Advancing Container Port Traffic Simulation: A Data-Driven Machine Learning Approach in Sparse Data Environments

Xinan Chen[a] (xinan.chen@xjtlu.edu.cn), Rong Qu[c] (rong.qu@nottingham.ac.uk), Jing Dong[d] (jd704@cam.ac.uk), Haibo Dong[b] (nick.dong@nottingham.edu.cn), Ruibin Bai[b,*] (ruibin.bai@nottingham.edu.cn)

[a] International Business School Suzhou, Xi'an Jiaotong-Liverpool University, Suzhou, China

[b] Digital Port Technologies Lab, School of Computer Science, University of Nottingham Ningbo China, Ningbo, China

[c] School of Computer Science, University of Nottingham, Nottingham, UK

[d] Department of Engineering, University of Cambridge, Cambridge, UK

Corresponding Author:

Ruibin Bai

School of Computer Science, University of Nottingham Ningbo China, China

Tel: (0086) 574 8818000 (ext 8278)

Email: ruibin.bai@nottingham.edu.cn

Advancing Container Port Traffic Simulation: A Data-Driven Machine Learning Approach in Sparse Data Environments

Xinan Chen[a], Rong Qu[b], Jing Dong[c], Haibo Dong[d], Ruibin Bai[d,*]

[a]*International Business School Suzhou, Xi'an Jiaotong-Liverpool University, Suzhou, China*
[b]*School of Computer Science, University of Nottingham, Nottingham, UK*
[c]*Department of Engineering, University of Cambridge, Cambridge, UK*
[d]*Digital Port Technologies Lab, School of Computer Science, University of Nottingham Ningbo China, Ningbo, China*

Abstract

Efficient truck dispatching strategies are paramount in container terminal operations. The quality of these strategies heavily relies on accurate and expedient simulations, which provide a crucial platform for training and evaluating dispatching algorithms. In this study, we introduce data-driven machine learning methods to enhance container port truck dispatching simulation accuracy. These methods effectively surrogate the intersections within the simulation, thereby increasing the accuracy of simulated outcomes without imposing significant computational overhead in sparse data environments. We incorporate three data-driven learning methods: genetic programming (GP), reinforcement learning (RL), and a GP and RL hybrid heuristic (GPRL-H) approach. The GPRL-H method proved the most efficacious through a detailed comparative study, striking an effective balance between simulation accuracy and computational efficiency. It reduced the error rate of simulation from approximately 35% to about 7%, while also halving the simulation time compared to the RL-based method. Our proposed method also does not rely on precise Global Positioning System (GPS) data to simulate truck operations within a port accurately.

*Corresponding author.
Email addresses: `xinan.chen@xjtlu.edu.cn` (Xinan Chen), `rong.qu@nottinahm.ac.uk` (Rong Qu), `jd704@cam.ac.uk` (Jing Dong), `nick.dong@nottingham.edu.cn` (Haibo Dong), `ruibin.bai@nottingham.edu.cn` (Ruibin Bai)

Demonstrating robustness and adaptability, this approach holds promise for extending beyond port operations to improve the simulation accuracy of vehicle operations in various scenarios characterized by sparse data.
Keywords: Intelligent Intersection, Transport Simulation, Reinforcement Learning, Genetic Programming, Port Optimization

1. Introduction

The importance of container shipping in world trade cannot be overstated, particularly in this era of globalization (Chuang et al., 2010). Container shipping volumes have been increasing at an unprecedented pace, challenging the capacities of container ports to keep up the rapid growth(Notteboom, 2016). In response, container ports constantly strive to improve turnover efficiency, aiming to accommodate more ships and optimize their operations.

A critical factor limiting ports' efficiency is the truck dispatching strategy. Container transport operations are largely dependent on container trucks; as such, the development of intelligent truck dispatching algorithms has been a priority for many port companies in their quests to boost efficiency (He et al., 2015). In this context, simulators play a crucial role. They are instrumental in the training and evaluation of dispatching algorithms, and their performance directly influences the efficacy of the resultant strategies.

However, we observed that regular event-based simulations (Hassan, 1993), which overlook some specific route/junction related regulations, often yield inflated performance estimates. Our comparative analysis of real port operation data and simulated results revealed a key discrepancy: the time consumed by trucks at intersections, typically ignored in event-based simulations, was significantly underestimated. In these simulations, trucks are moved from one crane to another, disregarding potential delays at intersections.

Initially, a time-stepped simulation was developed to track truck actions and interactions at every time step (Fig. 1). This approach facilitates comprehensive tracking of trucks' positions, resulting in more accurate performance

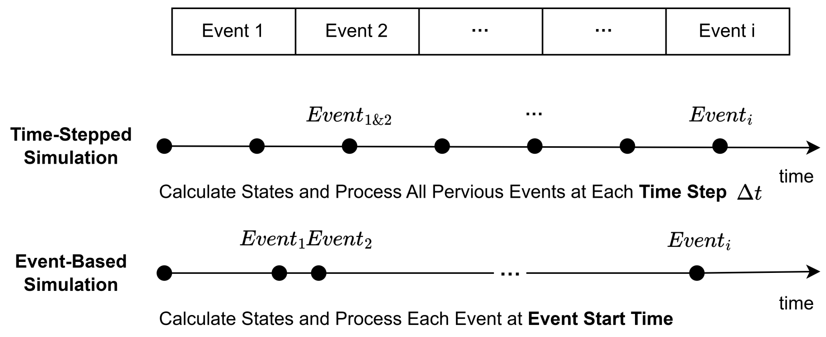
Event 1
Event 2
···
···
Event i
Time-Stepped Simulation
$Event_{1\&2}$
···
$Event_i$
time
Calculate States and Process All Pervious Events at Each **Time Step** Δt
Event-Based Simulation
$Event_1 Event_2$
···
$Event_i$
time
Calculate States and Process Each Event at **Event Start Time**

Figure 1: The Difference between Time-Stepped and Event-Based Simulation

estimation. However, time-stepped simulations present a dilemma regarding the setting of the time step. If the time step is excessively large, the precision of the truck route simulation will be compromised. Conversely, if the time step is too small, the computational cost becomes prohibitively high (Ramirez & Belytschko, 1989). Our findings indicate that a time-stepped simulator with a one-second time step is approximately 200 times more computationally intensive than an event-based simulation in the context of container truck dispatching. The significant computational overhead primarily arises from the need to accurately assess a truck's passage through an intersection, requiring evaluation of the truck's collision relationship at each time step. This computational cost is particularly prohibitive when training auto-generated truck dispatching strategies, which often necessitate tens of thousands of simulation runs (Chen et al., 2020).

Several state-of-the-art methods have been introduced to apply machine learning techniques in analyzing vehicle behavior at junctions, resulting in accurate predictions (Bagdatli & Dokuz, 2021; Yu & Zhou, 2019; Morgan et al., 2019). However, much of the research in port truck simulation for dispatching or scheduling rule training still relies on traditional discrete-event methods or simulation software (Afrapoli et al., 2019; Chen et al., 2024; Wei et al., 2023). This prevailing approach often overlooks a deep analysis of simulator accuracy and truck behavior at junctions. Consequently, while trained models may perform well in simulation environments, their effectiveness in real-world scenarios is limited. Therefore, this paper pioneers the integration of machine learning-driven intersection nodes into an event-based simulation model. By doing so, it addresses the limitations of traditional simulations in container ports, particularly in neglecting the intricacies of truck travel between cranes (Dragović et al., 2017), thereby enhancing simulation accuracy. Importantly, this framework achieves these advancements without significantly increasing computational costs, rendering it suitable for seamless integration into truck dispatching or crane scheduling training in container ports.

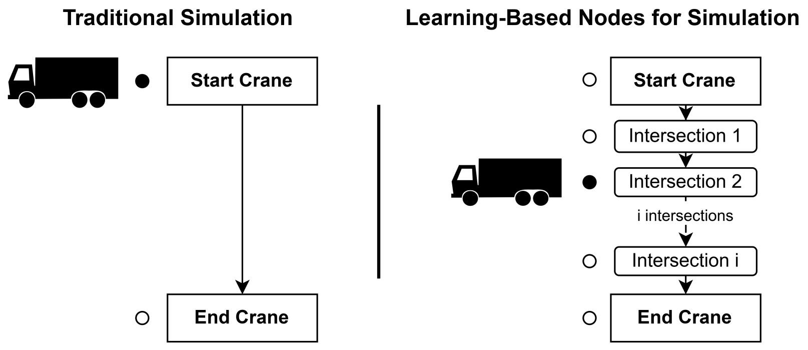

Figure 2: The Difference Between Traditional and Proposed Simulation

In the revised framework, the movement of trucks has been restructured to traverse node by node, departing from the conventional approach of moving directly from one crane to another (refer to Fig. 2). This enhancement entails deploying intelligent intersection nodes that autonomously compute the passage time of trucks at intersections. These computations are grounded in analyzing historical truck movement data and environmental conditions. A notable challenge in container port operations is the lack of reliable Global Positioning System (GPS) data for trucks. When such data is available, it often requires extensive pre-processing due to signal interference caused by the containers, which can obstruct satellite communication. Learning-based methodologies have been integrated into the simulation framework to address this issue. These methods are designed to deduce truck movements at intersections by leveraging existing operational data from the port, thereby compensating for the limitations of GPS data.

This paper compares two learning-based methods - Genetic Programming (GP) and Reinforcement Learning (RL) - to generate estimates of truck passing times at intersections. Our findings reveal that while RL outperforms GP in accuracy, it is also significantly more computationally intensive. To optimize this trade-off, we propose a hybrid method combining the advantages of GP and RL, augmented by an intersection importance analysis framework. Utilizing data insights, we rank all intersections' influence on simulation accuracy. Consequently, we designate the more critical intersections to be controlled by RL, while GP manages the remainder.

This innovative approach successfully strikes a balance between performance and computational cost. Furthermore, it enables us to generate more precise performance estimates and develop more efficient truck dispatching strategies. Importantly, our method can be readily applied to other transportation simulation challenges, significantly improving simulation accuracy even without detailed GPS location and other precise operation data.

The primary contributions of this paper are as follows:

- Introduction of machine learning-based intersection nodes in container port truck dispatching simulation, significantly improving simulation accuracy in sparse data environments.

- Proposal for integrating GP, RL, and a novel GPRL-H hybrid method to simulate truck actions at these intersections. This hybrid method achieves a preferable balance between simulation accuracy and computational efficiency, demonstrating robust performance even in sparse data scenarios.

- Design of timely rewards in traditional RL to effectively address the issue of sparse rewards in real-world simulation scenarios.

- Provision of experimental evidence showcasing the superior performance of the proposed methods.

The rest of this paper is organized as follows. Section 2 reviews related works and provides background information on the marine container terminal truck dispatching simulation problem. Section 3 introduces the dynamic truck dispatching and the simulation problem. The proposed learning-based methods are delineated in Section 4. Section 5 presents the experimental results, followed by a discussion in Section 6, which offers insights into the strong performance of our new framework. Finally, conclusions are drawn in Section 7.

2. Background and Literature Review

2.1. Container Port Truck Dispatching

A container port is a pivotal transfer station for importing and exporting goods in containers by shipping (Bonacich & Wilson, 2008). It functions primarily as an intermediary for container transfer between the ship and the hinterland. The container trucks, much like blood in the human body, facilitate smooth transformations of these operations. They pick up containers from quay cranes (QCs) and yard cranes (YCs) and transport them to the appropriate locations (Luo & Wu, 2020). Consequently, an efficient truck dispatching system is vital for the operation of ports, as all transportation activities rely heavily on trucks.

