at 0.001.

The details of DRL-GPHH and DRL-HH are as follows:

1) Environment: The above-mentioned event-based port simulator serves as the training environment for DRL-GPHH and DRL-HH. This simulator uses the map and historical data of the Ningbo Meishan port that we cooperate with to simulate the real-world port. The simulator inputs the current state (S) into DRL-HH, and after the action (assignment of the truck) made by DRL-HH, it will deduce according to the rules and historical data to obtain the subsequent state (S'). While the simulator is running, various metrics are calculated, which are used as rewards to assist the training of DRL-HH.

2) State: The state, a set of matrices, represents the current operating environment in which the DRL-GPHH and DRL-HH will learn to select distinct actions depending on the state. In this study, the state is generated based on the trucks requiring task assignment, the current tasks and queuing statuses of QC) and YCs, the QC type and working status, the remaining number of tasks, and the average operating time of the cranes. The state matrix is of dimensions $i * j$, where i denotes the number of parameters incorporated within the state to describe a QC status, and the number of QC is represented by j. The specific parameters include:

- The QC remaining task number.
- The QC available task number.
- The QC bounded trucks number.
- The QC working status: 0 for unload and 1 for load.
- The QC type: 0 for standard and 1 for remote control.
- The minimum truck move time to a task start crane.
- The minimum waiting truck number of beginning cranes.
- The minimum waiting truck number of ending cranes.
- The average loading time of the beginning cranes.
- The average unloading time of the ending cranes.

It is important to note that since this truck dispatching problem in the container port is an optimization problem involving uncertainty, the state transition in this problem does not follow

the conventional $s' = E(s, a)$ form. Instead, an uncertain parameter u is introduced, resulting in $s' = E(s, a, u)$. In this case, a represents the action chosen by the low-level heuristic selected by DRL-HH, while u is obtained during the simulator run.

3) Actions: DRL-GPHH and DRL-HH agents produce a virtual action to select a low-level heuristic, generating a real action for interacting with the environment.

DRL-GPHH uses the GP-generated low-level heuristics. Genetic programming was employed to learn and generate 10 distinct heuristics on training datasets across different scenarios. Each heuristic can manage the scenarios it has been trained on.

DRL-HH uses the manually designed low-level heuristics. These manual heuristics consider the distance from the task's starting point to the crane, the uniformity of the workload distribution among QC job lists, and the task's urgency. For each indicator, three thresholds are designed, resulting in nine manual heuristics. In conjunction with the aforementioned expert-designed manual heuristic IV-B, there are 10 heuristics available for selection.

4) Reward: Concerning rewards, the design for the DRL-GPHH and DRL-HH rewards adheres to the approach employed in the previous study [7], using the idle time of QCs as the primary reward component. For each assigned task i, its reward r_i is computed as $r_i = e_{i-1} - s_i$. The objective is to enhance port operational efficiency and reduce QC idle time; thus, the reward is a negative number, indicating that the smaller the QC idle time, the greater the reward. Moreover, since the QC idle time cannot be calculated when the action is executed, it must be updated after completion, requiring intermittent reward calculations. This process involves episodically computing the rewards for previous actions upon completing each task.

D. DRL-GPEHH & DRL-EHH

Although DRL-GPHH has demonstrated promising performance, it exhibits poor generalization when dealing with real historical data. Specifically, its performance deteriorates when encountering previously unseen data during training. This occurs because each scheduling operation in DRL-GPHH utilizes only one GP heuristic, with each heuristic incorporating information from the trained scenarios. Optimal performance is achieved only when the current scenario closely resembles one of the trained scenarios. However, the unselected heuristics in DRL-GPHH also encompass valuable information. By judiciously combining this information, the algorithm can adapt to a broader array of unseen scenarios and enhance performance.

As illustrated in Fig. 7, single-choice hyper-heuristics approaches (e.g., DRL-HH, DRL-GPHH) enable the high-level heuristic to select only one low-level heuristic for a given situation. For instance, if the high-level heuristic prioritizes task urgency, other factors, such as task proximity and node busyness, are consequently disregarded.

In contrast, ensemble hyper-heuristic frameworks are capable of selecting multiple heuristics, effectively countering the limitation of depending on a single heuristic. Both DRL-EHH and DRL-GPEHH methods dynamically allocate varying weights to different low-level heuristics, adapting effectively to various situations. This allows for a more nuanced consideration of multiple factors based on their assigned weights, including task proximity, node busyness, and task urgency. As a result, these ensemble frameworks make more accurate judgments and better handle multi-scenario, complex, real-world optimization problems, as evidenced by our experimental results.

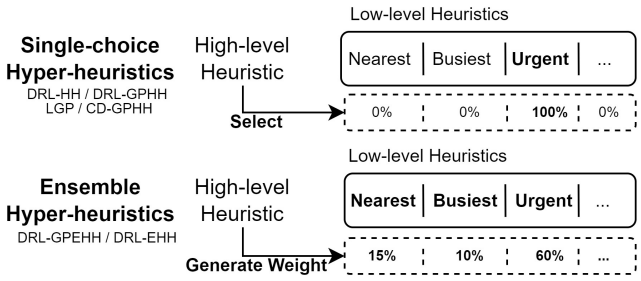

Fig. 7. Single-choice Hyper-heuristics vs. Ensemble Hyper-heuristic

Our experiments tested various methods for integrating multiple GP heuristics, as presented in Section V. We found that the best-performing approach was to use DRL to adjust the weights of each GP heuristic continuously. As illustrated in Fig. 6, DRL-GPEHH employs a reinforcement learning agent as a gating mechanism to assign weights to multiple heuristic experts, subsequently combining the results of these heuristics to produce the final assignment. The DRL-GPEHH offers several advantages over DRL-GPHH primarily due to the integration of multiple expert heuristics, which leads to improved decision-making and adaptability. The key benefits include:

