Reinforcement Learning for Digital Twin empowered Ride-sharing System Optimization

Journal:	*IEEE Network Magazine*
Manuscript ID	NETWORK-23-00647
Topic or Series:	Open Call Article
Date Submitted by the Author:	17-Sep-2023
Complete List of Authors:	Jiang, Kai; Wuhan University Cao, Yue; Wuhan University, Wang, Zhenning; Wuhan University Zhou, Huan; China Three Gorges University, ; China Three Gorges University Zhu, Hong; China National Institute of Standardization Liu, Zhi
Key Words:	Ride-sharing, Reinforcement Learning, Digital Twin

Reinforcement Learning for Digital Twin empowered Ride-sharing System Optimization

Kai Jiang, Yue Cao, *Senior Member, IEEE*, Zhenning Wang, Huan Zhou, *Member, IEEE*, Hong Zhu, and Zhi Liu, *Senior Member, IEEE*

***Abstract*—Recently, as the popularity of ride-sharing services continues to rise, a crucial technical requirement is to optimize operational efficiency for these systems. Digital Twin (DT) and Reinforcement Learning (RL) have emerged as promising solutions for this requirement. By creating virtual replicas of environments, DT enables RL to learn from data and facilitate more informed decisions that benefit all ride-sharing stakeholders. However, the integration of DT and RL is still in its infancy due to its dependence on processing power, memory resources, and high-quality data. Thus, this article provides a comprehensive overview of this novel field from a system-level perspective. First, we focus on a typical ride-sharing application under the umbrella of DT and RL, and explain their origins in this context. Then, we delve into how these two technologies can be integrated for ride-sharing system optimization. The potential advantages and impact of their integration are elaborated holistically. Subsequently, we discuss a specific embodiment of this integration for large-scale multi-vehicle order dispatching. Finally, we strive to shed light on potential challenges, which may facilitate the transformation of this topic from theory to practice.**

I. Introduction

With the popularity of shared mobility in traffic resource reconfiguration, on-demand ride-sharing services prevail as an essential pillar of the modern transportation system. The user-centric ride-sharing platforms, exemplified by Uber, DiDi Chuxing, *etc.*, have substantially revolutionized the transportation landscape. By one estimate, the global market value of the ride-sharing industry gains $135 billion in 2020, which is expected to reach $218 billion by 2025 [1].

Ride-sharing services can coordinate between finite supply (vehicle) and asymmetric demand (order). This efficient and sustainable traffic mode facilitates a common vehicle to synchronously serve multiple passengers with similar and time-diverse itineraries [2]. It alleviated traffic congestion, emission, and energy consumption via elevating idle seat utilization, thereby offering enormous potential to improve transportation efficiency and living conditions. However, as the popularity of ride-sharing services continues to rise, a crucial technical requirement is to optimize operational efficiency for these systems, *e.g.*, long pick-up/detour delays for passengers, the persistent gap between Supply and Demand (SD), and up to 40% vacant time for ride-sharing vehicles at large-scale [3]. Based on the advantages of simulation, prediction and control, Digital Twin (DT) and Reinforcement Learning (RL) have emerged as promising solutions, which can improve the overall efficiency and sustainability of the ride-sharing system.

Indeed, DT and RL are transformative technologies in the transportation industry. Thereinto, RL is a machine-learning technique that trains agents (individual decision-making units) to make decisions through trial-and-error interactions with the environment. With the revival of artificial intelligence, it has exhibited its particular potential for sequential decision-making with a long-term objective under uncertainty [4], [5]. In ride-sharing, it can predict demand patterns, optimize trip pricing, order matching, vehicle repositioning, routing guidance, and even prevent congestion. In contrast, the concept of DT is relatively new but has already gained widespread concern. It improves various system accuracy through better data utilization. Essentially, DT refers to creating a virtual replica (model) of a physical asset or system, such as vehicle fleets in ride-sharing. The created DT model is continuously updated in real-time to match its physical counterpart. It evolves in sync with the physical system throughout its entire lifecycle [6]. For ride-sharing, this virtual model can provide real-time data and insights, predictive inference, monitor and optimize the performance of the physical systems, and even allow for continuously improving simulations without directly affecting the physical system.

Certainly, the integration of DT and RL is still in its infancy, yet its potential to revolutionize the ride-sharing industry is undeniable. By integrating DT and RL, data interaction changes radically. We can create virtual replicas of ride-sharing environments and use state-of-the-art algorithms to optimize their performance. This is where DT comes in, enabling RL to learn from data and facilitating more informed decisions that benefit all ride-sharing stakeholders involved.

This article provides a comprehensive overview of this novel field from a systems-level perspective. We integrate DT and RL and explore how this integration can optimize operational efficiency for ride-sharing systems. Furthermore, despite this integration offering numerous advantages, we also highlight potential challenges which may facilitate the transformation of this topic from theory to practice.

II. PrIor Knowledge

A. Technical basis

1) **RL**: RL, developed from multiple disciplines such as statistics, control theory, and psychology, is an autonomous

K. Jiang, Y. Cao (corresponding author), and Z. Wang are with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430000, China. (e-mail: kai.jiang, yue.cao, zn.wang115@whu.edu.cn.).

H. Zhou is with the College of Computer, Northwestern Polytechnical University, Xi'an 710000, China. (e-mail: zhouhuan117@gmail.com).

H. Zhu is with the China National Institute of Standardization, Beijing 100191, China. (e-mail: zhuhong@cnis.ac.cn).

Z. Liu is with the Department of Computer and Network Engineering, University of Electro-Communications, Tokyo, Japan. (e-mail: liu@ieee.org).