Increasingly, research is being directed towards improving the efficiency of truck dispatching (Bai et al., 2023). There are two primary strategies for truck dispatching - static truck dispatching and dynamic truck dispatching. While static dispatching calculates a pre-determined truck dispatching plan for a given period (Schulte et al., 2017), dynamic dispatching dispatches a truck as soon as one becomes idle (Poss & Raack, 2013). Despite the inability of the dynamic method to guarantee optimality, its flexibility and robustness make it highly favored in real port applications (Chen et al., 2022).

Regardless of the approach, both static and dynamic truck dispatching require the support of simulators for strategy evaluation and performance evolution (Moradi Afrapoli, 2019). This is particularly crucial for learning-based strategies, which demand tens of thousands of training and testing iterations in simulators for continuous performance improvement (Zhang et al., 2022). Therefore, a quick and accurate simulator is indispensable for training and testing strategies, enhancing truck dispatching efficiency in container terminals.

2.2. Truck Dispatching Simulation

Given the critical importance of simulation in optimizing container truck dispatching within ports, previous research has extensively utilized simulation-supported optimization methods to address this challenge (Mirzaei-Nasirabad et al., 2023; Tang et al., 2024; Hu et al., 2024). However, the predominant use of event-based simulations, while offering simplicity and speed, has neglected the complexities of road truck travel.

Event-based simulations estimate travel time by calculating distances between start and end positions and dividing them by fixed or fluctuating speeds (Juan et al., 2013). Unfortunately, these simulations overlook crucial actions during travel, particularly truck interactions at intersections, which significantly impact simulation accuracy (Arvin et al., 2020).

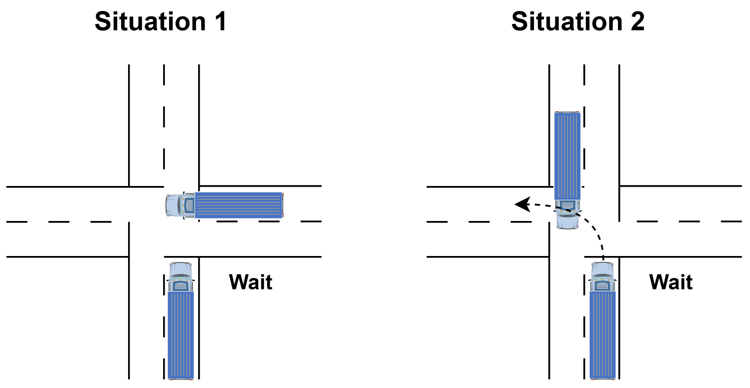

Figure 3: Examples of Intersection Actions

As depicted in Fig. 3, scenarios such as trucks waiting to cross intersections when others are present or approaching highlight the complexity. While these examples consider only two trucks, complexity amplifies with additional vehicles involved.

This complexity is pivotal in container port truck dispatching simulations and necessitates precise modeling of truck routes and actions at intersections. Ideally, time-stepped simulations would be employed to track truck statuses at each time point (Gould et al., 2007). However, few studies have adopted this approach due to its computational intensity and challenges in algorithm training and strategy optimization (Hu & Mohamed, 2013).

Several modified event-based methods have been proposed to simulate intersections, improving simulation accuracy while significantly reducing simulation time compared to time-stepped simulations (Xu et al., 2014). These methods are particularly suitable for training and evaluating optimization algorithms. However, they rely on accurate vehicle data at intersections to build intersection models (Harahap et al., 2020), posing challenges for application in environments like container ports where precise GPS data may be unavailable.

Thus, this paper proposes a learning-based approach to constructing intersection models for simulation using sparse data, aiming to accurately represent port traffic in scenarios where data availability is limited. Our method improves simulation accuracy without imposing significant computational overhead in sparse data environments. The following sections present two potential learning models within our framework.

2.3. Genetic Programming

Genetic Programming (GP), first introduced by Koza in the 1990s (Koza, 1994), has been widely recognized for its ability to generate solutions to complex problems automatically. It utilizes principles of natural evolution, particularly the notions of mutation, crossover, and selection, to evolve populations of solutions over time (Ahvanooey et al., 2019).

Over the years, researchers have successfully applied GP in many data-driven applications, demonstrating its capability to extract valuable knowledge from diverse data sets (Banzhaf et al., 1998; Bi et al., 2021). Notable examples include optimizing the control of autonomous vehicles (Ardeh et al., 2022), predicting financial market trends (Christodoulaki et al., 2022), classifying images (Fan et al., 2022), and dynamic truck dispatching (Chen et al., 2019).

In container terminal simulation, despite the absence of precise GPS data for direct learning of truck movements at intersections, GP capitalizes on existing data as a reference point, persistently probing intersection passing rules during its evolutionary progression to ascertain the correct protocols. Throughout this evolution, GP operates independently of prior knowledge of either the rule or the intersection passing data, instead utilizing the ultimate truck traveling time as a guiding estimator to steer it towards accurate prediction (Elhenawy et al., 2014). This inherent flexibility positions GP as an appealing choice for rule generation within our proposed data-driven, learning-based intersection nodes.

This choice aligns with our broader objective: to optimize the efficiency and accuracy of truck dispatching simulations while minimizing computational costs. By leveraging the power of GP, we can effectively model intersection behaviors in event-based simulation, advancing the precision of container port truck dispatching strategies.

2.4. Reinforcement Learning

Reinforcement learning (RL), a significant subfield of machine learning, focuses on how an agent should act in an environment to maximize a cumulative reward (Sutton et al., 1998). This concept originates from behavioral psychology and has gained immense popularity in recent decades due to its successful applications across various complex problems.

Since its inception, RL has evolved significantly with the advent of deep reinforcement learning (DRL), which integrates deep learning and RL. DRL has been instrumental in solving numerous high-dimensional problems that were traditionally challenging for ordinary RL (Arulkumaran et al., 2017). Noteworthy applications of RL and DRL span various domains, such as traffic controlling (Lee et al., 2020), metaheuristic designing (Yi et al., 2022), and dynamic resource allocation (Yu et al., 2021).

Diverging from conventional supervised learning methods, which typically predict vehicle behavior at intersections based on models trained with real vehicle intersection passing data (Shirazi & Morris, 2017), RL brings an innovative approach to the table. Initially, RL hypothesizes a rule for truck passing behavior, given the current state (Yung & Ye, 1999). This tentative rule is then iteratively adjusted via a reward mechanism, which, in this context, is the real travel time of the truck. This continuous feedback loop guides RL in refining the predicted rule until it culminates in a highly accurate model for estimating truck passing times.

Given the substantial learning capabilities of RL, we elected to employ it as a robust contender in predicting actions at intersections within our simulations. However, even though RL has so many successful applications in truck dispatching (Jin et al., 2023), berth allocation (Lv et al., 2024), and ship traffic scheduling (Zhang et al., 2024), it has rarely been used in boosting simulation accuracy. All the performance results of these results of these dispatching are based on the simulation. If the simulation is incorrect, how can we trust the results? RL offers more than merely the potential for accurate simulation of intersection behaviors; it also provides an avenue for revealing novel, data-driven insights into truck dispatching. This ability, in turn, can significantly enhance the simulation accuracy and decision-making process in container port truck dispatching.

Recognizing the significant capabilities of RL in predictive modeling, we chose to integrate RL into our simulation framework as a robust method for predicting actions at intersections. While RL has found numerous successful applications in truck dispatching (Jin et al., 2023; de Carvalho & Dimitrakopoulos, 2021), jobshop scheduling (Xu et al., 2024) and vehicle routing (Hildebrandt et al., 2023; Koh et al., 2020), its use in enhancing simulation accuracy, particularly in the context of container port truck dispatching, remains relatively unexplored. This gap presents a notable opportunity, as the performance of dispatching and routing strategies heavily relies on the accuracy of underlying simulations. With RL, there is the potential for precise simulation of intersection behaviors and for uncovering novel, data-driven insights into truck dispatching.

These insights can significantly refine the decision-making process in container port management, leading to more efficient and effective strategies. The capability of RL to transcend mere accuracy in simulation and contribute to deeper understanding and optimization of dispatching processes underscores its value in our simulation model.

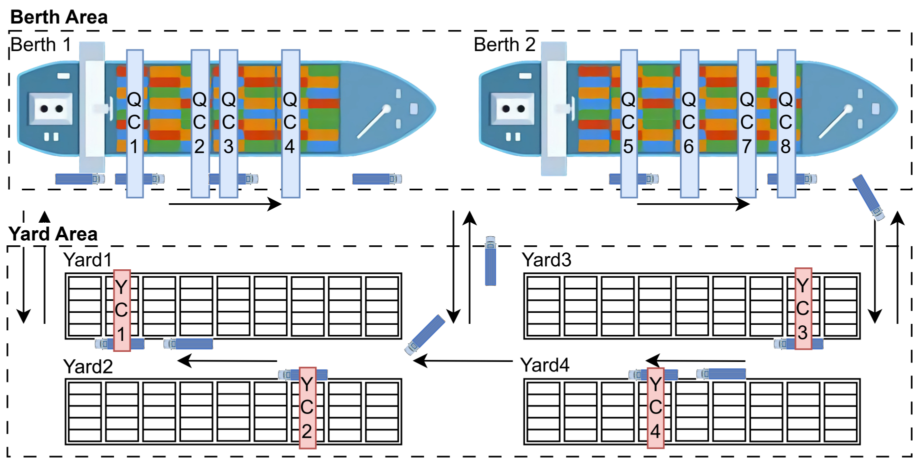

Figure 4: Sample Map of Container Port

3. Problem Description

The primary aim of a container port is to maximize the number of ships served within a given time period, thereby increasing the port's turnover efficiency. As illustrated in Fig. 4, container ports comprise two significant areas – the berths and the yards – linked by container trucks.

Fundamentally, there are two primary operations within a container terminal: loading and unloading containers onto or from ships at the berths and to or from the containers in the yards with the help of container trucks. There are two prevalent container sizes: the small Twenty-Foot Equivalent Unit (TEU) and the large container, equivalent to two TEUs. Each truck can carry one large container or two small containers. Typically, a small container is bundled with another small container to form a standard `task`, while a large container is treated as a task. However, in specific scenarios, dynamic truck dispatching algorithms may be necessitated to dispatch two separate small containers when possible.

Throughout loading and unloading operations, trucks convey containers between yards and ships. QCs are tasked with loading or unloading containers from and onto ships, while YCs manage the yard operations. QCs can handle one large container or two small containers in one action, whereas YCs can only operate one small or large container due to differences in clamp types. The fact that QCs and YCs can only operate containers from one truck at a time results in waiting periods and congestion under these cranes. Moreover, as illustrated by the passage direction arrows in Fig 4, trucks must comply with the port's traffic regulations. Certain routes permit bidirectional traffic, whereas others allow for unidirectional movement only. This regulatory structure is critical for maintaining a safe and efficient circulation of trucks within the port.

Table 1: Example of Work Instructions

ID	**ContainerID**	**Src**	**Dst**	**Type**	**TEUs**	**Ton**	**TwinID**
1731	FCIU3705890	CR12	J4	DSCH	2	17	0
1287	ECMU9249162	Q2	CR1	LOAD	1	23	1514
137	NYKU2797417	Q5	CR7	LOAD	2	15	0
1514	TCLU5546292	CR12	CR15	DSCH	1	19	1287

All container operations adhere to pre-designed work instructions (tasks) as outlined in Table 1. Each task is identified by an 'ID', and 'ContainerID' represents the unique identifier for each container involved. 'Src' and 'Dst' denote the container's source and destination locations, respectively. Tasks are categorized into 'DSCH' for unloading from ship to yard and 'LOAD' for loading from yard to ship. The size of the container is indicated by 'TEUs', while 'Ton' represents its weight. Typically, containers are either 40-foot or 20-foot, allowing each truck to carry either two small or one large container.

To accommodate this, small containers are paired to create a combined 'bind task', identified by 'TwinID'. Each task contains unique information and must be executed in sequence to maintain a balanced ship's allotment. However, for unloading tasks, the sequence can be swapped within a certain number (sn) to optimize operations.