- Diversity and adaptability: DRL-GPEHH incorporates various GP heuristics, each with strengths and weaknesses. This diversity allows the algorithm to adapt to different situations and select the most suitable heuristic for a given scenario, leading to better overall performance and leveraging the strengths of other heuristics to compensate for their limitations.
- Learning efficiency: By utilizing the knowledge and experience embedded in multiple heuristics, DRL-GPEHH can potentially accelerate the learning process. As a result, the algorithm can converge to a near-optimal solution more quickly than a normal DRL-HH, which relies solely on its learning and exploration.
- Knowledge transfer: DRL-GPEHH can benefit from knowledge transfer between the heuristics, allowing the algorithm to capitalize on their combined knowledge. This leads to more effective exploration and exploitation strategies, ultimately improving the solutions' quality.
- Scalability: The ensemble approach enables DRL-HH to handle a broader range of problems and larger-scale instances. By combining the expertise of multiple heuristics, the algorithm can scale better to tackle complex tasks and adapt to new, unseen scenarios.
- Algorithm automation: The utilization of GP-generated

heuristics obviates the need for expert inputs, thereby significantly enhancing the automaticity of algorithm generation. This culminates in an efficient and self-reliant design process. This heightened level of automation simplifies the task of addressing complex real-world problems, thereby extending the versatility and applicability of the method.

In summary, DRL-GPEHH outperforms DRL-GPHH by leveraging multiple GP heuristics' strengths, resulting in improved adaptability, robustness, learning efficiency, knowledge transfer, and scalability. These advantages make it more suitable for solving complex and dynamic problems like container terminal truck dispatching in various domains.

Furthermore, to facilitate a more precise comparison of the performance of DRL-HH, DRL-GPHH, and DRL-GPEHH, DRL-EHH is proposed to act as a control. DRL-EHH employs the same manual heuristics as those in DRL-HH, replacing the GP-generated low-level heuristics in DRL-GPEHH. Except for the low-level heuristics, all settings of DRL-EHH are identical to those of DRL-GPEHH. Consequently, we only describe the structure of DRL-GPEHH.

To handle multi-dimensional actions and output continuous weights for multiple heuristics simultaneously, DRL-GPEHH employs a multi-action proximal policy optimization (PPO) as a gate agent instead of the double deep Q-Network (DDQN) used in DRL-GPHH.

PPO is an advanced policy optimization algorithm introduced in 2017 [45] designed to overcome the challenges faced by other algorithms, such as trust region policy optimization (TRPO) and asynchronous advantage actor-critic (A3C). It enhances sample efficiency and stability by utilizing a trust region approach and employing a clipped objective function shown in (10). Here, $r_t(\theta)$ is the probability ratio between the current policy and the old policy, represented as $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. The variable $\hat{A}_t$ denotes the estimated advantage function at time step t, and ϵ is a hyperparameter controlling the degree of trust region in the policy update.

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t) \right] \tag{10}$$

Combined with the formula, the training process of the multi-action PPO is described in Algorithm 1.

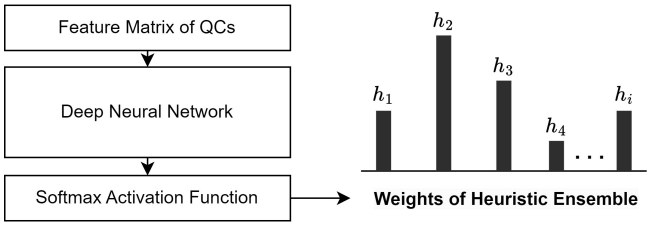

Fig. 8. Neural Network Output of DRL-GPEHH and DRL-EHH

To maintain fair competition, DRL-GPEHH retains the deep neural network (DNN) settings of DRL-GPHH. However, it doubles the networks to serve as a policy network and a value network, respectively. Furthermore, a softmax activation function is incorporated following the DNN output. As illustrated in Fig. 8, DRL-GPEHH produces a continuous weight vector for each heuristic. Based on these weights and the outputs of the heuristics, DRL-GPEHH synthesizes the final real action for dispatching the truck.

The environment and status settings of DRL-GPEHH are consistent with DRL-HH, but some modifications have been made in the following parts:

1) Actions: Similar to DRL-GPHH, DRL-GPEHH does not directly interact with the environment through actions as depicted in Fig. 6. Instead, it employs the same GP-generated heuristics used in DRL-GPHH as low-level heuristics. The distinction between the two lies in their utilization of low-level heuristics: in DRL-GPEHH, instead of directly selecting the most appropriate task for output as in DRL-GPHH, the low-level heuristics generate a task ranking based on their internal rules.