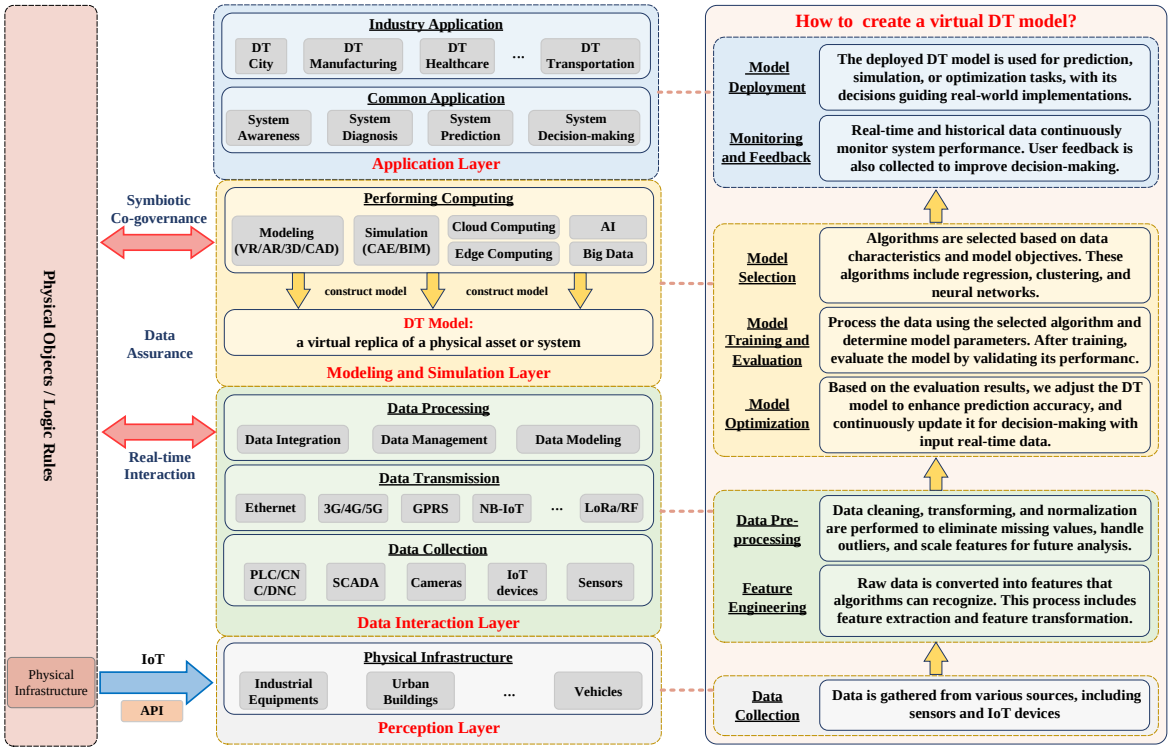

Fig. 1. **Hierarchical architecture and main steps for creating a virtual model in DT ecosystem.**

learning method based on the Markov Decision Process (MDP) framework. It is essentially interactive learning, where the agent makes decisions individually through repeated interaction with the uncertain environment [7]. Specifically, each agent selects and executes an action based on the observed state, and adjusts its action selection mechanism according to the received rewards. Eventually, the goal for the agent is to take a series of actions in response to the environment and obtain the expected maximum reward (*a.k.a.* cumulative discounted reward).

MDP with Markov property is the mathematical theory foundation of RL, typically represented by a quintuple ⟨*state, action, state transition probability, reward, discounting factor*⟩. Once the MDP describes the sequential decision-making problem, the solution can be obtained by formalizing the value function or state value function through the Bellman equation. That is to say, RL optimizes the Bellman equation to solve MDP problems. This simplifies the solution process by converting it into the optimization of the Bellman equation. Further details regarding parameter definitions and mathematical solving processes can be found in [8].

2) **DT**: Recently, the concept and implementation of DT are slowly gaining traction, which provides an excellent capability to simulate real-world entities in the industrial environment. According to IBM (2020): "DT is a virtual representation of an object or system that spans its lifecycle, is updated from real-time data, and uses simulation, machine learning, and reasoning to help decision-making." Specifically, DT involves creating a virtual replica of a physical object or system using digital data. This digital replica is supported by Building Information Modelling (BIM), machine learning, cloud/edge computing, and abundant sensor data. It provides real-time data and insights, allowing for monitoring and optimising the physical system performance, and even continuously improving simulations without directly affecting the physical system [9].

As shown in **Fig. 1** (left side), the DT ecosystem consists of four layers. The perception layer comprises physical infrastructure used in urban construction, fleet management, and industrial sectors. The data interaction layer includes data collection, transmission, and processing. The modeling and simulation layer constructs the DT model, which involves modeling, simulation, and control via correlative techniques. Finally, according to scenarios and application scopes, the application layer can be divided into common applications and industrial applications.

B. Ride-sharing Architecture

Compared to taxi-hailing services (*i.e.*, cruising on streets to discover passengers), ride-sharing services require real-time decision-making for vehicles over an uncertain future demand on the centralized platform. The ride-sharing platform is the brain of the ride-sharing system, enabling efficient service and strict operational regulation. Passengers and drivers[1] establish connections through the platform, thereby SD sides of this bilateral market were formed. Based on self-interest, passengers expect service experience-related indicators (*e.g.*, pick-up/detour delays) and apportioned costs should be considered when establishing ride-sharing service modules. Meanwhile,

[1]For ease of presentation, we have not distinguished the paraphrase of "driver" and "vehicle" in this paper.