Considering the crucial role of maintaining continuous operation of the QCs for optimal ship operation efficiency, the QC waiting time (QCW) (Chen et al., 2016) has emerged as a vital metric for evaluating dispatching strategies. Let s_{ij} and e_{ij} denote the start and end times, respectively, for task j of QC i, with n_i representing the total number of tasks for QC i and m being the number of QCs. The total QC waiting time can be represented in Equation (1). Another important metric is the number of TEUs processed per hour (TEU/h, TPH) (Lubulwa et al., 2010) for the entire port. This metric directly impacts the ship docking time and overall turnover efficiency of the port. Let n be the total number of tasks, $size_i$ represent the size of task i, E denote the end times for the task set, and S represent the start times for the task set. The TPH can then be expressed in Equation (2).

These two metrics are paramount for port companies' operational endeavors and are widely adopted across numerous research focusing on optimizing port operations. Consequently, in our paper, we consider the computational error rate of these two metrics as the primary benchmark for comparing the performance of all algorithms. By aligning our evaluation with the industry-standard metrics, we ensure the relevance and applicability of our findings to real-world port operations.

$$QCW = \sum_{i=1}^{m} \sum_{j=2}^{n} s_{ij} - e_{i(j-1)} \tag{1}$$

$$TPH = \frac{\sum_{i=1}^{n} size_i}{\max E - \min S} \tag{2}$$

Given the formidable challenges of testing dispatching strategies in real-world scenarios, most research efforts have turned to surrogate simulators.

These replicate the port operation process and train and test dispatching algorithms (Jackson et al., 2024; Raza et al., 2024; Sarmiento et al., 2019). Work instructions are typically input into these simulators, which simulate the entire task completion process. The dispatching algorithms within the simulator distribute trucks, subsequently outputting TPH and QCW values to evaluate algorithmic performance. Although many studies have demonstrated substantial improvements in these metrics, the development of their surrogate simulators frequently neglects the interaction between trucks during transit. Although it's customary to exclude ostensibly non-essential processes when constructing surrogate simulation models, this approach can introduce complications in specific applications, such as truck dispatching in container terminals.

Our tests at Ningbo Meishan Port highlighted that dispatching algorithms, trained using conventional event-based simulators, do not consider intersection conditions during truck dispatching. This omission can lead to inadequate dispatching decisions, resulting in an uneven distribution of trucks across QCs, with some experiencing a surplus and others a shortage. Such imbalances necessitate constant monitoring of QC statuses by port dispatching operators, who must make on-the-fly adjustments to dispatching plans to maintain operational efficiency within the port. It is clear from these findings that there is a need for a more sophisticated surrogate simulator to train and test dispatching algorithms effectively, ensuring that the strategies developed are both feasible and efficient in real-world container port settings.

Furthermore, as detailed in Section 5, our analysis revealed that normal event-based simulators often overestimate TPH and QCW values. Such miscalculations could lead to misjudgments of algorithm performance in real-world scenarios. Recognizing this limitation, we were motivated to develop an innovative truck dispatching simulator. By integrating a data-driven, learning-based approach, we aim to enhance simulation accuracy without significantly increasing computational overhead. In the subsequent sections, we outline the structure of our simulator and discuss the algorithms implemented to emulate truck movements at intersection nodes accurately.

4. Methodology

This section outlines our event-based container truck dispatching simulator framework, detailing our novel learning-based methods for simulation. We explore three algorithms: GP, RL, and a hybrid GP-RL method, focusing on their utility in simulating intersection node operations.

A significant challenge we faced was the lack of high-quality GPS data for trucks at intersections, as relayed by our industry collaborator, Ningbo Port. This limitation hindered our ability to establish accurate operational rules for intersections based on trucks' movement data. To overcome this, we utilized available data, including task timings, crane distances, truck positions, and truck routes, to train our learning-based models. This information, sourced from Radio-Frequency Identification (RFID) and Terminal Operating Systems (TOS), provides reliable and precise information on truck locations. Based on these accurate data, we built learning-based models to mimic additional data, thereby enhancing the simulation's accuracy.

Our experiments demonstrate that GP and RL algorithms significantly enhance simulation accuracy, with RL showing superior results but at higher computational costs. To optimize accuracy and efficiency, we developed the GPRL-H method, which strategically employs RL for critical nodes and GP for others. This approach ensures a balanced trade-off between computational demand and simulation precision.

4.1. Event-Based Simulator Framework

Event-based simulation is a form of discrete-event simulation (DES) that propels the simulation process by initiating sequences of events. In an event-based simulation, the simulation step corresponds to the time difference between occurrences of events, and the simulator proceeds through the simulation step by step in alignment with the event sequence.

The principal events in container port truck dispatching include container loading, container unloading, and truck movements. Leveraging the principles

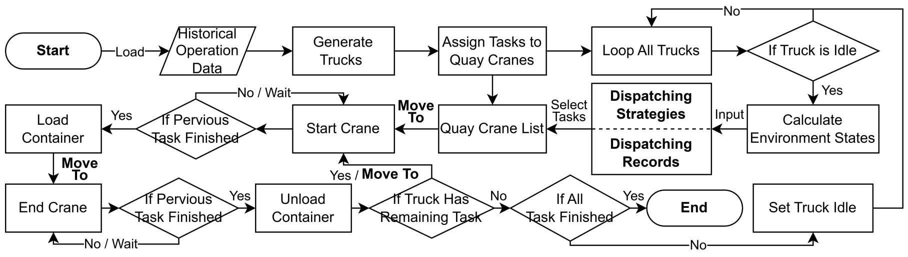

Figure 5: Event-based Port Simulator Flow Chart

of event-based simulation and the operation process of container port truck dispatching, we have developed a simulator as illustrated in Fig. 5. This simulator must accurately evaluate the efficiency of truck dispatching strategies. Often, accuracy is checked with historical data.

In its initial state, the simulator draws upon historical port operation data, including tasks, truck numbers, and current truck locations. It generates trucks at their given positions and assigns tasks to their respective QCs. Upon the commencement of the main simulation loop, the simulator scrutinizes all trucks, particularly idle ones.

For idle trucks, the simulator computes the environmental states and implements them into the dispatching strategies to determine the suitable tasks for the trucks. Nevertheless, as this paper's primary concern is enhancing simulation accuracy, we suggest an alternative approach. This technique employs dispatching records from historical operation data to maintain a fixed task assignment sequence, thus mitigating the influence of various dispatching strategies.

Following task assignment, the truck performs its designated task, including loading containers at a start crane and unloading them at an end crane. After completion, the truck either proceeds to the next task or returns to the idle state if no tasks remain. The simulator then checks whether all tasks have been completed to conclude the simulation.

In this process, a significant source of inaccuracy arises from truck movement, particularly within intersection nodes, where the timing can be hard to estimate accurately. Addressing this issue is key to improving the overall accuracy of the simulator, which is the central focus of our subsequent sections.

Upon the completion of the simulation, TPH and QCW metrics are readily calculable. It is important to acknowledge that in this conventional event-based simulator, the truck's journey is oversimplified - it directly transits from one crane to another, while the details of the travel process are mostly neglected. As we have explained, this simplification significantly compromises the simulation's accuracy. To rectify this, we propose a learning-based method for our event-based simulation, outlined in the following subsection.

4.2. Learning-Based Methods for Simulation

In our endeavor to simulate the truck movement process more precisely, we have segmented the traveling route of trucks, which typically extends from crane to crane. The route now comprises intermediary stops at intersection nodes, as depicted in Fig. 2. To elucidate the role of these intersection nodes in the simulation, we have drawn a logic map of the previous sample container port map as seen in Fig. 4.

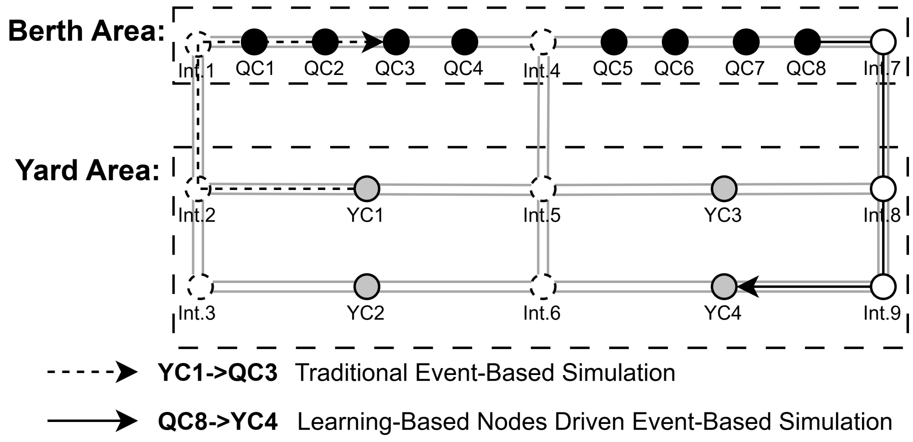
Berth Area:
Int.1
QC1
QC2
QC3
QC4
Int.4
QC5
QC6
QC7
QC8
Int.7
Yard Area:
Int.2
YC1
Int.5
YC3
Int.8
Int.3
YC2
Int.6
YC4
Int.9
YC1->QC3 Traditional Event-Based Simulation
QC8->YC4 Learning-Based Nodes Driven Event-Based Simulation

Figure 6: Example of Truck Routing in Logic Map

As illustrated in Fig. 6, the traditional event-based simulation calculates the truck movement path based on the actual path length but neglects intersections. This method is akin to a truck moving instantaneously from YC1 to QC3, bypassing all intermediate steps such as Intersection (Int.) 1 and 2. However, our proposed learning-based simulation method moves the truck from node to node. All nodes on the path, such as Int. 7, 8, and 9, are considered, as demonstrated in the figure. This segmented path can more accurately simulate truck movement, considering all actions and interactions at intersection nodes. For instance, a situation as depicted in Fig. 3, wherein situation 1 occurs at an intersection node along the truck's route. Upon the truck's arrival at this intersection node, the intelligent node autonomously assesses its current state. Drawing from this assessment, the node determines a passage time that accommodates the truck's waiting, deceleration, and acceleration phases. This comprehensive computation of passage time enables adjustments to truck speeds and accurately simulates truck actions at intersections. As a result, it significantly enhances the precision of the truck dispatching simulation.

The calculation incorporates the following state factors in this paper:

- The current number of trucks at the node.
- The maximum passage time of the current node trucks.
- The truck number coming from the left/right/up/down.
- The closest truck distance from the left/right/up/down.

The above description presents the truck operation function in Algorithm 1. In this function, in addition to a dispatchable truck list and a to be completed task list, *routing* is a procedure that calculates the shortest path from the start crane to the end crane of each task while respecting traffic regulations set by the port. To simplify the experimental process, the route from one crane to another is a pre-computed fixed path. For the path from QC8 to YC4, the traditional event-based simulation output merely the O-D pair [QC8, YC4] with a constant travel time, with truck interactions at intersections being overlooked. In our proposed learning-based simulation, however, the output path will be [QC8, Int.7, Int.8, Int.9, YC4]. When passing the intersection nodes, the function *n.passing*() will generate a passing time for this node by using the forecast model with the environmental states as inputs. Upon completing a

Algorithm 1 Truck Operation Function

```
Require: $truck, task, routing, env$
1: $truck.start\_task(task)$
2: $truck.path \leftarrow routing(truck.pos, task.start\_crane)$
3: for $n$ in $truck.path$ do
4:     $truck.wait(truck.travel(n))$
5:     $truck.pos \leftarrow n$
6:     if $n.type = crane$ then
7:         $truck.wait\_until(n.available)$
8:         $truck.wait(task.operate())$
9:         if $truck.pos = task.twin\_task.start\_crane$ or $truck.pos = task.twin\_task.end\_crane$ and $task.twin\_task$ then
10:            if $n.type \neq quay\_crane$ then
11:                $wait(task.twin\_task.operate())$
12:            end if
13:        end if
14:    else if $n.type = intersection$ then
15:        $truck.wait($**n.passing(**$env.states$**)**$)$
16:    end if
17:    $truck.leave(n)$
18: end for
19: $truck.end\_task(task)$
```

task at the designated location, the truck assesses whether it carries additional containers (*twin_tasks*, as a single truck can accommodate two small containers). If no further containers are to be loaded or unloaded, the truck proceeds to the unloading site to finalize its task after the loading is completed. From the perspective of the entire simulator workflow, the most challenging aspect to compute is the movement time of the trucks. This is influenced by varying congestion levels at different intersections during each transit, resulting in distinct travel times for the trucks. Consequently, accurately calculating the time trucks take to pass through each intersection has become a pivotal factor constraining the effectiveness of container truck dispatching simulations.