Algorithm 1 Proximal Policy Optimization (PPO) Training

Initialize policy parameters θ and value function parameters ϕ
for each iteration **do**
  Collect a set of trajectories τ using the current policy π_θ
  Compute rewards-to-go R_t for each time step in trajectories
  Compute advantage estimates A_t using value function V_ϕ
  for each optimization epoch **do**
    for each time step t in trajectories **do**
      Compute probability ratio $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
      Compute surrogate objective $L_t(\theta) = \min(\rho_t(\theta)A_t, \text{clip}(\rho_t(\theta), 1-\epsilon, 1+\epsilon)A_t)$
      Perform gradient ascent on θ to maximize $E_t[L_t(\theta)]$
      Update value function parameters ϕ by minimizing the value loss
    end for
  end for
  Update policy $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
end for

Algorithm 2 Weighted Ensemble Scheduling

Require: $tasks, heuristics, weights$
Ensure: $best\_task$
$scores \leftarrow \{ta : 0 \mid ta \in tasks\}$
for h, w in zip($heuristics, weights$) **do**
  $ranked\_tasks \leftarrow h.rank(tasks)$
  for i, t in enumerate($ranked\_tasks$) **do**
    $scores[t] \leftarrow scores[t] + (i+1) * w$
  end for
end for
$best\_task \leftarrow \arg\min_{t \in tasks} scores[t]$ **return** $best\_task$

Based on the rankings of available tasks generated by heuristics, DRL-GPEHH employs the weighted ensemble

ranking method, as illustrated in Algorithm 2, to combine the ranks and determine the best task. This algorithm takes a set of tasks, an array of heuristics, and their respective weights as input, with the objective of identifying the optimal task from the available options.

2) Reward: Given that the action space of DRL-GPEHH is much larger than that of DRL-GPHH, and considering the challenges introduced by delayed rewards in terms of temporal credit assignment, exploration, and convergence, the rewards employed in DRL-GPHH become less suitable for DRL-GPEHH. Specifically, the temporal credit assignment problem arises due to the difficulty associating the correct action with an observed reward when rewards are delayed. This can slow the learning process or cause the agent to learn suboptimal policies. Additionally, delayed rewards can impact the exploration-exploitation trade-off, as the agent may need to explore the environment extensively before discovering the long-term consequences of its actions, potentially delaying convergence to an optimal policy.

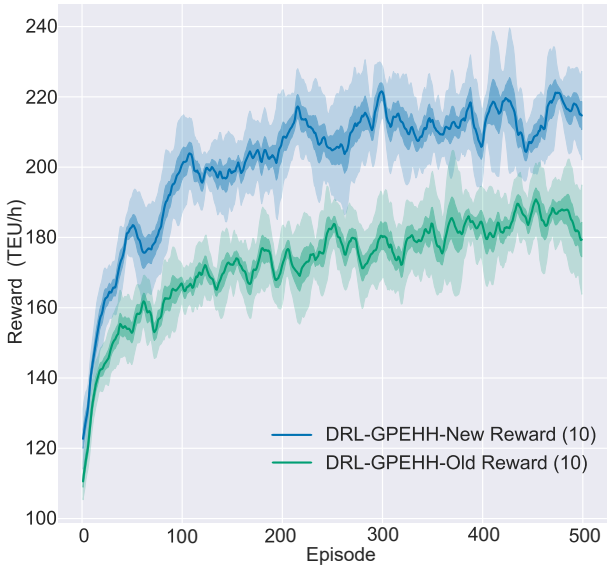

Fig. 9. Performance of DRL-GPEHH with Varying Reward Structures

To address these challenges, we introduced a new reward function that combines reward shaping and imitation learning to enhance the learning process in the presence of delayed rewards by improving credit assignment, encouraging efficient exploration, and stabilizing convergence. The new rewards are designed as $reward = e_{i-1} - s_i - \delta cov(O_r, O_m)$, where δ is the weight, cov is the covariance calculation function, O_r is the task ranking given by DRL-GPEHH, and O_m is the ranking given by manual heuristics described in Section IV-B.

The reward for rankings similar to the manual heuristic can be adjusted by setting different weight values, denoted by δ. In this paper, δ is set to κ/en, where κ is a scaling factor that can be adjusted according to the size of the previous reward term, which is set to 10 in this study, and en represents the number of training episodes. The weight of this reward gradually decreases as the number of training generations increases, encouraging the algorithm to learn from the manual expert heuristic initially and reducing the influence of convergence to the manual heuristic on the algorithm's ability to reach a superior solution during later stages of learning.

Although the reward component of $e_{i-1} - s_i$ in DRL-GPEHH is the same as that in DRL-HH, it must be calculated after the task completion, while the $-\delta cov(O_r, O_m)$ component can be obtained immediately, addressing the reward delay issue. In our newly designed reward structure, we guide the DRL-GPEHH to learn like the manual heuristic by encouraging reward allocations that resemble task rankings generated by the manual heuristic. According to the experimental results in the training datasets detailed in Section V with 10 random seeds shown in Fig. 9, the newly designed rewards indeed achieve better performance, speed up convergence, and resolve the problem of delayed rewards in container terminal truck dispatching problem.

V. Experiments and Discussion

In the ensuing section, we undertake a comprehensive evaluation of DRL-GPHH and DRL-GPEHH, focusing on a multifaceted container port dispatching problem marked by uncertain parameters. This analysis is positioned against DRL-HH and DRL-EHH to elucidate the advantages of integrating GP with RL. Given the proven robustness and reliability of the manually delineated heuristic outlined in Section IV-B, which has demonstrated considerable efficacy in practical port applications, we employ it as a benchmark baseline. Thus, all ensuing comparative analyses are premised on enhancements made relative to this manual heuristic (Imp.). Additionally, this section incorporates ablation studies and sensitivity analysis, furnishing insights into the conducive elements underpinning the exceptional performance of DRL-GPEHH.