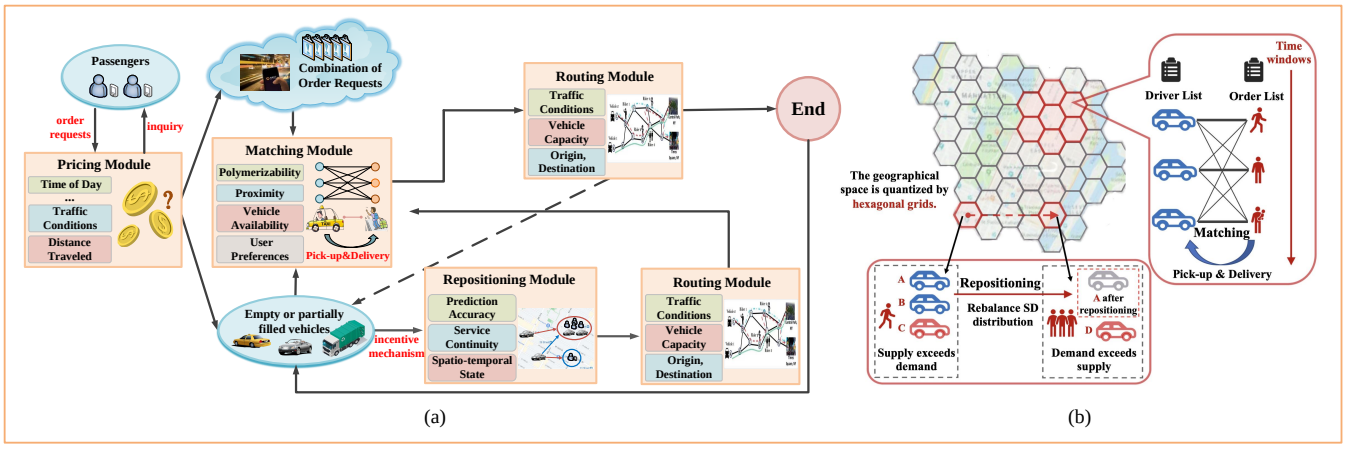

Fig. 2. **(a) The architecture of a ride-sharing platform. (b) An overview of the matching and repositioning stages.**

drivers prioritize the profitability of ride-sharing trips and fewer vacant time.

From the perspective of platforms, drivers, and passengers, the successful implementation of the system relies on situation awareness, real-time regulation, adaptive tailoring of SD, and the combinatorial optimization of all integrated modules that deliver services [3]. **Fig. 2(a)** illustrates the architecture of a ride-sharing platform, which typically consists of four modules: pricing, matching, routing, and repositioning.

1) **Pricing Module**: Positioned upstream of other modules, the pricing module serves as a lever to achieve coordination of SD at a macro level. Price fluctuations can affect the profit (price paid by passengers and driver incomes) and satisfaction of both ride-sharing shareholders, which indirectly change the balance of SD. For this, dynamic pricing (*i.e.*, surge pricing during peak hours) enables real-time adjustment of trip fares with considering SD changes. Potential passengers with travel demands submit the order request, and can either accept or reject offers based on the received fares from the platform. In addition, the pricing module heavily relies on appropriate economic models that capture the incentives of both ride-sharing shareholders. This leads to two major challenges: fairness and privacy. Fairness refers to the requirement that the pricing algorithm in the economic model should be transparent and fair, enabling passengers to understand how fares are calculated. Privacy corresponds to protecting the collected real-time data during the algorithm execution process.

2) **Matching Module**: After the passenger accepts the pricing, the matching module combines multiple order requests and assigns them to available vehicles. This process and its generalized forms may appear under different names, like order matching, order dispatching, or vehicle dispatching. Indeed, it corresponds to an online stochastic problem originating from traditional bipartite graph matching. The uncertainty arises from demand arrivals, trip times, SD distributions, and the significant spatiotemporal nature of ride-sharing. Generally, order matching in ride-sharing requires strong system-level coordination, as it is non-trivial to assign one or more passengers to a demand-matched vehicle (empty or partially filled). Based on the real-time availability of supply, order requests are batched in each fixed time window until successful matches are made [10]. Whether a driver is eligible to match a ride-sharing request partly depends on the spatial distance and the destination.

During the matching process, drivers and passengers typically exhibit asymmetric exit mechanisms, *i.e.*, orders may be canceled during the passenger waiting period before matching (pre-matching cancellation), or the driver pick-up period after a successful matching (post-matching cancellations). Different trip requests usually change the spatial status of drivers, which affects the supply distribution for future matches. After successfully delivering passengers to their destinations, the driver will receive the fare and update their capacity status. Notably, considering the practical deployment, balances among multiple indicators need to be struck in order matching, such as profits and satisfaction.

3) **Repositioning Module**: The backend repositioning module in ride-sharing guides non-occupied vehicles (*i.e.*, those without any assigned order requests) to specific locations to meet anticipated future demand. System-level vehicle repositioning, also known as vehicle rebalancing or fleet management, achieves a balanced ride supply across geographic dimensions by dispatching non-occupied vehicles to areas with a higher possibility of demands. At this point, the effectiveness of repositioning depends on the accuracy of demand predictions for specific areas, and vehicle repositioning should satisfy continuously changing regional travel demand.