Despite the considerable advances in our simulation strategy, we still face a fundamental issue: how to use state factors to reliably predict the truck passing time to boost the precision of our simulations. Conventionally, we can access historical data elucidating the relationship between the truck's passing time and intersection status parameters. In that case, a forecast model can be built to leverage this rule to calculate passing time. Moreover, it is possible to extract this correlational data from detailed truck GPS information to build a deeper understanding.

Container ports present a significant challenge due to a scarcity of high-quality truck GPS data. This dearth of reliable information impedes our ability to identify hidden relationships between various operational parameters. Despite substantial investments in real time kinematic (RTK) devices and numerous engineering efforts, the GPS-unfriendly port environment still yields a high proportion of erroneous positioning data. In the absence of precise GPS data for trucks and information on the queuing wait time at each crane, the accurate replication of the trucks' historical route poses a substantial challenge.

We propose a learning-based method that estimates intersection passing times using a reward-based system to address this issue. This approach facilitates the discovery of traffic flow rules, even in contexts where operational data might be sparse or limited. Learning-based methods like GP and RL do not rely on explicit programming or hand-crafted rules. Instead, they learn from available data, iterating over numerous cycles, making and learning from errors, and progressively improving their predictions. The central mechanism is a reward-based system: the more a prediction or model's action aligns with actual outcomes, the higher the reward. The algorithms strive to maximize these rewards, gradually improving their performance despite limited data.

GP and RL are particularly suited to the task at hand due to their flexibility, ability to manage complex, non-linear relationships, and resilience against noisy or sparse data. These methods are not only trained on previous historical data but can also uncover hidden insights through interaction with the simulation environment, countering the drawbacks of insufficient data. These attributes make them ideal for enhancing the accuracy of our truck dispatching simulator. In the subsequent sections, we provide detailed insights into the selected algorithms and learning methods, highlighting their contribution to accurately predicting intersection crossing times despite data limitations.

4.2.1. Data-driven Genetic Programming

GP is an evolutionary computation method inspired by the principles of biological evolution. Its essence lies in refining a population of solutions through consistent modifications of crossover and mutations. Hence, by designing a fitness function that encourages the generated individuals to provide truck intersection passing times that match the historical data best, we can allow the model to learn hidden rules through its evolution.

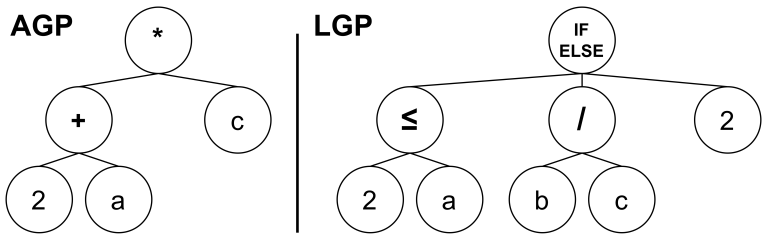

Figure 7: AGP and LGP Structure

GP employs various representations, but the tree structure is the most common, easy to understand, and adopted in this study. As depicted in Fig. 7, GP can be divided into two types: arithmetic genetic programming (AGP) and logical genetic programming (LGP), contingent upon whether logical operators are included or not.

AGP utilizes addition, subtraction, multiplication, and protected division operations in this paper. Meanwhile, LGP integrates additional logic operators, including greater than or equal to, less than or equal to, if-else, and, or, maximum, and minimum. The terminals in GP are the state factors enumerated in the previous subsection, amounting to 10 state factors plus one random constant.

In the context of the learning-based simulation, GP individuals act as the $n.passing()$ function to compute the passing time of trucks at intersections. To train these GP individuals, the fitness function is set up according to Equation (3), which combines the metrics detailed in Equations (1) and (2).

$$\begin{aligned} R_f = \delta \sum_{i=1}^{m} \sum_{j=2}^{n} |s_{ij} - e_{i(j-1)} - s'_{ij} + e'_{i(j-1)}| \\ + |\max E - \min S - \max E' + \max S'| \end{aligned} \tag{3}$$

In the equation above, S/s and E/e represent the task start and end times observed in actual operations, respectively, analyzed collectively and individually. Correspondingly, S'/s' and E'/e' symbolize the same times in the simulated results.

The fitness function, denoted as R_f, measures the disparity between the actual data from operational records and the simulation outcomes. This function focuses on two main metrics: TPH and QCW, as observed in the training dataset. By gauging the performance of GP individuals, this fitness function aids in pinpointing the entities possessing accurate knowledge necessary for estimating truck intersection passing times. As a result, the simulation developed is highly reflective of actual operational data. The parameter δ is implemented to balance the relative importance of the two metrics, and for this study, it is set to 1. It's important to note that in this paper, we don't distinguish between different intersections; we instead employ a single model to fit all intersections.

This approach is consistent across both GP and RL methods.

The GP evolution process, as outlined in Algorithm 2, continually uses genetic operations such as crossover and mutation methods to produce new individuals. This method screens out and eliminates individuals with low fitness, allowing the entire population to evolve toward individuals with higher fitness. This paper employs tournament selection for better control on the selection pressure. The crossover and mutation methods align with those mentioned in our previous work (Chen et al., 2022). Notably, since the fitness of GP signifies the disparity between metric values of real data and simulated results, lower fitness indicates higher simulation accuracy.

Algorithm 2 AGP, LGP Evolution Function

```
Require: Initial Parameters $initial$
  $p \leftarrow NewPopulation$
  $p.initial\_individuals(initial.population\_size)$
  $generation \leftarrow 0$
  while $generation < initial.max\_generation$ do
      $p.calculate\_fitness()$
      $p.penalize\_long\_individuals()$
      $next\_generation \leftarrow NewPopulation$
      while $next\_generation.size() < p.size()$ do
          Insert an $individual$ to $next\_generation$ by
          Crossover, Mutation, or Reproduction in $p$
      end while
      $p \leftarrow next\_generation$
      $generation \leftarrow generation + 1$
  end while
```

While our experimental results indicate that both AGP and LGP can significantly improve the accuracy of simulations, they have a key limitation: the fitness function of traditional GP can only be calculated upon completion of the entire simulation process. This delay means that the valuable data generated during the simulation cannot be fully exploited to optimize the accuracy of simulations in real time.

In light of these considerations, we believe that leveraging RL to learn the truck's passing time at intersections could greatly enhance the efficiency and accuracy of our simulations. Using RL, we can fully utilize the historical data and acquire more precise and informative insights in real time, ultimately leading to more accurate simulations. This RL-based approach will be discussed in detail in the following subsection.

4.2.2. Reinforcement Learning

RL is a machine learning algorithm that trains an agent to select an appropriate action to obtain maximum rewards. Unlike GP, which relies on evolutionary principles to refine its predictive models, RL leverages its ongoing interaction history, often via mechanisms like experience replay, to progressively enhance its decision-making policies. In the traditional GP context, R_f is calculated as a reward (fitness) after completing all task simulations, providing limited feedback to the learning process. In this environment, the trajectory-based experiences gathered by the agent do not offer substantial information about the quality of the action, which in turn considerably impacts the quality of the final simulation results.

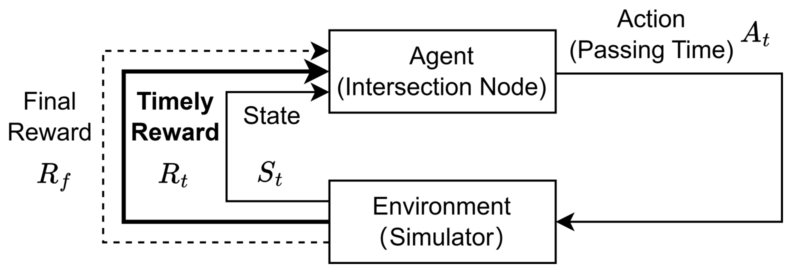

Figure 8: RL Structure

However, RL introduces a notable paradigm shift in the learning approach compared to GP. Rather than being completely disconnected from the environment during the evolutionary process, as with GP, RL allows the agent to interact continually with the environment. The agent performs actions and receives feedback in the form of rewards, as depicted in Fig. 8. This dynamic interaction facilitates a more adaptive learning process and can help to optimize the actions more effectively. Specific aspects of this feedback mechanism, including its immediacy and influence on the learning process, will be explored in detail in the results section.

Thus, we introduce a timely reward, R_t, formulated as per Equation (4) in the RL method. The symbols s_{ij} and e_{ij} represent the start and end times in the actual data for the jth task of the ith QC, respectively, while s'_{ij} and e'_{ij} denote the start and end times in the simulation. The reward R_t is calculated as the discrepancy between each task's actual and simulated truck movement times. As with the reward R_f, smaller R_t values signify a higher simulation accuracy.

$$R_t = |E_{ij} - S_{ij} - E'_{ij} + S'_{ij}| \tag{4}$$

The novel aspect of our RL approach is the concurrent use of the real-time computed reward R_t and the episode-end reward R_f to guide the learning process. This combined guidance allows the RL agent to learn the logic of truck operations at the intersection more effectively from historical data. By incorporating the real-time computed reward R_t following each action A_t by the RL agent, we can better use the previously ignored data on the truck movement time for each task. Consequently, the agent can acquire more information, improving the quality of actions and, ultimately, a more accurate simulation.

In our experiment on the test dataset, we ran 10 iterations and trained for 500 episodes to assess the impact of different rewards on RL performance. As illustrated in Fig. 9, using the traditional final reward approach did not lead to effective convergence. RL failed to acquire helpful knowledge, resulting in a high simulation error rate. Conversely, our proposed combination of two types of rewards significantly improved RL's convergence. This approach enabled RL to converge on complex problems, overcoming challenges associated with

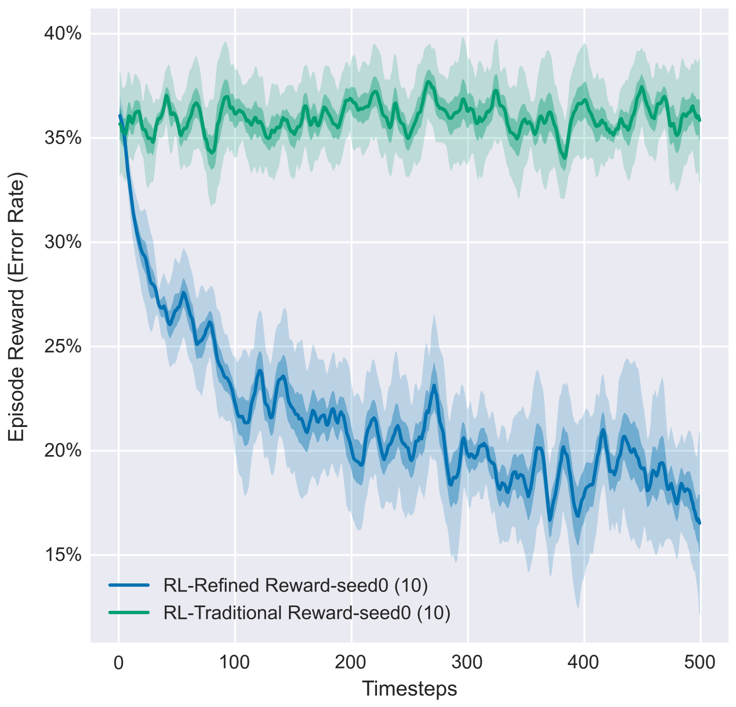

Figure 9: Performance of RL with Different Rewards

RL in intersection simulations and substantially increasing the accuracy of the simulations.