A. Experiment Design

As this work aims to develop an algorithm that can be deployed in a real-world port to enhance its operational efficiency, all data used in the experiments are derived from actual historical operating data of Ningbo Meishan Port, with which we collaborate. We sampled 20 days of operation data to generate 10 training sets and 10 test sets, each containing 4,000 tasks. All experiments are conducted using the event-driven simulator we developed, with simulator parameters adjusted based on historical operating data. There are two ship berths 10 operating QCs, and the number of container trucks varies between 50 and 80. Moreover, as mentioned in Section IV-A, the operating times for QC and YC are derived from real-world historical data, while the truck travel times are drawn from the historical time distribution.

We trained 10 GPs for 300 generations with 100 different random seeds on the same set of 10 training datasets and selected one best-performing individual from each dataset to form 10 GP-generated low-level heuristics. The GP algorithm and parameters are consistent with the GP algorithm with logic operators (LGP) described in our previous paper [9], featuring a population size of 1024 and crossover, mutation, and reproduction rates of 60%, 30%, and 10%, respectively. All algorithms were executed 100 times with different random

seeds, and the training phase consisted of 1000 episodes. Subsequently, we assessed the performance of DRL-HH [7], DRL-GPHH, DRL-EHH, and DRL-GPEHH on the test sets. The average results from 100 runs for training and testing are presented in Tables I and II.

B. Experiment Results

The experimental results demonstrate that, regardless of whether using DRL to select single heuristics or a set of heuristics, GP-generated heuristics outperform manually-designed heuristics. Replacing human-designed low-level heuristics in DRL-HH with GP-generated heuristics provides a performance boost of 1.42%, while DRL-GPEHH achieves a 9.88% improvement over DRL-EHH when multiple GP heuristics are jointly involved in decision making as an ensemble at each step. It is worth noting that due to the theoretical optimization upper bound in the port truck dispatch problem, the closer the TEU/h is to the upper limit, the more difficult it becomes to further improve performance. The 17.77% performance improvement of DRL-GPEHH over manual heuristics demonstrates its effective combination of knowledge from multiple GP-generated heuristics. Unlike DRL-GPHH, although both use the same 10 heuristics, the ensemble-based approach leverages the knowledge from multiple GP-generated heuristics to achieve better performance.

TABLE I
DRL-HH, DRL-GPHH, DRL-EHH AND DRL-GPEHH TRAINING RESULTS (TEU/H)

No.	Manual	DRL HH	DRL GPHH[1]	DRL EHH	DRL GPEHH[1]
1	202.36	222.91	225.50	218.18	240.75
2	188.54	205.56	204.56	201.46	225.10
3	182.31	199.26	201.73	198.14	212.46
4	191.33	205.66	207.86	204.54	219.60
5	186.26	196.52	204.75	202.54	221.28
6	190.69	204.51	211.14	205.85	220.87
7	193.33	202.14	204.17	208.60	234.28
8	191.59	203.24	198.27	203.33	224.02
9	186.14	198.19	204.97	203.05	216.79
10	190.98	204.97	206.91	208.05	226.54
Avg.	190.35	204.30	206.98	205.37	224.17
Imp.	**N.A.**	**7.32%**	**8.74%**	**7.89%**	**17.77%**

[1] DRL-GPHH and DRL-GPEHH significantly differ from other algorithms, $p < 0.05$.

In contrast to the excellent performance of the GP-generated heuristic ensemble, the application of a manual heuristic ensemble in DRL-EHH results in only a 0.57% performance improvement compared to DRL-HH. This can be attributed to the simplicity and high similarity across the adopted manual heuristics, which, while capable of producing satisfactory results for straightforward and popular scenarios, makes it challenging to improve performance further. Conversely, the GP-generated heuristic ensemble, owing to its complex internal structure and knowledge that encompasses various scenarios, has the potential to achieve superior performance. However, continuous weight adjustments are required in different environments to maximize the utilization of knowledge from multiple GP heuristics. The experiments supporting this statement and the reasons for the excellent performance of DRL-GPEHH will be presented in the subsequent subsection.

Next, we put the trained DRL-HH, DRL-GPHH, DRL-EHH, and DRL-GPEHH into a test environment completely different from the training environment with a broadly similar baseline.

As delineated in Table II, the algorithms DRL-GPHH and DRL-GPEHH, which employ GP-generated low-level heuristics, outperform their manually designed heuristic counterparts, DRL-HH and DRL-EHH, by margins of 0.43% and 8.14%, respectively. This substantiates the notion that integrating DRL with GP-generated heuristic ensembles can yield substantial performance improvements, even in unfamiliar testing conditions. Moreover, DRL-EHH and DRL-GPEHH exhibit enhanced consistency in the test set by leveraging an ensemble of GP heuristics at each decision point. The performance decrement observed for these ensemble-based models on the test set is approximately 1% less than that for the non-ensemble alternatives. This finding not only reinforces the effectiveness of ensemble approaches in navigating unknown scenarios but also highlights the robustness and practical applicability of our proposed techniques, which demonstrate minimal performance attrition in dynamically changing real-world contexts.