Notably, referring back to the previous subsection, we found that repositioning and matching are similar, as they both involve dispatching vehicles to different locations. In theory, repositioning can be viewed as matching vehicles with virtual order requests. However, in practice, these two issues are usually addressed separately by independent modules due to the heterogeneous review interval of matching and repositioning (*i.e.*, matching and repositioning decisions are typically made asynchronously.). **Fig. 2(b)** provides an overview of matching and repositioning in ride-sharing, where the geographical space is quantized by hexagonal grids. Additionally, since vehicles following repositioning strategies are voluntary, the platform must provide corresponding incentive mechanisms

for drivers to encourage repositioning execution.

4) **Routing Module**: The routing module in ride-sharing provides link-level turn-by-turn navigation for drivers. Its goal is to fulfill order requests and execute repositioning operations within minimal delay. The dynamics arise from the insertion of new order requests into the route while existing orders are completed. The routing module needs to accommodate the non-stationary delivery sequence and road network conditions in real-time. In this case, the problem scope generally differs from the static Vehicle Routing Problem (VRP), where destinations that vehicles will traverse to are pre-known. The routing module has to determine how to provide services for each order request in the constructed setting (*e.g.*, deciding the individual service priority).

Furthermore, vehicle routing largely depends on the order information matched during the time dimension. Whenever a passenger is assigned to the available vehicle, the VRP solver in the routing module must re-plan the route to consider additional pick-up/drop-off behaviors, update the travel time for all passengers, and check the constraints. Indeed, the routing stage can be viewed as the subsequent operation for matching and repositioning decisions. A robust and reliable routing strategy is crucial for providing empowered navigation information back to the prior stage. Furthermore, the unresolved problem is that the routing decision of multiple passengers should be consistent with the global objective of ride-sharing.

III. RL for DT Empowered Ride-sharing System Optimization

A. RL in Ride-sharing

The stochastic and non-stationary settings in ride-sharing settings make predicting and modeling in spatiotemporal space difficult. Meanwhile, the multi-step sequential decision-making nature leads to strong spatiotemporal dependency. Conventional optimization methods [11], [12] often rely on regular distribution hypotheses and predetermined rules. These methods are quasi-static and myopic, which show weaknesses in prediction accuracy, model complexity, and adaptability to uncertainty. They learn only from the current immediate returns and neglect the long-term influences. In contrast, RL offers adaptive and foresighted decision-making by estimating anticipated future value, making it well-suited for sequential decision-making under uncertainty. Meanwhile, RL can effectively resolve conflicts among multiple levers in ride-sharing optimization through continuous real-time feedback.

Generally, the state of an agent in ride-sharing consists of global SD information, and vehicle situations. Vehicle situations contain additional key information on occupancy status, and the origin-destination of the passengers on board. The action space depends on whether the agent is modeled at the vehicle or system levels. A system-level agent has to make action decisions for the entire fleet together. Although configuring the agent from a centralized perspective (*i.e.*, system-level) might achieve superior performance, the complexity of action space is exponential. Alternatively, vehicle-level modeling is easier to implement for the straightforward definition of state transition, action, and reward. An individual vehicle agent can match to a possible combination of order requests, reposition to another location, or both in batches, where the available trips for each vehicle are typically generated through a graph theory-based method. **Fig. 3** shows the interaction between the agent and its environment, which also displays the several actions the agent can take.

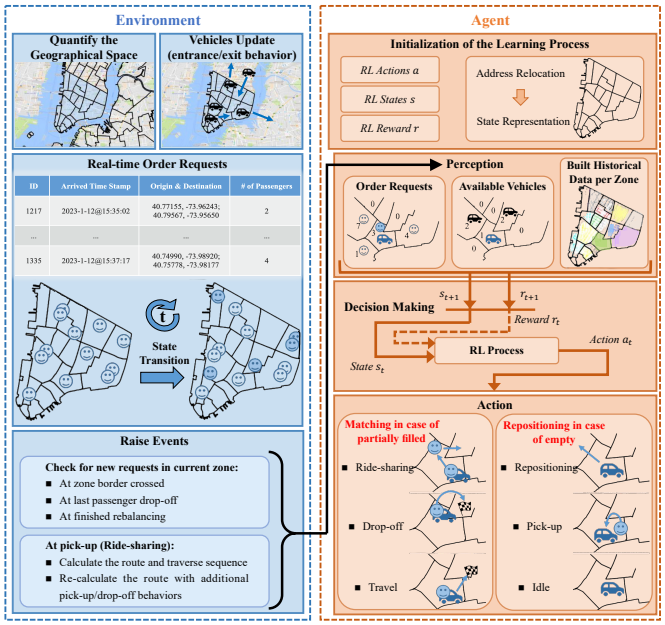

Fig. 3. **The interaction between the agent and its environment.**

In addition, a vehicle-level policy commonly trains a single agent and applies to all vehicles independently. For this setting, we can train a central agent (a central agent refers to the arbitrator, which combines the return of all agents in command arbitration.) by crowdsourcing the experience of all vehicles. Then, the central agent is applied to all other agents to generate their dispatching decisions, which means that a single experience can lead to multiple updates.

B. DT in Ride-sharing

Traditional simulation technologies convert a model into software to simulate the physical world, but lack analysis and optimisation capabilities. They solely perform offline simulations, lacking real-time and closed-loop properties (*i.e.*, the ability to interact and communicate with the real-world counterpart in a continuous feedback loop).

Fig. 1 (right side) outlines an overview of the main steps for creating a virtual DT model. Meanwhile, based on the suitability of different scenarios, we divide the application of DT in ride-sharing into four categories [13].