In the context of this research, RL is expected to output the truck crossing time at an intersection as a continuous variable. This requirement presents a challenge to traditional RL methods such as deep Q-learning (Mnih et al., 2013) (DQN) and double deep Q-learning (Van Hasselt et al., 2016) (DDQN), which are not optimized for handling continuous action spaces effectively. We conducted a comparison of various deep reinforcement learning models. From this comparison, we found that the policy optimization (PPO) method achieved the best balance between performance and simulation time. Therefore, we have adopted it to calculate the intersection's truck crossing time.

PPO is a policy-based optimization algorithm introduced in 2017 (Schulman et al., 2017). It is designed to tackle the challenges posed by earlier algorithms, such as trust region policy optimization (Schulman et al., 2015) (TRPO) and asynchronous advantage actor-critic (Mnih et al., 2016) (A3C). PPO enhances sample efficiency and stability by employing a trust region approach and a clipped objective function, as depicted in Equation (5).

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min(r_t(\theta)\hat{A}_t, \mathrm{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t)\right] \tag{5}$$

In this equation, $r_t(\theta)$ represents the probability ratio between the current policy and the old policy, given by $\frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$. The variable $\hat{A}_t$ indicates the estimated advantage function at time step t, while ϵ is a hyperparameter that defines the degree of trust region in the policy update.

The RL training environment used in this study is similar to the one for the port dispatch simulator previously discussed. Like in the Genetic Programming (GP) method, the RL agent performs the function *n.passing*() in Algorithm 1, producing a continuous number that represents the time a truck took to cross the intersection, given the current environment parameters. The 10 state parameters described in Section 4.1 define the environment state.

During training, the simulator provides the agent with intersection state information S_t, and the agent outputs the predicted intersection passage time based on its learned knowledge. Each time the agent makes an action A_t, the simulator returns a reward R_t. Once all actions are completed, a final reward R_f is computed. The entire training process of the PPO-based RL algorithm is illustrated in Algorithm 3.

Algorithm 3 Proximal Policy Optimization (PPO) Training

```
1: Initialize policy parameters $\theta$ and value function parameters $\phi$
2: for each iteration do
3:     Collect a set of trajectories $\tau$ using the current policy $\pi_\theta$
4:     Compute the timely reword $R_t$ for each time step in trajectories
5:     if Simulation finished then
6:         Compute the final reword $R_t$
7:     end if
8:     Compute advantage estimates $A_t$ using value function $V_\phi$
9:     for each optimization epoch do
10:        for each time step $t$ in trajectories do
11:            Compute probability ratio $\rho_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)}$
12:            Compute surrogate objective $L_t(\theta) = \min(\rho_t(\theta)A_t, \text{clip}(\rho_t(\theta), 1 - \epsilon, 1 + \epsilon)A_t)$
13:            Perform gradient ascent on $\theta$ to maximize $E_t[L_t(\theta)]$
14:            Update value function parameters $\phi$ by minimizing the value loss
15:        end for
16:    end for
17:    Update policy $\pi_{\theta_{\text{old}}} \leftarrow \pi_\theta$
18: end for
```

Experimental results indicate that RL demonstrates superior simulation accuracy compared to GP. This improvement is largely attributed to the inclusion of the timely reward R_t, which provides RL agents with an immediate response to their actions, leading to more refined decisions. However, this added precision has a significant drawback: increased computational cost. Our experiments show that utilizing RL for simulation requires approximately twice the computational time of using GP.

The higher computational cost results from the RL-generated agent being invoked to predict every time a truck passes an intersection. While individual predictions may take minimal time, a complete simulation requires hundreds or thousands of computations. This cumulative computation time can render the RL's learning-based simulation too time-intensive when used to train dispatching strategies, thereby limiting its practicality.

We propose a hybrid GP and RL simulation approach to balance accuracy and computational efficiency. This strategy involves using GP to simulate less important intersections and RL to simulate intersections of high importance. By leveraging the strengths of both methods in this way, we aim to minimize the computational cost while preserving the accuracy of the simulation. The effectiveness and efficiency of this GPRL-H simulation approach will be examined in the subsequent part.

4.2.3. Hybridizing GP and RL

Unlike GP, wherein the acquired knowledge is directly translatable into arithmetic functions, thereby substantially accelerating each truck pass time calculation, RL enhances simulation accuracy and considerably extends the simulator's runtime. Empirical results indicate that RL-based simulations demand about thrice the computational time compared to GPs. To reduce this simulation time while preserving the accuracy of the simulations, this paper puts forward an innovative approach: the GPRL-H method for fast estimation of truck passage times at intersections.

As visualized in Fig. 10 and algorithm 4, when the simulation necessitates the computation of intersection passing time, the GPRL-H method determines whether to employ GP or RL for this calculation based on the current intersection's significance where the truck is positioned. Our historical truck movement data analysis revealed that certain intersections, particularly those at critical locations, frequently witness multiple trucks crossing simultaneously. In con-

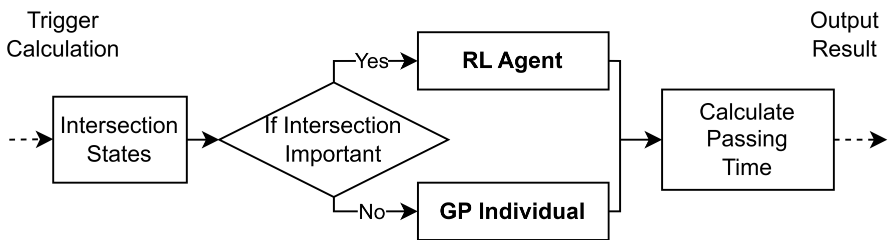

Figure 10: GPRL-H Method Flow Chart

trast, at other intersections, such instances are rather rare. Consequently, given their complex traffic conditions, we hypothesize that these critical intersections demand a detailed simulation facilitated by RL. In contrast, intersections with less complex traffic patterns could be effectively simulated using GP, yielding a level of accuracy similar to that of a comprehensive RL simulation. This hypothesis forms the basis for our proposal of the GPRL-H method, striking a balance between simulation accuracy and computational efficiency.

Algorithm 4 GPRL-H Method for Intersection Simulation

```
1: Input: Intersection significance, Historical truck movement data
2: Output: Computed passing time for truck
3: procedure COMPUTE_PASSING_TIME(intersection, truck_position)
4:     if is_critical(intersection) then
5:         use Reinforcement Learning (RL)
6:         RL_cross_time ← RL_Model(intersection, truck_position)
7:         return RL_pass_time
8:     else
9:         use Genetic Programming (GP)
10:        GP_cross_time ← GP_Model(intersection, truck_position)
11:        return GP_pass_time
12:    end if
13: end procedure
```

Nevertheless, the GPRL-H approach brings forth a challenge - determining the importance of each intersection. Given the unavailability of accurate GPS data, direct analysis of each intersection's congestion levels using historical GPS data to determine intersection significance is unfeasible. Therefore, drawing inspiration from the Pareto chart (Harvey & Sotardi, 2018), we propose a data-driven method incorporating the previously mentioned RL-based intersection simulation to evaluate the significance of various intersection nodes.

Denote that there are a total of o intersections within the port. In this research, we have $o = 66$, and all intersections in the default simulator sd are controlled by RL agents. The process can be outlined as follows:

1. Train the GP-based and RL-based agents independently on the training sets.
2. Replace the RL agent of one of the o intersections in the sd with the GP individual sequentially, generating o distinctive GPRL-H simulators.
3. Execute these substituted o simulators on test data sets and tally the final simulation results.
4. Implement the paired sample t-test method to compare the results of the o simulations with the sd, and compute the t-value. The higher t-test results indicate a greater discrepancy between the outcomes before and after removing a specific node. This suggests the higher importance of the node, implying that it should not be substituted with GP for control.
5. Choose the replaced intersection in the simulator with the lowest t-value (lowest influence on simulation accuracy) as the least important intersection and replace the RL agent at this intersection with a GP individual in default simulator sd. Set $o = o - 1$ to save simulation time and return to step 2 to continue replacing the next intersection until GP individuals control all intersections.

Moreover, we utilize the LGP generated individual as the GP part in the GPRL-H method. This approach has been adopted due to its superior performance compared to AGP in our experiments while maintaining a comparable execution time. Following the previously outlined steps, we derive the intersection importance diagram depicted in Fig. 11. This diagram reveals that, when substituting 40 out of 66 intersections with GP agents, approximately 96% of the performance characteristics observed in a fully RL-driven simulation are maintained. As a result, the proposed GPRL-H method in this paper defaults to controlling these 40 less significant intersections using GP, thereby ensuring an effective balance between simulation accuracy and speed.

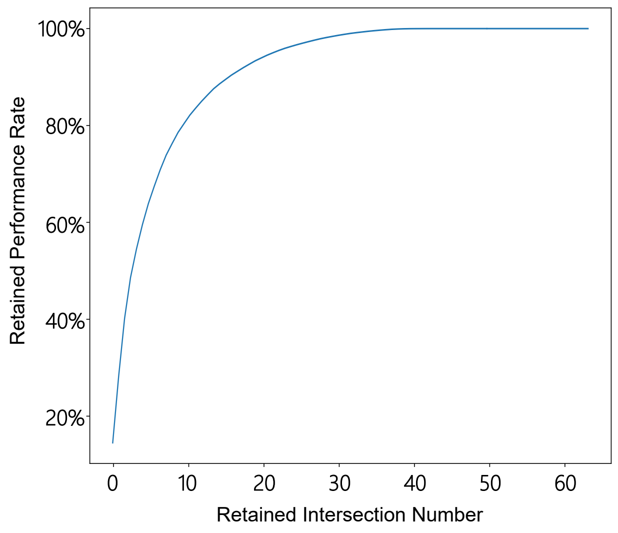

Figure 11: Intersection Importance Analysis Result

Our experimental results reveal that the proposed GPRL-H method combines the strengths of both GP and RL and strikes an effective balance between simulation accuracy and computational cost. We propose that this innovative method can be applied to the port dispatch simulation discussed in this paper and to a broader range of road traffic simulation applications, including factory automated guided vehicle (AGV) path simulation, urban traffic simulation, and mine vehicle scheduling. To demonstrate the superiority of the GPRL-H algorithm, a series of experiments and comparisons will be conducted in the next section.

5. Experiments and Results

In this section, we present the validation of the traditional event-based container port truck dispatching simulation proposed by (Shabayek & Yeung, 2002) as well as the learning-based AGP (Fogel et al., 1966), LGP (Wong & Leung, 1995), RL (Schulman et al., 2017), and GPRL-H methods that we have proposed, utilizing real-life historical data obtained from the Ningbo Meishan Port. Our primary objective is to highlight the excellent performance of our emphasized GPRL-H method. As articulated in Section 3, our central focus remains on the computation time and errors of two principal metrics: QCW and TPH.

The validation leverages real-world data and maps from Ningbo Meishan Port. The designated truck routes comply with the stipulations of the port; each route from crane to crane is unique and pre-set. The historical operational data chiefly constitutes the initiation s_{ij} and termination e_{ij} timings for the truck operations at the loading crane i and task j, supplemented with the start and end times of operations at the unloading cranes.