TABLE II
DRL-HH, DRL-GPHH, DRL-EHH AND DRL-GPEHH TEST RESULTS (TEU/H)

No.	Manual	DRL HH	DRL GPHH[1]	DRL EHH	DRL GPEHH[1]
1	190.48	199.74	203.41	215.88	227.40
2	203.54	209.55	210.20	203.01	231.24
3	189.82	195.91	205.08	203.83	217.97
4	187.04	198.26	195.73	202.24	213.35
5	191.43	202.51	202.96	199.57	216.45
6	182.90	193.37	192.97	203.92	207.67
7	193.62	204.61	201.89	202.78	221.44
8	185.86	194.05	193.95	201.60	212.82
9	191.29	200.59	203.29	198.85	217.13
10	188.87	195.22	192.55	199.95	221.30
Avg.	190.48	199.38	200.20	203.16	218.68
Imp.	**N.A.**	**4.67%**	**5.10%**	**6.66%**	**14.80%**
Dec.	N.A.	-2.65%	-3.64%	-1.24%	-2.96%

[1] DRL-GPHH and DRL-GPEHH significantly differ from other algorithms, $p < 0.05$.

Across both training and test sets, DRL-GPEHH outperforms all other algorithms, substantiating the synergistic potential between DRL and GP ensembles in augmenting algorithmic performance. This ensemble approach not only enhances generalization and robustness but also resolves the limitations of DRL-HH, which is overly reliant on the quality and diversity of pre-defined heuristics. Incorporating GPHH adds a new dimension of diversity, adaptability, and efficiency. Furthermore, by automating the generation of low-level heuristics through GP, both DRL-GPHH, and DRL-GPEHH eliminate the need for manual expert design, thereby considerably increasing the level of automation in algorithm design for complex problems like marine port truck dispatching. This automated approach proves advantageous in adapting to various complex, real-world optimization challenges, offering a

TABLE III
PERFORMANCE OF OTHER ALGORITHMS (TEU/H)

	Perf.	Manual	Random	DDQN	PPO	LGP	CD-GPHH	VMHE	RMHE	WRMHE	BGPE	VGPE	RGPE	WRGPE
Train	**Avg.**	190.35	160.84	168.51	172.24	198.43	208.49	169.23	177.03	201.65	202.72	203.02	208.25	204.39
	Imp.	**N.A.**	**-15.50%**	**-11.47%**	**-9.51%**	**4.24%**	**9.53%**	**-11.09%**	**-7.00%**	**5.94%**	**6.50%**	**6.66%**	**9.40%**	**7.37%**
Test	**Avg.**	190.48	161.28	161.35	167.84	194.26	198.21	169.64	176.71	196.66	197.13	198.92	204.25	203.54
	Imp.	**N.A.**	**-15.33%**	**-15.29%**	**-11.89%**	**1.98%**	**4.06%**	**-10.94%**	**-7.23%**	**3.24%**	**3.49%**	**4.43%**	**7.22%**	**6.86%**

scalable, flexible, and robust solution across diverse problems.

Finally, it has been substantiated that DRL-GPEHH can yield superior results compared to DRL-HH, DRL-GPHH, DRL-EHH, and manual heuristics in container port truck dispatching. Although DRL-GPEHH has not yet been implemented in real-life port settings, our manually crafted heuristic algorithm, used as a baseline in this study, has been successfully utilized at Ningbo Port for several years. Statistical analysis conducted by the port reveals that work efficiency has increased by 8.1%, while ship docking time has decreased by 2.2%. This high-performing algorithm has resulted in time savings, facilitated the handling of more ships, and consequently, significantly enhanced the profitability of the port company. As a future endeavor, we plan to collaborate with relevant stakeholders to comprehensively evaluate and deploy the proposed DRL-GPEHH algorithm in real-world scenarios.

C. Ablation and Sensitivity Analysis

Subsequently, we extend our experimental investigation to corroborate the superior performance of the DRL-GPEHH. We formulate three hypotheses to guide this analysis: first, that standard RL algorithms struggle to converge in complex dynamic environments with extensive action spaces, thereby necessitating a hyper-heuristics framework; second, that basic GP hyper-heuristics are limited in their generalizability and capacity to handle intricate scenarios, thus impeding overall algorithmic performance on test sets; third, The optimal performance of the GP ensemble methodology is realized by dynamically assigning weights to GP individuals.

To empirically evaluate the performance of standard RL algorithms in complex test environments, we trained DDQN and PPO algorithms using training sets. The configurations for DDQN and PPO were strictly in line with established settings in the literature [44], [45], and each was subjected to 1,000 training episodes. For a more comprehensive assessment, we also incorporated a random dispatching algorithm into our evaluation framework. As the test results presented in Table III, it is unequivocally evident that DDQN failed to acquire any meaningful information during training, performing on par with randomized strategies. Similarly, PPO exhibited only marginal improvements over random dispatching and did not approach the effectiveness of heuristic methods. These experimental outcomes compellingly substantiate our first hypothesis: conventional RL algorithms face significant challenges in converging within complex and dynamic environments characterized by sparse rewards and expansive action spaces. This underscores the necessity for adopting a hyper-heuristics framework, where low-level heuristics replace traditional actions to stabilize the learning environment. Such an arrangement permits RL algorithms to assimilate valuable information and exhibit improved performance, as the DRL-HH model exemplifies.