1) **DT for creating simulation models:** simulation environments are crucial for ride-sharing strategy development. DT enables the creation of virtual environments to simulate aspects like SD distribution, road conditions, and vehicle status in the dispatch process. This allows studying interactions among these elements without real-world engagement. Parallel simulations are commonly used to assess matching decisions and determine the optimal choice prior to implementation.

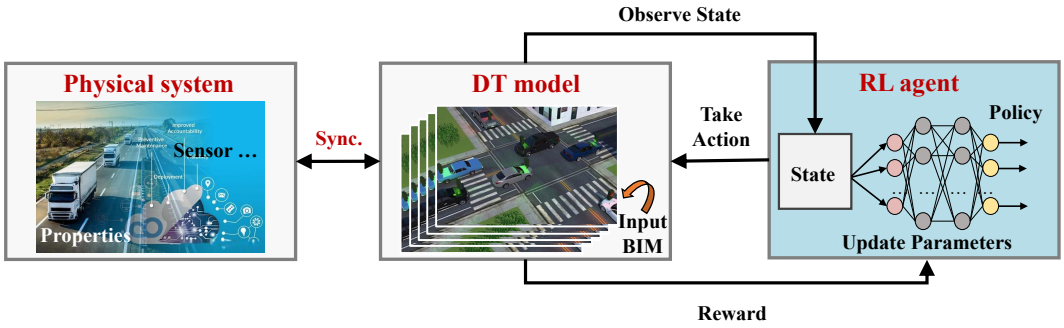

Fig. 4. **The RL and DT integration framework.**

2) **DT for supporting algorithm training:** AI algorithms are commonly employed for enhanced ride-sharing decisions. These algorithms use simulation data from the DT model for training and can be directly deployed to production. They can adjust strategies based on DT model recommendations. Continual training and adaptation of the DT model lead to improved efficiency and accuracy in the physical system.
3) **DT for updating and validation:** to ensure that an agent trained in the DT model can be directly deployed to production, the model should have high fidelity to its physical counterpart. As the system varies, the DT model has to continuously update state information and maintain synchronization with the physical system by real-time data. The data generated in the DT model will then be compared with real data to evaluate the model accuracy. This process refines the DT model and increases its fidelity over time, enabling timely adjustment and valid validation in practice.
4) **DT for prediction and routing:** as the DT model is intended to be a direct representation of the physical system, it can perform extensive computations to simulate a range of future scenarios and perform predictive analytics accordingly. This evolution facilitates the ride-sharing platform to proactively optimize its operations, make more informed decisions, and reduce waiting times for passengers and vacant time for drivers.

C. Integrating DT and RL

RL enables the system or driver to learn from data without explicit programming, which can lead to a knowledge gap. To address this gap, the DT model is integrated as the RL environment to enhance system performance. DT captures changing conditions in near real-time. It creates a virtual simulation environment with which the agent can interact. In this environment, the DT model conveys the SD distribution, traffic conditions, and other information to the RL algorithm. The RL agent then receives rewards through these interactions and accordingly updates its hyper-parameters. This process steadily improves the performance of the system, increasing its profitability and passenger satisfaction as well. For instance, the DT model can forecast passenger behaviour, allowing the RL algorithm to devise personalized pricing strategies based on their behavioural patterns.

Fig. 4 shows the RL and DT integration framework consisting of three components. This framework demonstrates how simulative data is collected and updated in the DT model. It also shows how the RL agent interacts with the data to observe states, execute actions, and receive rewards. Thereinto, the DT model serves as a dynamic learning environment that allows real-time observation of the overall road network state. Such overall observation is required for the RL agent to learn adaptive ride-sharing decisions, which in turn increases performance at the platform level rather than at an individual vehicle level. For instance, the agent may learn a repositioning strategy that dispatches vehicles while comprehensively considering the state of other vehicles and passengers. Furthermore, this learning framework is innovative and realistic. The decision learned from the DT environment has a narrower simulation-to-reality gap.

IV. Case Study: A Learning Framework for Multi-vehicle Order Dispatching at Large-scale

This part discusses a specific embodiment of order dispatching with consideration of vehicle repositioning by integrating DT and RL.

A. DT Modeling and Problem Formulation

Assisted with 3D modeling software and big data analysis, this embodiment uses DT to construct a virtual road network for continuously improving simulations. The virtual road network is updated in real-time to match its physical counterpart through input data from various IoT devices, onboard units, and satellite imagery.

Passengers with travel requirements submit orders to the centralized decision-making platform via applications on mobile clients. The platform collects order requests and dispatches them to available vehicles within each time window. Furthermore, to quantify the geographical space, the virtual road network is discretized into blocks via street intersections, as shown in step ① of **Fig. 5**, where vehicles can only pick up and drop passengers off at these intersections. Naturally, the origin and destination of an order are input in the form of the nearest street intersections.