We have processed 20 days of historical operational data to generate 10 training and 10 test sets. Each set comprises roughly 20,000 job tasks, approximating the daily job volume at the port. The port comprises 5 berths, 35 QCs, and 75 YCs. Moreover, there are 66 intersections, and the number of trucks varies between 100 and 200, which is determined by the actual count of trucks in the historical data.

The parameter configurations and evolution methodologies for all GP methods are consistent with our prior work on AGP and LGP methods (Chen et al., 2022), having a population size of 1024, and crossover, mutation, and reproduction rates of 60%, 30%, and 10%, respectively. The GPRL-H method utilizes RL to regulate the 26 crucial intersections and employs LGP to fit 40 less critical intersections. The AGP, LGP, RL, and GPRL-H methods are trained on the 10 training sets using 100 distinct random seeds for 1000 generations each.

For the RL component of our study, we have chosen PPO and configured it to use two distinct deep neural networks. One of these networks serves as the policy network, or the 'actor network,' while the other functions as the value network, often termed the 'critic network.' The learning rate has been predetermined at 0.0003 for the actor network, and for the critic network, it is set at 0.001. These neural networks consist of layers with neuron quantities of 10, 100, 180, and 1, respectively. The Rectified Linear Unit (ReLU) is implemented as the activation function across all layers. Additionally, other hyperparameter settings of the RL model follow those outlined in (Chen et al., 2024), and early stopping is used to prevent overfitting. The final performance of the models, post-training, on the training sets is illustrated in Table 2.

Table 2: Traditional, AGP, LGP, RL, GPRL-H Methods Training Results

No.	**Historical**		**Traditional**		**AGP**		**LGP**		**RL**[1]		**GPRL-H**[1]	
	TPH	QCW	TPH	QCW	TPH	QCW	TPH	QCW	TPH	QCW	TPH	QCW
1	472.04	70.94	719.22	90.43	624.20	41.05	409.13	83.29	476.21	70.12	466.90	71.95
2	433.40	78.52	548.19	71.09	431.56	78.89	601.79	45.45	426.79	79.82	444.74	76.30
3	393.73	86.32	585.19	107.41	489.23	67.56	387.75	87.49	323.06	100.19	327.53	99.32
4	446.71	75.91	613.20	114.21	481.70	69.04	432.70	78.66	532.21	59.12	538.52	57.88
5	431.88	78.82	403.06	130.02	380.11	88.99	463.59	72.60	501.68	65.12	503.52	64.75
6	371.38	90.71	390.09	148.52	381.82	88.65	269.85	110.64	368.59	91.25	386.79	87.68
7	436.93	77.83	621.52	92.81	267.84	111.04	374.50	90.09	429.31	79.33	435.45	78.12
8	470.64	71.21	581.70	76.30	648.65	36.25	544.00	56.80	476.61	70.04	476.90	69.98
9	371.41	90.70	345.27	133.50	531.44	59.27	329.86	98.86	371.47	90.69	376.88	89.62
10	382.13	88.59	659.96	97.29	404.18	84.26	504.38	64.58	380.01	89.01	394.86	86.09
Avg.	421.02	80.96	546.74	106.16	464.07	72.50	431.75	78.85	428.59	79.47	435.21	78.17
Error	**N.A.**	**N.A.**	**36.43%**	**34.32%**	**24.36%**	**26.25%**	**17.89%**	**18.22%**	**6.17%**	**6.50%**	**6.86%**	**7.07%**

[1] RL and GPRL-H significantly differ from other methods, $p < 0.05$.

As outlined in the table, the unit of measure for TPH is TEU/hour, while that for QCW is hours. The error term represents the average absolute deviation across the ten datasets. Notably, our learning-based simulations consistently outperform conventional event-based simulations, irrespective of the intersection simulation method employed. We also incorporated an error parameter into our experiments to gauge the accuracy of various simulation techniques. It is crucial to clarify that this error is not merely an average of the last two metrics, but rather represents the cumulative error in predicting individual segments of truck travel time. Hence, scenarios may arise where the final metrics exhibit a smaller mean error but a larger cumulative error. Traditional event-based simulations display a significant error rate of approximately 35%. In contrast, AGP and LGP demonstrate considerably improved performance, with respective error rates of around 25% and 18%. Intriguingly, incorporating logical operators into GP enables LGP to outperform AGP. This suggests that intersection simulation is not a simple linear problem; it encapsulates diverse conditions that warrant the inclusion of logical operators for more accurate fitting. Remarkably, reinforcement learning (RL) exhibits a significantly lower error rate of around 6.5%, substantiating our claims in Section 4.2.2 that GP's performance is considerably influenced by its singular interactions with the environment during each evolutionary cycle. Although LGP manages some less critical nodes, the GPRL-H method maintains performance metrics comparable to RL, with an error rate of approximately 7%, essentially on par with RL.

Subsequently, we deployed the AGP, LGP, RL, and GPRL-H methods, initially trained on the training sets, into the test sets as shown in Table 3. To mitigate the risk of distortion arising from discrepancies in historical data when comparing training and test sets, we carefully selected test sets that closely mirror the characteristics of the training sets and historical data. This approach ensures consistency, particularly in the performance of manual heuristic methods, which exhibit similar effectiveness on both training and testing sets. This strategy aids in maintaining the reliability of our comparisons and the validity of our findings. The performance of the conventional method remained relatively unchanged, with an error rate persisting around 36%. Meanwhile, the AGP, LGP, RL, and GPRL-H methods all experienced nominal decreases, though these were not significant, merely around 1%-2%. This suggests our training did not lead to overfitting, and the intersection traffic rules gleaned from the learning-based methods are indeed generalizable, yielding effective performance even on datasets not previously encountered. This further affirms the versatility of our proposed method and its potential applicability in other simulations.

Table 3: Normal, AGP, LGP, RL, GPRL-H Methods Test Results

No.	**Historical**		**Normal**		**AGP**		**LGP**		**RL**[1]		**GPRL-H**[1]	
	TPH	QCW	TPH	QCW	TPH	QCW	TPH	QCW	TPH	QCW	TPH	QCW
1	457.65	73.76	740.50	18.21	614.41	42.98	713.78	23.46	433.02	78.60	434.44	78.32
2	344.21	96.04	413.43	82.45	354.14	94.09	294.82	105.74	306.60	103.43	302.29	104.27
3	482.55	68.87	553.73	54.89	569.48	51.80	463.86	72.54	516.05	62.29	538.12	57.96
4	362.13	92.52	520.68	61.38	273.62	109.90	247.04	115.12	327.51	99.32	329.73	98.88
5	492.18	66.98	614.07	43.04	480.49	69.28	413.87	82.36	504.43	64.58	509.66	63.55
6	405.43	84.02	461.49	73.01	337.86	97.29	430.48	79.10	455.10	74.26	468.66	71.60
7	418.06	81.54	544.09	56.79	303.39	104.06	367.91	91.39	410.01	83.12	408.67	83.38
8	433.57	78.49	504.59	64.54	500.73	65.30	318.17	101.15	403.09	84.48	419.80	81.19
9	457.73	73.75	720.02	22.24	564.02	52.87	459.40	73.42	412.24	82.68	407.73	83.57
10	365.82	91.80	524.78	60.58	113.09	141.43	377.77	89.45	332.69	98.30	332.34	98.37
Avg.	421.93	80.78	559.74	53.71	411.12	82.90	408.71	83.37	410.07	83.11	415.14	82.11
Error	**N.A.**	**N.A.**	**35.65%**	**37.46%**	**26.13%**	**26.10%**	**18.23%**	**19.29%**	**7.74%**	**7.71%**	**8.46%**	**8.58%**

[1] RL and GPRL-H significantly differ from other methods, $p < 0.05$.

Additionally, we computed the simulation time (SimT) consumed over each instance in seconds for each method. As presented in Table 4, it is evident that the standard event-based simulation is the most time-efficient, with an average simulation duration of approximately 5 seconds. The mean simulation times for AGP and LGP do not exhibit a notable difference, hovering around 11 seconds, which is approximately twice as long as the standard method. However, RL incurs the longest simulation time, with an astounding average of 35 seconds—sevenfold that of the standard method. Despite RL exhibiting the most impressive performance among all methods, such an extensive runtime is untenable. In employing the simulator to train port container truck dispatching strategies, many simulation executions are required. A simulation time seven times longer implies a corresponding increase in training duration when using the standard simulator, substantially impeding training efficiency. In contrast, our innovative GPRL-H method necessitates an average simulation time of around 17 seconds, merely half of that required by the RL method, while retaining near-equivalent simulation precision. This underscores that the GPRL-H method achieves a commendable equilibrium between two critical metrics: simulation time and simulation accuracy. It is capable of substantially reducing simulation time while preserving adequate simulation precision.

Table 4: Normal, AGP, LGP, RL, GPRL-H Methods Simulation Time (Second)

	Normal	**AGP**	**LGP**	**RL**	**GPRL-H**
Avg.	**4.97**	**10.87**	**11.27**	**36.49**	**17.50**

The empirical results compellingly demonstrate the efficacy of our machine learning based simulation methodology for container port truck dispatching. Under the purview of our proposed framework, applying AGP, LGP, RL, and GPRL-H methodologies to model truck throughput times at intersections substantially enhances the simulation's precision. Our GPRL-H method warrants specific mention. By amalgamating GP's computational speed and RL's superior accuracy, this approach is uniquely positioned to deliver high-grade simulation accuracy within a significantly reduced time frame.

This study's key strength resides in its capacity to provide a sophisticated, learning-based tool that transcends conventional event-based simulations. The GPRL-H method, being anchored in both GP and RL, exploits the strengths of these paradigms to yield a potent framework that is both time-efficient and high in precision. This innovative fusion of techniques renders the model responsive and adaptable, making it a powerful tool for real-world applications.

Integrating GP and RL in our method is an innovative approach that could be employed in other fixed-area vehicle simulations to improve simulation accuracy. While surrogate models are prevalent in simulation studies, our unique combination of GP and RL provides a novel contribution to the field. We must note, however, that the true reach of this method and its applicability beyond port dispatch is a promising prospect that warrants further investigation and corroboration. Its potential to significantly contribute to sectors where precise and efficient simulation is paramount is a compelling avenue for future exploration.

6. Discussion and Supplementary Experiments

In the previous experimental section, we provided evidence supporting the efficacy of our proposed learning-based simulation approach. In this section, we delve further into the necessity of implementing intersection simulations and the importance of integrating learning-based methods for more precise simulation. The cornerstone of our hypothesis is that the waiting times incurred by trucks at intersections during container transport markedly influence the overall transportation time. By enhancing the precision of truck passage time simulations at intersections, we can significantly improve the overall accuracy of the container port truck dispatch simulation. In turn, refining the simulation of truck passage through intersections bolsters the accuracy of the entire simulation. Moreover, calculating waiting times for trucks at intersections presents a complex task, necessitating the consideration of various conditions present at the intersection during the truck's passage. Thus, a constant intersection waiting time falls short of accurately simulating varying truck traffic conditions.

We reviewed our experimental data and noted that the standard event-based simulation generally overestimated TPH and underestimated QWT. This observation could potentially stem from the disregard for additional transit times that trucks incur at intersections during the transportation process. Consequently, we first introduced the fixed extra travel time (FETT) method, rooted in an exhaustive search approach (Nievergelt, 2000). This method endeavors to identify a constant additional truck travel time that minimizes the average error on the training set by exhaustive searching.