Based on the second hypothesis, we trained two distinct populations using LGP and Cooperative Double-Layer Genetic Programming hyper-heuristics (CD-GPHH) [9]. Both approaches employ a hyper-heuristics framework that leverages GP as a high-level selector for low-level GP heuristics. The configurations for LGP and CD-GPHH were consistent with those outlined in our prior work [9]. During training, each population was exposed to a sequence of 1000 generations across multiple training sets, each initialized with 100 different random seeds. The best-performing individuals were then selected for further analysis. Evaluation of the training set yielded average performance improvements over the manual heuristic of 4.24% and 9.53% for LGP and CD-GPHH, respectively. However, the corresponding gains on the test set were comparatively modest: 1.98% and 4.06%, as shown in Figure III. Although CD-GPHH outperformed LGP due to its double-layer architecture, it fell short of the 15% improvement observed in smaller test datasets from our previous study [9]. The diminished performance on the test set underscores the limitations of relying solely on GP hyper-heuristics for complex optimization tasks in large-scale, multi-scenario environments. This substantiates our decision to integrate RL with heuristic ensembles for tackling real-world, large-scale challenges.

To further validate our second hypothesis, we implemented three distinct ensemble methods—voting, ranking, and weighted ranking—to amalgamate manual and GP heuristics. The effectiveness of these methods was then assessed using the test set delineated in Section V. Each heuristic selects an optimal task in the voting mechanism, with the majority vote determining the final output. Ranking involves each heuristic assigning a rank to all tasks, culminating in the task with the highest aggregated rank chosen as the output. The weighted ranking method extends this by applying weights to the summed ranks according to Algorithm 2. For comparative analysis, we also introduced the Best GP Ensemble (BGPE) method, which involves assessing 10 unique GP low-level heuristics on each dataset and subsequently choosing the most productive one.

Then, to get the weight value used in the weighted ranking method, we calculated the probabilities of DRL-HH and DRL-GPHH, selecting each heuristic, as depicted in Figure 10. For manual heuristics, DRL methods preferred one or two specific actions, such as Act1 and Act2, while less frequently opting for actions like Act8, Act9, and Act10. In contrast, the

TABLE IV
STANDARD DEVIATION OF ACTION SELECT RATIO IN DRL-EHH AND DRL-GPEHH

	Act 1	Act 2	Act 3	Act 4	Act 5	Act 6	Act 7	Act 8	Act 9	Act 10	Avg.
DRL-EHH	5.03%	5.10%	0.62%	3.13%	0.23%	0.99%	3.43%	0.03%	0.00%	0.22%	**1.88%**
DRL-GPEHH	3.81%	6.84%	6.75%	7.40%	7.64%	6.90%	6.25%	4.22%	5.92%	8.94%	**6.47%**

selection probabilities for GP heuristics were more evenly distributed, with no single heuristic predominating. This suggests that while DRL methods gravitate towards better-performing manual heuristics, the uniform performance exhibited by GP heuristics makes it challenging for DRL-GPHH to identify a singular, superior heuristic. Such findings imply that the GP ensemble harbors a wealth of knowledge, making optimally assigning fixed weights nontrivial.

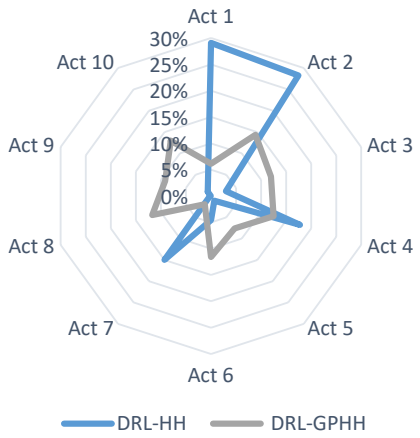

Fig. 10. DRL-HH/DRL-GPHH Action Selection Ratio

The inferior performance of the vote manual heuristic ensemble (VMHE) and rank manual heuristic ensemble (RMHE) in Table III, compared to the expert manual heuristic, indicates that when multiple manual heuristics make decisions, the subpar performance of certain heuristics adversely affects the overall decision-making performance. This leads to a final performance that is 10.94% and 7.23% worse than the individual expert manual heuristics in the manual heuristic ensemble, respectively. Moreover, when utilizing the statistical DRL probabilities to select each action shown in Fig. 10 as weights for the weighted rank manual heuristic ensemble (WRMHE) method, it achieves performance comparable to DRL-GPHH. This further underscores the significance of adjusting the weights of heuristic ensembles.

However, as demonstrated in Table III, we find that the vote GP ensemble (VGPE) and rank GP ensemble (RGPE) methods, which use the GP heuristic ensemble, perform significantly better than the best GP ensemble (BGPE) method that relies on a single heuristic. Nevertheless, the weighted rank GP ensemble (WRGPE) method, which utilizes the proportions of different actions selected by DRL-GPHH as weights, does not improve performance compared to RGPE. The performance of WRGPE is worse than that of the unweighted RGPE. This observation supports the argument that applying fixed weights to the GP heuristic ensemble does not enhance performance and may prove detrimental.

Furthermore, we calculated the standard deviations of the weights assigned to different actions by DRL-EHH and DRL-GPEHH. As shown in Table IV, the standard deviation of DRL-EHH is 1.38%, indicating that for manual heuristics, DRL-EHH tends to assign relatively consistent weights to each action. In contrast, the standard deviation of DRL-GPEHH is 6.47%, illustrating that the weights of each action in DRL-GPEHH vary during each scheduling, contributing to its superior performance.

These supplementary experiments substantiate our initial hypotheses: Conventional RL methods struggle with large-scale real-world problems and thus necessitate integrating a hyper-heuristics framework. Likewise, simplistic GP hyper-heuristics approaches fail to formulate a universally applicable model capable of managing large-scale, real-world scenarios. However, DRL-GPEHH by continually fine-tuning the weights of diverse low-level GP heuristics during the decision-making process—effectively capitalizes on the rich knowledge reservoir inherent in the GP heuristic ensemble. This leads to marked enhancements in performance, stability, and generalization capabilities.