In particular, passengers expect prompt service, and vehicles must decide whether to actively pick up new orders based on their status. After that, the routing module determines

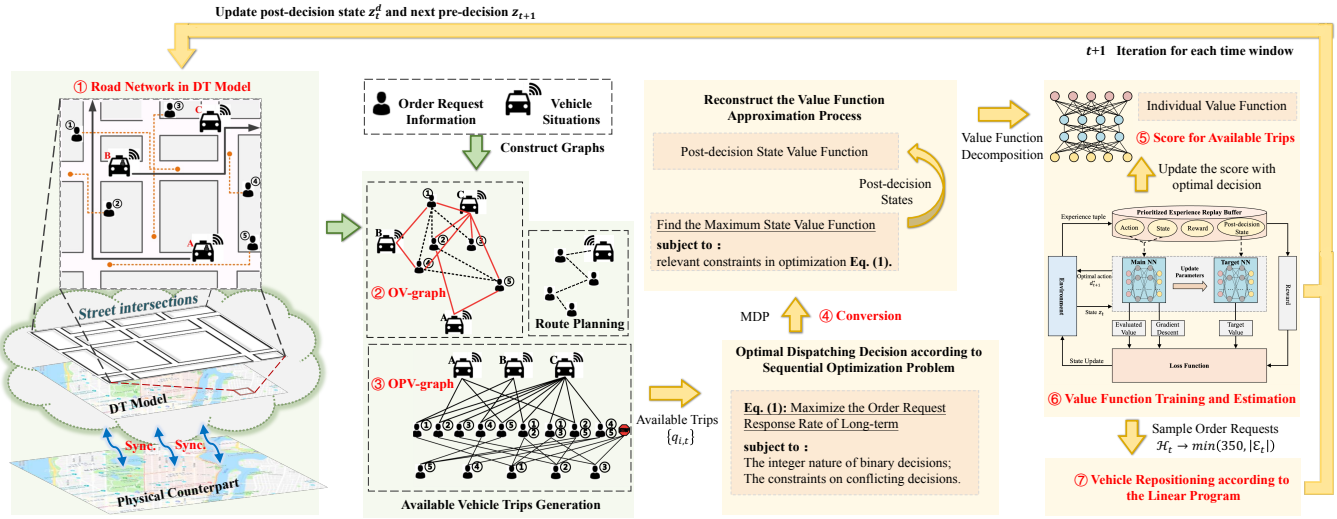

Fig. 5. **Schematic overview of the embodiment, green and yellow boxes indicate action generation and evaluation components, respectively.** ① The virtual road network in DT model. ② We construct the OV-graph to reflect the ride-sharing potential. A potential trip is available only if the clique (sub-graph) in OV-graph is closed-loop. ③ We construct the OPV-graph to generate available trips for each vehicle. We only confirm the availability of a potential trip when all of its sub-trips are available. ④ We convert the sequential optimization problem to a maximum state value function. ⑤ We offline score the individual value function of available trips under the current state via the neural network. ⑥ We make an order dispatching by estimating the optimal overall value function with neural network-based parameter approximation. ⑦ We perform vehicle repositioning to balance ride supply across geographic dimensions.

the optimal sequence of locations for the vehicle to traverse. Underlying these is a sequential decision-making problem, which corresponds to assigning an available trip (a combination of order requests) to a demand-matched vehicle under constraints. Considering the long-term (*e.g.*, several hours or a day) influences, the objective Eq. (1) in this embodiment is to maximize the order request response rate over time, while satisfying current demand and optimizing the anticipated utility.

$$\max_{x_{i,q},q_i} \lim_{\Gamma\to\infty} \frac{1}{\Gamma} \sum_{t=1}^{\Gamma} \sum_{v_{i,t}\in\mathcal{N}_t} \sum_{q_{i,t}\in\mathcal{Q}_{i,t}} \frac{x_{i,q}^t \eta_{i,q}^t}{|O_t|}. \tag{1}$$

Here, $\mathcal{N}_t$ and O_t respectively denote the set of available vehicles and order requests in time window t; $v_{i,t} \in \mathcal{N}_t$ signifies any available vehicle in time window t, with i denoting the index of the vehicle; $\eta_{i,q}^t$ is the number of order requests that vehicle $v_{i,t}$ can serve through dispatched trip $q_{i,t}$; binary decision $x_{i,q}^t \in \{0,1\}$ denotes the trip choice of vehicle $v_{i,t}$ for the available trip $q_{i,t}$. Besides, relevant constraints include the integer nature of binary decisions, and the constraints on conflicting decisions, *i.e.*, each order request can only be assigned to one vehicle.

B. MDP Definition and Available Trips Generation

Next, the virtual road network is enabled to support RL training and modification. To exploit RL in this context, we formulate the optimization problem as a MDP, where a series of agents under the time sequence model determines the order dispatching process. Each vehicle corresponds to an agent for straightforward action, reward and state transition definitions.

Accordingly, in each time window t, state z_t comprises the current demand information and vehicle situations, while action d_t is to adopt an available trip. Specifically, the action involves the available trip that an agent can execute and its decision for this available trip. Furthermore, as the optimization objective is positively correlated with the goal of an agent that tries to achieve a maximum cumulative discounted reward, reward function $r(z_t, d_t)$ is defined as the optimization objective, with punitive negative values for violating constraints.

Notably, as shown in step ②-③ of **Fig. 5**, available trips $\{q_{i,t}\}$ are generated for each agent i through a graph theory-based method. Specifically, the process proceeds in two steps. First, an order-vehicle graph (OV-graph) is constructed to reflect ride-sharing potential. Second, a graph of potential trips and corresponding pick-up vehicles (OPV-graph) is constructed to generate available trips, which explores the clique partition (the closed-loop sub-graph) in the OV-graph.

C. RL Framework Design

In the following, we expect to use the Adaptive Dynamic Programming (ADP) framework [14] to optimize the problem from a data-driven perspective, taking advantage of its large-scale decision optimization capabilities. ADP performs order dispatching through a function approximation structure, which estimates the anticipated future value of executing a particular trip with spatiotemporal dependency.