Table 5: Other Algorithems' Performance

Method		**Train**			**Test**		
		Average	Error	SimT	Average	Error	SimT
FETT	**TPH**	466.15	27.35%	5.37	424.34	34.50%	5.62
	QCW	72.09	29.28%		80.30	34.93%	
FIPT	**TPH**	421.48	20.26%	8.72	427.28	28.63%	8.66
	QCW	80.87	21.35%		78.73	28.95%	
DT	**TPH**	412.15	22.35%	6.35	424.34	33.50%	6.21
	QCW	82.09	23.28%		79.43	32.93%	
DNN	**TPH**	431.48	18.26%	6.87	415.28	30.63%	6.52
	QCW	77.87	17.35%		85.73	31.95%	
XGBoost	**TPH**	419.15	5.54%	7.32	432.34	27.63%	7.24
	QCW	79.89	5.26%		78.30	26.95%	

As illustrated in Table 5, introducing a fixed extra travel time decreases the error on the training set to approximately 28%, substantiating the effectiveness of considering additional travel time. Nonetheless, this reduction in error is notably limited. This observation suggests that given the substantial variance in the time required for each truck transit, it is impractical to identify a universal extra truck travel time that significantly reduces the average simulation error across multiple training sets. Furthermore, it was noted that the error reduction achieved with the FETT method on the test sets closely mirrored that of the conventional event-based method. This suggests that, while it may be feasible to identify a relatively optimal extra truck travel time for a specific subset of data sets, this value lacks broad generalizability. When the data set changes, the error rate escalates dramatically, underscoring the need for more adaptable, learning-based approaches.

To further our exploration, we proposed the fixed intersection passing time (FIPT) simulation method (Belbasi & Foulaadvand, 2008), anchored in the intersection simulation framework elaborated in this paper. Like the FETT method, the FIPT approach also leverages an exhaustive search but instead aims to identify a fixed intersection passing time that minimizes the average error on the training set. Remarkably, this method demonstrates a significantly lower error rate on the test sets than the FETT method, with an error rate of approximately 20%. This figure is comparable to the error rate associated with the GP methods and is even slightly superior to that of the AGP. These findings suggest a relatively satisfactory solution can be achieved upon introducing data-driven methodologies to learn intersection passing times, even with a simple exhaustive search method. This highlights the significance of simulating intersections and employing data-driven approaches. The exhaustive search method yielded a relatively fine solution through continuous testing on historical data, thereby fully substantiating the efficacy of the data-driven, learning-based simulation approach proposed in this study. Conversely, the FIPT method also exhibited a considerable increase in error on the training sets. This indicates that a fixed intersection passing time is not a universal rule; it solely represents the knowledge derived from a specific data set and does not constitute an abstract principle. This underlines the importance of incorporating sophisticated algorithms like GP and RL to learn the real-world dynamics of intersection passage.

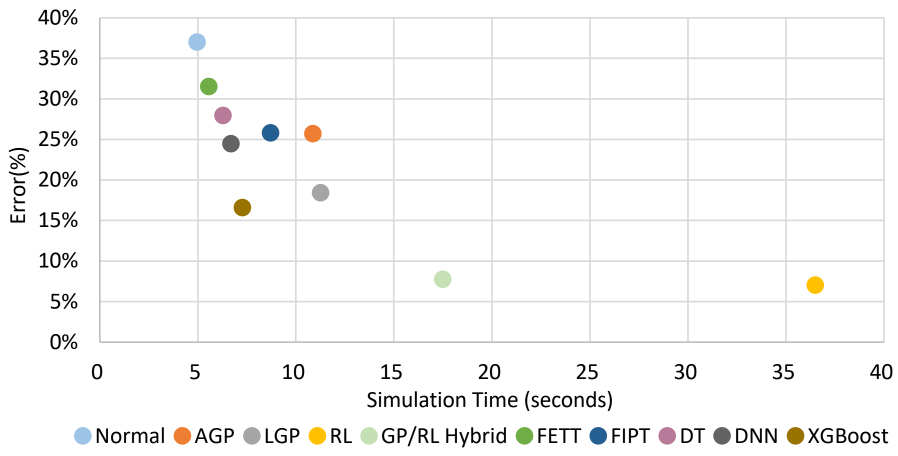

Figure 12: Error and Simulation Time (seconds) of All Methods in Test

After evaluating the impact of fixed extra travel time and fixed intersection passing time, we extended our comparative analysis to include three state-of-the-art supervised learning methods: decision tree (DT) (Myles et al., 2004), deep neural network (DNN) (Montavon et al., 2018), and extreme gradient boosting (XGBoost) (Chen & Guestrin, 2016). While we lack real-time data on truck intersection passing, we do possess comprehensive data on total travel time between work cranes. Consequently, we employed these supervised learning algorithms to predict truck travel time under various states. All algorithms were trained on a uniform dataset, utilizing state descriptors such as task type, container type, total number of trucks operating under the current bridge crane, port-bound truck count, target crane workload, historical average travel time, and overall port truck count. The algorithms were trained using default hyperparameters from the scikit-learn library, and their performance is tabulated in Table 5. Both DT and DNN exhibited poor convergence on the training set with approximately 20% error, while XGBoost outperformed them with an error rate of roughly 5.5%. However, when transitioned to the test set, all three algorithms displayed an error rate of about 30%, comparable to the simple FETT and FIPT methods. Moreover, XGBoost demonstrated significant performance degradation, indicative of overfitting during training.

From these experiments, it becomes evident that predicting truck travel time solely based on current port operation states is inherently flawed, especially in the absence of data on truck intersection passages. Despite XGBoost's initially promising training performance, it too falters on the test set due to overfitting. This underscores the crucial role of incorporating intersection simulations to represent port truck operations accurately. Notably, even a rudimentary method like FIPT approximates the performance of these supervised algorithms, attesting to the importance of intersection simulation in this context. Given the unavailability of intersection-specific data, employing unsupervised learning approaches in port truck dispatching simulation becomes indispensable, thereby justifying our initial focus on data-driven unsupervised learning in intersection simulation for more precise port simulation.

In the subsequent analysis, we plotted the simulation time and associated error in test for the Traditional, AGP, LGP, RL, GPRL-H, FETT, FIPT, DT, DNN , and XGBoost methods in Fig. 12. It is discernible from the graph that RL and GPRL-H methods markedly outperform the other approaches in terms of simulation error. Simultaneously, it is evident that each method holds its unique advantages when optimizing the two key objectives - simulation time and accuracy. Thus, in practical applications, decision-makers can tailor their method to align with their specific requirements. Furthermore, the graph underlines that the GPRL-H method attains high simulation accuracy without significantly increasing simulation time. This reinforces the superiority of the method introduced in this study, lending further credence to its effectiveness in balancing simulation accuracy and efficiency.

As a result of comprehensive experimental analysis, our research validates the initial hypothesis that truck transit times at intersections play a crucial role in the accuracy of simulation models. Furthermore, incorporating unsupervised learning techniques to model these transit times significantly enhances the precision of the simulation. The key findings of our study are summarized as follows:

- Experimental results substantiate the pivotal impact that truck transit times at intersections exert on the accuracy of simulation models.
- The inclusion of simple transit events in the simulation substantially improves model accuracy, emphasizing the necessity of accurately representing complex real-world conditions.
- Findings from using the FIPT method reveal the inherent complexity of intersection passing rules, which simplistic temporal metrics cannot capture.
- Traditional supervised learning methods, such as DT, DNN, and XGBoost, are inadequate for precisely predicting truck travel times without intersection considerations.
- The observed complexities validate the need to employ unsupervised algorithms like GP and RL to decode intricate intersection rules. This lends credence to the efficacy of a data-driven, learning-based approach for addressing the challenges in container port truck dispatch scenarios.

While the data-driven machine learning-based simulation method enhances accuracy in port truck dispatching simulations, it has limitations. The method requires additional training and may not perform optimally in unfamiliar environments. To refine simulation accuracy further, integration of GPS data is essential. The learning algorithm can automatically correct and refine this data, leading to more precise simulation outcomes. Moreover, the increasing adoption of IoT (Internet of Things) technologies in modern container terminals offers potential for more accurate data collection. As IoT devices become more prevalent, they can provide precise data that further enhances simulation accuracy. Furthermore, we conducted simulation experiments based solely on the historical operation data of Meishan Terminal of Ningbo Port and the map layout of it. These experiments have limitations; Ningbo Meishan Port is a semi-automated terminal with a different yard layout compared to a fully-automated terminal. Additionally, variations in port operation tasks, weather, and shoreline length may affect the final simulation results. However, overall, the advantages of the RLGP-H method are evident, making it a highly viable and practical solution for current real-world applications.

7. Conclusion and Future Work

The importance of intersection simulation in container port logistics cannot be overstated, significantly influencing the final simulation accuracy. As demonstrated in this research, the application of data-driven machine learning based simulations drastically improves the precision of intersection simulations, thereby enhancing the overall efficiency of truck dispatch operations. The intrinsic complexity of intersection passing rules necessitates the integration of advanced machine learning algorithms to create realistic simulations successfully.

Our comparative analysis, including AGP, LGP, RL, GPRL-H, DT, DNN, and XGBoost, highlights GPRL-H's advantages. GPRL-H achieves superior accuracy without excessive computation time by combining RL and GP strengths. This balance makes it ideal for complex intersection simulations, marking a shift in research toward data-driven methods for transportation simulation. With robust performance, GPRL-H shows promise for broader vehicle and equipment movement simulation applications.

In terms of applications, the machine learning-based simulation method reduces simulation errors, opening avenues for integration with various existing optimal scheduling algorithms. This integration can aid in training the algorithm or validating its performance, enhancing algorithmic robustness and facilitating more effective practical solutions. Ultimately, this approach can assist port companies in mitigating performance degradation of scheduling algorithms when transitioning from simulated to real environments, thereby enhancing operational logistic efficiency in real-world scenarios.

Looking forward, our focus is to integrate RL and GP methods further, leveraging the unique strengths of each to achieve optimal performance. Specifically, we plan to explore techniques such as using RL to optimize GP individuals, guiding GP evolution with RL, and determining the genotype fitness of GP individuals using RL. Through extensive integration of RL and GP, we aim to harness their combined strengths and develop a synergistic approach. This comprehensive integration has broad applicability beyond simulation, extending to scheduling, routing, packing, and other problem domains. Additionally, we aim to conduct more exploratory data analysis to identify pain points in port logistics operations and further enhance the port's operation efficiency by solving these problems.

References

Afrapoli, A. M., Tabesh, M., & Askari-Nasab, H. (2019). A multiple objective transportation problem approach to dynamic truck dispatching in surface mines. *European Journal of Operational Research*, *276*, 331–342.

Ahvanooey, M. T., Li, Q., Wu, M., & Wang, S. (2019). A survey of genetic programming and its applications. *KSII Trans. Internet Inf. Syst.*, *13*, 1765–1794.

Ardeh, M. A., Mei, Y., Zhang, M., & Yao, X. (2022). Knowledge transfer genetic programming with auxiliary population for solving uncertain capacitated arc routing problem. *IEEE Transactions on Evolutionary Computation*, .

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34*, 26–38.

Arvin, R., Khattak, A. J., Kamrani, M., & Rio-Torres, J. (2020). Safety evaluation of connected and automated vehicles in mixed traffic with conventional vehicles at intersections. *Journal of Intelligent Transportation Systems*, *25*, 170–187.

Bagdatli, M. E. C., & Dokuz, A. S. (2021). Vehicle delay estimation at signalized intersections using machine learning algorithms. *Transportation research record*, *2675*, 110–126.

Bai, R., Chen, X., Chen, Z.-L., Cui, T., Gong, S., He, W., Jiang, X., Jin, H., Jin, J., Kendall, G. et al. (2023). Analytics and machine learning in vehicle routing research. *International Journal of Production Research*, *61*, 4–30.

Banzhaf, W., Nordin, P., Keller, R. E., & Francone, F. D. (1998). *Genetic programming: an introduction: on the automatic evolution of computer programs and its applications*. Morgan Kaufmann Publishers Inc.