VI. CONCLUSION

This paper presents DRL-GPHH and DRL-GPEHH, two advanced learning-assisted genetic programming methodologies tailored explicitly for dynamic truck dispatching in container terminals. Notably, the DRL-GPEHH approach rectifies limitations inherent in both DRL-HH and DRL-GPHH by ingeniously integrating an ensemble of GP heuristics. These heuristics, generated autonomously without the need for intricate expert input, empower more effective decision-making, furnish superior adaptability to diverse operational contexts, and significantly enhance the automation of algorithmic development.

In a comprehensive experimental study, DRL-GPEHH consistently outperforms competing algorithms across multiple metrics—namely, dispatching efficiency, generalization capabilities, and adaptability to fluctuating port operational scenarios. By pioneering a deep reinforcement learning agent as a gating mechanism, DRL-GPEHH astutely assigns weights to individual GP heuristics, thereby amalgamating their outputs to optimize task assignments. This elegant cooperation between DRL and the GP ensemble further strengthened by an inventive action and reward design—accelerates the convergence rate, thereby boosting both algorithmic stability and generalization capabilities.

The successful implementation of learning-assisted GP in dynamic truck dispatching highlights the considerable promise of GP and its ensemble variants within the broader framework of reinforcement learning hyper-heuristics, especially for

tackling intricate real-world optimization challenges. Future research could delve into co-training strategies that integrate GP with DRL, examining the impact of the number of low-level GP heuristics and determining optimal quantities. Additionally, there is potential to expand the application of this method to a broader range of dynamic scheduling problems.

REFERENCES

[1] E. Ahmed, M. S. El-Abbasy, T. Zayed, G. Alfalah, S. Alkass, Synchronized scheduling model for container terminals using simulated double-cycling strategy, Computers & Industrial Engineering 154 (2021) 107118.

[2] S. Nguyen, P. S.-L. Chen, Y. Du, Container shipping operational risks: an overview of assessment and analysis, Maritime Policy & Management 49 (2) (2022) 279–299.

[3] J. R. Gordon, P.-M. Lee, H. C. Lucas Jr, A resource-based view of competitive advantage at the port of singapore, The Journal of Strategic Information Systems 14 (1) (2005) 69–86.

[4] J. Chen, R. Bai, H. Dong, R. Qu, G. Kendall, A dynamic truck dispatching problem in marine container terminal, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2016, pp. 1–8.

[5] X. Chen, B. Feiyang, R. Qu, D. Jing, R. Bai, Neural network assisted genetic programming in dynamic container port truck dispatching, in: 2023 IEEE International Conference on Intelligent Transportation Systems (ITSC), IEEE, 2023, pp. 1–6.

[6] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, R. Qu, Hyper-heuristics: A survey of the state of the art, Journal of the Operational Research Society 64 (2013) 1695–1724.

[7] Y. Zhang, R. Bai, R. Qu, C. Tu, J. Jin, A deep reinforcement learning based hyper-heuristic for combinatorial optimisation with uncertainties, European Journal of Operational Research 300 (2) (2022) 418–427.

[8] X. Chen, R. Bai, R. Qu, H. Dong, J. Chen, A data-driven genetic programming heuristic for real-world dynamic seaport container terminal truck dispatching, in: 2020 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2020, pp. 1–8.

[9] X. Chen, R. Bai, R. Qu, H. Dong, Cooperative double-layer genetic programming hyper-heuristic for online container terminal truck dispatching, IEEE Transactions on Evolutionary Computation.

[10] D. Kizilay, D. T. Eliiyi, A comprehensive review of quay crane scheduling, yard operations and integrations thereof in container terminals, Flexible Services and Manufacturing Journal 33 (1) (2021) 1–42.

[11] A. Ramírez-Nafarrate, R. G. González-Ramírez, N. R. Smith, R. Guerra-Olivares, S. Voß, Impact on yard efficiency of a truck appointment system for a port terminal, Annals of Operations Research 258 (2017) 195–216.

[12] V. D. Nguyen, K. H. Kim, A dispatching method for automated lifting vehicles in automated port container terminals, Computers & Industrial Engineering 56 (3) (2009) 1002–1020.

[13] H.-A. Lu, J.-Y. Jeng, Modeling and solution for yard truck dispatch planning at container terminal, in: Operations Research Proceedings 2005, Springer, 2006, pp. 117–122.

[14] Y.-L. Cheng, H.-C. Sen, K. Natarajan, C.-P. Teo, K.-C. Tan, Dispatching automated guided vehicles in a container terminal, in: Supply chain optimization, Springer, 2005, pp. 355–389.

[15] H. R. Choi, B. K. Park, J. Lee, C. Park, Dispatching of container trucks using genetic algorithm, in: The 4th International Conference on Interaction Sciences, IEEE, 2011, pp. 146–151.

[16] S. Desale, A. Rasool, S. Andhale, P. Rane, Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey, Int. J. Comput. Eng. Res. Trends 351 (5) (2015) 2349–7084.

[17] J. Mohan, K. Lanka, A. N. Rao, A review of dynamic job shop scheduling techniques, Procedia Manufacturing 30 (2019) 34–39.

[18] M. Sánchez, J. M. Cruz-Duarte, J. carlos Ortíz-Bayliss, H. Ceballos, H. Terashima-Marin, I. Amaya, A systematic review of hyper-heuristics on combinatorial optimization problems, IEEE Access 8 (2020) 128068–128095.