The approximate structure focuses on the agent. At each time window, the agent observes the current state and takes action based on its policy. After that, it will receive an immediate reward and consider an anticipated reward as it proceeds. Considering the long-term influences, ADP introduces a recursive state value function $V^\pi(z_t)$, which maps the state to the expected cumulative discounted reward value. In each time window t, the goal of the agent is to take an optimal action that maximizes this state value function $V^\pi(z_t)$. Consequently, the

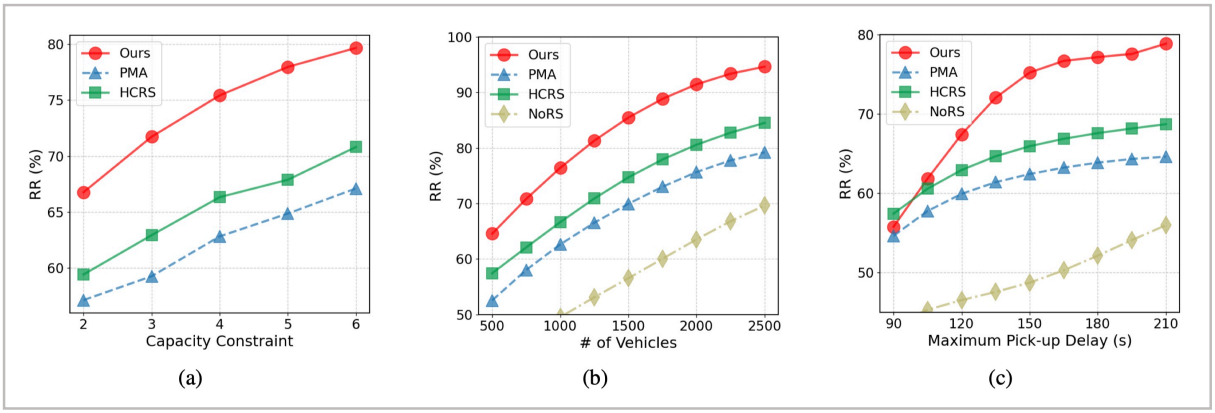

Fig. 6. (a) Order request response rate versus the capacity constraint. (b) Order request response rate versus the number of vehicles. (c) Order request response rate versus the maximum pick-up delay.

optimization objective in this embodiment can be converted to finding the maximum state value function in each time window t, as shown in step ④ of **Fig. 5**.

Moreover, the value function training and estimation process is exhibited in step ⑥ of **Fig. 5**. What needs to be clarified is that we reconstruct the Bellman update process around the post-decision states. The purpose is to avoid explicit approximation of embedded expectations. For non-linear function approximation, we convert the value function into a linear combination by a quadratic decomposition, and estimate the decomposed value function with neural network-based parameter approximation. Then, with a sample path in each episode, we update the neural network parameters and obtain optimal behaviour decisions by minimizing the loss function. In addition, to balance ride supply across geographic dimensions, vehicle repositioning is performed after each batch as well. In each time window t, we reposition all non-occupied vehicles to move to the areas of samples $\mathcal{H}_t \rightarrow min(350, |\mathcal{E}_t|)$ by a linear program (⑦ of **Fig. 5**), where $\mathcal{E}_t$ denotes the set of these non-occupied vehicles.

Afterwards, the DT-based emulator updates post-decision and next pre-decision states, while evaluating model accuracy by comparing them with real data. To continue the process to the next time window, the aforementioned value function approximation is repeated until a finite time horizon within an episode. Agents can adjust their strategies based on DT model recommendations, and the stable desired value function is then deployed to production.

D. Simulation and Results

Using real-world data from New York City taxi daily operations [15], we conduct quantitive performance analyses for order dispatching by populating ride-sharing vehicles over the Manhattan borough. Unless specified, we use 1000 vehicles with a 4-passenger capacity constraint. The initial maximum pick-up delay is 150s, while the maximum detour delay is twice that value. In subsequent analysis, we vary these values to affect instance sizes for comparison. We process multiple order requests in batch sequences with a time window size of 40s. The hyper-parameters include an initial learning rate of 0.015 and a discount factor of 0.9, respectively. All experiments are conducted in Python 3.8 and tested on a processor (AMD®Threadripper™PRO5995WX@2.70GHz).

For performance comparison, we compare our method with three baselines: PMA, HCRS, and NoRS. PMA and HCRS are greedy and heuristic methods, while NoRS is a particular case of our method, dispatching only one order request to a vehicle. We compare the order request response rate across various vehicle capacity constraints, vehicle quantities, and maximum pick-up delays. **Fig. 6** illustrates the results, demonstrating that our method performs best among all baselines. Notably, a particular observation in **Fig. 6(c)** is worth highlighting, where our method initially performs weaker than HCRS. This discrepancy can likely be attributed to stringent delay constraints. Our method will consistently outperform it with enough search for available action generation.

V. Open Challenges

Finally, we aim to highlight potential challenges that can bridge theory and practice in this topic.

A. Scalability

For integrated ride-sharing platforms, scalability is a concern. Existing RL-based solutions struggle to keep up with the increasing complexity and size of the system, especially when dealing with multiple interacting objects. Meanwhile, storing the vast volumes of data generated by physical objects in DT components necessitates expensive storage infrastructure.

B. Interpretability

Another concern for the platform is to enhance trust in its decision-making processes, which boosts revenue and market share. However, achieving model interpretability has long been a hurdle for implementing RL in DT-empowered systems. The integration of DT and RL can be complex and opaque, making system participants struggle to grasp how decisions are made. To gain the trust and acceptance, it is essential to ensure these decisions are transparent and interpretable.