Belbasi, S., & Foulaadvand, M. E. (2008). Simulation of traffic flow at a signalized intersection. *Journal of Statistical Mechanics: Theory and Experiment*, *2008*, P07021.

Bi, Y., Xue, B., Zhang, M., Bi, Y., Xue, B., & Zhang, M. (2021). Evolutionary computation and genetic programming. *Genetic Programming for Image Classification: An Automated Approach to Feature Learning*, (pp. 49–74).

Bonacich, E., & Wilson, J. B. (2008). *Getting the goods: Ports, labor, and the logistics revolution*. Cornell University Press.

de Carvalho, J. P., & Dimitrakopoulos, R. (2021). Integrating production planning with truck-dispatching decisions through reinforcement learning while managing uncertainty. *Minerals*, *11*, 587.

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

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794).

Chen, X., Bai, R., & Dong, H. (2019). A multi-layer gp hyper-heuristic for real-time truck dispatching at a marine container terminal. *MISTA 2019*, .

Chen, X., Bai, R., Qu, R., & Dong, H. (2022). Cooperative double-layer genetic programming hyper-heuristic for online container terminal truck dispatching. *IEEE Transactions on Evolutionary Computation*, .

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

Chen, X., Bai, R., Qu, R., Dong, J., & Jin, Y. (2024). Deep reinforcement learning assisted genetic programming ensemble hyper-heuristics for dynamic scheduling of container port trucks. *IEEE Transactions on Evolutionary Computation*, .

Christodoulaki, E., Kampouridis, M., & Kanellopoulos, P. (2022). Technical and sentiment analysis in financial forecasting with genetic programming. In *2022 IEEE Symposium on Computational Intelligence for Financial Engineering and Economics (CIFEr)* (pp. 1–8). IEEE.

Chuang, T.-N., Lin, C.-T., Kung, J.-Y., & Lin, M.-D. (2010). Planning the route of container ships: A fuzzy genetic approach. *Expert Systems with Applications*, *37*, 2948–2956.

Dragović, B., Tzannatos, E., & Park, N. K. (2017). Simulation modelling in ports and container terminals: literature overview and analysis by research field, application area and tool. *Flexible Services and Manufacturing Journal*, *29*, 4–34.

Elhenawy, M., Chen, H., & Rakha, H. A. (2014). Dynamic travel time prediction using data clustering and genetic programming. *Transportation Research Part C: Emerging Technologies*, *42*, 82–98. URL: https:

//www.sciencedirect.com/science/article/pii/S0968090X14000588. doi:https://doi.org/10.1016/j.trc.2014.02.016.

Fan, Q., Bi, Y., Xue, B., & Zhang, M. (2022). Genetic programming for feature extraction and construction in image classification. *Applied Soft Computing*, *118*, 108509.

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). *Artificial intelligence through simulated evolution*. Wiley.

Gould, H., Tobochnik, J., & Christian, W. (2007). An introduction to computer simulation methods. *Comput. Phys*, *10*, 652–653.

Harahap, E., Darmawan, D., & Badruzzaman, F. (2020). Simulation of traffic t-junction at cibiru-cileunyi lane using simevents matlab. In *Journal of Physics: Conference Series* (p. 012074). IOP Publishing volume 1613.

Harvey, H. B., & Sotardi, S. T. (2018). The pareto principle. *Journal of the American College of Radiology*, *15*, 931.

Hassan, S. A. (1993). Port activity simulation: an overview. *ACM SIGSIM Simulation Digest*, *23*, 17–36.

He, J., Huang, Y., Yan, W., & Wang, S. (2015). Integrated internal truck, yard crane and quay crane scheduling in a container terminal considering energy consumption. *Expert Systems with Applications*, *42*, 2464–2487.

Hildebrandt, F. D., Thomas, B. W., & Ulmer, M. W. (2023). Opportunities for reinforcement learning in stochastic dynamic vehicle routing. *Computers & operations research*, *150*, 106071.

Hu, D., & Mohamed, Y. (2013). Time-stepped, simulation-based scheduling system for large-scale industrial construction projects. In *2013 Winter Simulations Conference (WSC)* (pp. 3249–3256). IEEE.

Hu, J., Yan, X., Wang, G., Tu, M., Zhang, X., Wang, H., Gruyer, D., & Lai, J. (2024). A simulation platform for truck platooning evaluation in an interactive traffic environment. *IEEE Transactions on Intelligent Transportation Systems*, .

Jackson, I., Jesus Saenz, M., & Ivanov, D. (2024). From natural language to simulations: applying ai to automate simulation modelling of logistics systems. *International Journal of Production Research*, *62*, 1434–1457.

Jin, J., Cui, T., Bai, R., & Qu, R. (2023). Container port truck dispatching optimization using real2sim based deep reinforcement learning. *European Journal of Operational Research*, .

Juan, A. A., Faulin, J., Pérez-Bernabeu, E., & Domínguez, O. (2013). Simulation-optimization methods in vehicle routing problems: a literature review and an example. In *Modeling and Simulation in Engineering, Economics, and Management: International Conference, MS 2013, Castellón de la Plana, Spain, June 6-7, 2013. Proceedings* (pp. 115–124). Springer.

Koh, S., Zhou, B., Fang, H., Yang, P., Yang, Z., Yang, Q., Guan, L., & Ji, Z. (2020). Real-time deep reinforcement learning based vehicle navigation. *Applied Soft Computing*, *96*, 106694.

Koza, J. R. (1994). *Genetic programming II: automatic discovery of reusable programs*. MIT press.

Lee, S., Kim, Y., Kahng, H., Lee, S.-K., Chung, S., Cheong, T., Shin, K., Park, J., & Kim, S. B. (2020). Intelligent traffic control for autonomous vehicle systems based on machine learning. *Expert Systems with Applications*, *144*, 113074.

Lubulwa, G., Lightfoot, A., & Malarz, A. (2010). Analyses of stevedoring productivity in australia's five major container ports. In *Australian Transport Forum Paper* (pp. 1–19).

Luo, J., & Wu, Y. (2020). Scheduling of container-handling equipment during the loading process at an automated container terminal. *Computers & Industrial Engineering*, *149*, 106848.

Lv, Y., Zou, M., Li, J., & Liu, J. (2024). Dynamic berth allocation under uncertainties based on deep reinforcement learning towards resilient ports. *Ocean & Coastal Management*, *252*, 107113.

Mirzaei-Nasirabad, H., Mohtasham, M., Askari-Nasab, H., & Alizadeh, B. (2023). An optimization model for the real-time truck dispatching problem in open-pit mining operations. *Optimization and Engineering*, (pp. 1–25).

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., & Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning* (pp. 1928–1937). PMLR.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, .

Montavon, G., Samek, W., & Müller, K.-R. (2018). Methods for interpreting and understanding deep neural networks. *Digital signal processing*, *73*, 1–15.

Moradi Afrapoli, A. (2019). A hybrid simulation and optimization approach towards truck dispatching problem in surface mines, .

Morgan, P. L., Williams, C., Flower, J., Alford, C., & Parkin, J. (2019). Trust in an autonomously driven simulator and vehicle performing maneuvers at a t-junction with and without other vehicles. In *Advances in Human Aspects of Transportation: Proceedings of the AHFE 2018 International Conference on Human Factors in Transportation, July 21-25, 2018, Loews Sapphire Falls Resort at Universal Studios, Orlando, Florida, USA 9* (pp. 363–375). Springer.

Myles, A. J., Feudale, R. N., Liu, Y., Woody, N. A., & Brown, S. D. (2004). An introduction to decision tree modeling. *Journal of Chemometrics: A Journal of the Chemometrics Society*, *18*, 275–285.

Nievergelt, J. (2000). Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power. In *Sofsem* (pp. 18–35). Springer.

Notteboom, T. (2016). The adaptive capacity of container ports in an era of mega vessels: The case of upstream seaports antwerp and hamburg. *Journal of Transport Geography*, *54*, 295–309.

Poss, M., & Raack, C. (2013). Affine recourse for the robust network design problem: between static and dynamic routing. *Networks*, *61*, 180–198.

Ramirez, M. R., & Belytschko, T. (1989). An expert system for setting time steps in dynamic finite element programs. *Engineering with computers*, *5*, 205–219.

Raza, A., Li, J., Adnan, M., & Ahmad, I. (2024). Optimal load forecasting and scheduling strategies for smart homes peer-to-peer energy networks: A comprehensive survey with critical simulation analysis. *Results in Engineering*, (p. 102188).

Sarmiento, M. G. C., Epprecht, E. K., Oliveira, F. L. C., Rodrigues, A. T. S., & Canchumuni, S. W. A. (2019). The use of simulation to model the dispatch of inbound containers in port terminals. *Pesquisa Operacional*, *39*, 155–175.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In *International conference on machine learning* (pp. 1889–1897). PMLR.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, .

Schulte, F., Lalla-Ruiz, E., González-Ramírez, R. G., & Voß, S. (2017). Reducing port-related empty truck emissions: A mathematical approach for truck appointments with collaboration. *Transportation Research Part E: Logistics and Transportation Review*, *105*, 195–212. URL: https://www.sciencedirect.com/science/article/pii/S1366554516300801. doi:https://doi.org/10.1016/j.tre.2017.03.008.

Shabayek, A., & Yeung, W. (2002). A simulation model for the kwai chung container terminals in hong kong. *European Journal of Operational Research*, *140*, 1–11.

Shirazi, M. S., & Morris, B. T. (2017). Looking at intersections: A survey of intersection monitoring, behavior and safety analysis of recent studies. *IEEE Transactions on Intelligent Transportation Systems*, *18*, 4–24. doi:10.1109/TITS.2016.2568920.

Sutton, R. S., Barto, A. G. et al. (1998). *Introduction to reinforcement learning* volume 135. MIT press Cambridge.

Tang, W., Chen, X., Lang, M., Li, S., Liu, Y., & Li, W. (2024). Optimization of truck–cargo online matching for the less-than-truck-load logistics hub under real-time demand. *Mathematics*, *12*, 755.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*. volume 30.

Wei, M., He, J., Tan, C., Yue, J., & Yu, H. (2023). Quay crane scheduling with time windows constraints for automated container port. *Ocean & Coastal Management*, *231*, 106401.

Wong, M. L., & Leung, K. S. (1995). Combining genetic programming and inductive logic programming using logic grammars. In *Proceedings of 1995 IEEE International Conference on Evolutionary Computation* (pp. 733–736). IEEE volume 2.

Xu, M., Mei, Y., Zhang, F., & Zhang, M. (2024). Niching genetic programming to learn actions for deep reinforcement learning in dynamic flexible scheduling. *IEEE Transactions on Evolutionary Computation*, .

Xu, Y., Song, X., Weng, Z., & Tan, G. (2014). An entry time-based supply framework (etsf) for mesoscopic traffic simulations. *Simulation Modelling Practice and Theory*, *47*, 182–195.

Yi, W., Qu, R., Jiao, L., & Niu, B. (2022). Automated design of metaheuristics using reinforcement learning within a novel general search framework. *IEEE Transactions on Evolutionary Computation*, .

Yu, L., Zhang, C., Jiang, J., Yang, H., & Shang, H. (2021). Reinforcement learning approach for resource allocation in humanitarian logistics. *Expert Systems with Applications*, *173*, 114663.

Yu, Q., & Zhou, Y. (2019). Traffic safety analysis on mixed traffic flows at signalized intersection based on haar-adaboost algorithm and machine learning. *Safety Science*, *120*, 248–253.

Yung, N. H., & Ye, C. (1999). An intelligent mobile vehicle navigator based on fuzzy logic and reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *29*, 314–321.

Zhang, X., Li, R., Wang, C., Xue, B., & Guo, W. (2024). Robust optimization for a class of ship traffic scheduling problem with uncertain arrival and departure times. *Engineering Applications of Artificial Intelligence*, *133*, 108257.

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