[19] F. Garza-Santisteban, R. Sánchez-Pámanes, L. A. Puente-Rodríguez, I. Amaya, J. C. Ortiz-Bayliss, S. Conant-Pablos, H. Terashima-Marin, A simulated annealing hyper-heuristic for job shop scheduling problems, in: 2019 IEEE congress on evolutionary computation (CEC), IEEE, 2019, pp. 57–64.

[20] L. N. Ahmed, E. Özcan, A. Kheiri, Solving high school timetabling problems worldwide using selection hyper-heuristics, Expert Systems with Applications 42 (13) (2015) 5463–5471.

[21] R. Bai, J. Blazewicz, E. K. Burke, G. Kendall, B. McCollum, A simulated annealing hyper-heuristic methodology for flexible decision support, 4OR 10 (2012) 43–66.

[22] N. R. Sabar, M. Ayob, G. Kendall, R. Qu, Grammatical evolution hyper-heuristic for combinatorial optimization problems, IEEE Transactions on Evolutionary Computation 17 (6) (2013) 840–861.

[23] J. R. Koza, Genetic programming as a means for programming computers by natural selection, Statistics and computing 4 (1994) 87–112.

[24] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, M. Zhang, Instance rotation based surrogate in genetic programming with brood recombination for dynamic job shop scheduling, IEEE Transactions on Evolutionary Computation.

[25] Y. Jin, J. Branke, Evolutionary optimization in uncertain environments-a survey, IEEE Transactions on evolutionary computation 9 (3) (2005) 303–317.

[26] Z. Fan, Z. Wang, W. Li, X. Zhu, B. Hu, A.-M. Zou, W. Bao, M. Gu, Z. Hao, Y. Jin, Automated pattern generation for swarm robots using constrained multi-objective genetic programming, Swarm and Evolutionary Computation (2023) 101337.

[27] Y. Mei, Q. Chen, A. Lensen, B. Xue, M. Zhang, Explainable artificial intelligence by genetic programming: A survey, IEEE Transactions on Evolutionary Computation.

[28] S. Nguyen, Y. Mei, M. Zhang, Genetic programming for production scheduling: a survey with a unified framework, Complex & Intelligent Systems 3 (1) (2017) 41–66.

[29] M. T. Ahvanooey, Q. Li, M. Wu, S. Wang, A survey of genetic programming and its applications, KSII Transactions on Internet and Information Systems (TIIS) 13 (4) (2019) 1765–1794.

[30] L. I. Kuncheva, Combining pattern classifiers: methods and algorithms, John Wiley & Sons, 2014.

[31] R. Polikar, Ensemble based systems in decision making, IEEE Circuits and systems magazine 6 (3) (2006) 21–45.

[32] K. Theodorakos, O. M. Agudelo, J. Schreurs, J. A. Suykens, B. De Moor, Island transpeciation: A co-evolutionary neural architecture search, applied to country-scale air-quality forecasting, IEEE Transactions on Evolutionary Computation.

[33] H. Zhang, A. Zhou, H. Zhang, An evolutionary forest for regression, IEEE Transactions on Evolutionary Computation 26 (4) (2021) 735–749.

[34] F. Ecer, S. Ardabili, S. S. Band, A. Mosavi, Training multilayer perceptron with genetic algorithms and particle swarm optimization for modeling stock price index prediction, Entropy 22 (11) (2020) 1239.

[35] W. G. Jackson, E. Özcan, J. H. Drake, Late acceptance-based selection hyper-heuristics for cross-domain heuristic search, in: 2013 13th UK Workshop on Computational Intelligence (UKCI), IEEE, 2013, pp. 228–235.

[36] W. B. Yates, E. C. Keedwell, Offline learning for selection hyper-heuristics with elman networks, in: Artificial Evolution: 13th International Conference, Évolution Artificielle, EA 2017, Paris, France, October 25–27, 2017, Revised Selected Papers 13, Springer, 2018, pp. 217–230.

[37] M. L. Minsky, Theory of neural-analog reinforcement systems and its application to the brain-model problem, Princeton University, 1954.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, M. Riedmiller, Playing atari with deep reinforcement learning, arXiv preprint arXiv:1312.5602.

[39] C. D. Hubbs, H. D. Perez, O. Sarwar, N. V. Sahinidis, I. E. Grossmann, J. M. Wassick, Or-gym: A reinforcement learning library for operations research problems, arXiv preprint arXiv:2008.06319.

[40] D. Zeng, L. Gu, S. Pan, J. Cai, S. Guo, Resource management at the network edge: A deep reinforcement learning approach, IEEE Network 33 (3) (2019) 26–33.

[41] R. Emuna, A. Borowsky, A. Biess, Deep reinforcement learning for human-like driving policies in collision avoidance tasks of self-driving cars, arXiv preprint arXiv:2006.04218.

[42] H. Khorasgani, H. Wang, C. Gupta, Challenges of applying deep reinforcement learning in dynamic dispatching, arXiv preprint arXiv:2011.05570.

[43] J. C. Chu, S. Yan, K.-L. Chen, Optimization of earth recycling and dump truck dispatching, Computers & Industrial Engineering 62 (1) (2012) 108–118.

[44] H. Van Hasselt, A. Guez, D. Silver, Deep reinforcement learning with double q-learning, in: Proceedings of the AAAI conference on artificial intelligence, Vol. 30, 2016.

[45] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, arXiv preprint arXiv:1707.06347.