C. Real-time and Accuracy

To maintain the responsiveness of the DT model, accurate real-time data input is essential for technology implementation. However, handling the substantial volumes of data generated can be challenging due to potential noise, incompleteness, or bias. This challenge becomes even more pronounced when managing high-velocity data streams from IoT devices.

D. Security and Privacy

The final concern is the potential for security breaches and privacy violations. The system relies heavily on the collection, processing, and storage of data, including personal and sensitive information. The vulnerability of this data to cyber-attacks and unauthorized access poses risks to the security and privacy of ride-sharing shareholders.

VI. Conclusion

In this article, we conducted a comprehensive overview of the integration of DT and RL in ride-sharing from a system-level perspective. Specifically, we first focused on a typical ride-sharing application under the umbrella of DT and RL, and explained their origins in this context. Then, we delved into how these two technologies can be integrated for ride-sharing system optimization. The potential advantages and impact of their integration were elaborated holistically. Subsequently, we discussed a specific embodiment of this integration for large-scale multi-vehicle order dispatching. Finally, we strived to shed light on potential challenges, which may facilitate the transformation of this topic from theory to practice.

References

[1] S. C. K. Chau, S. Shen, and Y. Zhou, "Decentralized Ride-Sharing and Vehicle-Pooling Based on Fair Cost-Sharing Mechanisms," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 1936-1946, 2022.

[2] X. Tang, et al., "A deep value-network based approach for multi-driver order dispatching," *in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, pp. 1780–1790, 2019.

[3] Z. Qin, H. Zhu, and J. Ye, "Reinforcement learning for ridesharing: An extended survey," *Transp. Res. C, Emerg. Technol.*, vol. 144, pp. 1-28, Nov. 2022.

[4] Y. Tong, et al., "Combinatorial Optimization Meets Reinforcement Learning: Effective Taxi Order Dispatching at Large-Scale," *IEEE Trans. Knowl. Data Eng.*, Nov. 2021.

[5] V. L. Nguyen, et al., "Toward the Age of Intelligent Vehicular Networks for Connected and Autonomous Vehicles in 6G," *IEEE Network*, vol. 37, no. 3, pp. 44–51, 2023.

[6] J. Zheng, et al., "Digital Twin Empowered Heterogeneous Network Selection in Vehicular Networks With Knowledge Transfer," *IEEE Trans. Veh. Technol.*, vol. 71, no. 11, pp. 12154–12168, Nov. 2022.

[7] Z. Xiong, et al., "Deep Reinforcement Learning for Mobile 5G and Beyond: Fundamentals, Applications, and Challenges," *IEEE Veh. Technol. Mag.*, vol. 14, no. 2, pp. 44–52, 2019.

[8] H. Zhou, et al., "Distributed Multi-Agent Reinforcement Learning for Cooperative Edge Caching in Internet-of-Vehicles," *IEEE Trans. Wire. Commun.*, early access, 2023, doi: 10.1109/TWC.2023.3272348.

[9] B. Tan, et al., "Toward a Future Network Architecture for Intelligence Services: A Cyber Digital Twin-Based Approach," *IEEE Network*, vol. 36, no. 1, pp. 98-104, 2022.

[10] L. Zhang, et al., "A taxi order dispatch model based on combinatorial optimization," *in Proc. ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, pp. 2151-2159, 2017.

[11] P. Cheng, H. Xin, and L. Chen, "Utility-aware ridesharing on road networks," *in Proc. ACM Int. Conf. Manage. Data*, pp. 1197-1210, 2017.

[12] X. Bei, and S. Zhang, "Algorithms for trip-vehicle assignment in ridesharing," *in Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, pp. 3-9, 2018.

[13] W. Xiang, et al., "Digital Twin Empowered Industrial IoT Based on Credibility-weighted Swarm Learning," *IEEE Trans. Indus. info.*, early access, 2023, doi: 10.1109/TII.2023.3264289.

[14] I. Hull, "Approximate dynamic programming with post-decision states as a solution method for dynamic economic models," *Journal of Economic Dynamics and Control*, vol. 55, pp. 57-70, 2015.

[15] NYC Taxi & Limousine Commission. (2020). *New York City's Taxi Trip Data.* [Online]. Available: https://www.nyc.gov/site/tlc/about/ tlc-trip-record-data.page.

Biographies

KAI JIANG [S'21] (kai.jiang@whu.edu.cn) is currently pursuing the Ph. D. degree in cyberspace security at Wuhan University, China. His research interests include Deep Reinforcement Learning, Intelligent Transportation, Internet of Vehicles, and Mobile Edge Computing.

YUE CAO [SM'18] (yue.cao@whu.edu.cn) received the Ph.D. degree from the Institute for Communication Systems (ICS) formerly known as Centre for Communication Systems Research, University of Surrey, Guildford, U.K., in 2013. Further to his PhD study, he had conducted a Research Fellow with the University of Surrey, and academic faculty with Northumbria University, U.K., Lancaster University, U.K., and Beihang University, China. He is currently a Professor with the School of Cyber Science and Engineering, Wuhan University, Wuhan, China. His research interests include E-Mobility, V2X, and Edge Computing.

ZHENNING WANG [S'21] (zn.wang115@whu.edu.cn) is currently pursuing the Ph. D. degree in cyberspace security at Wuhan University, China. His research interests include Game Theory, Crowdsensing, and Mobile Edge Computing.

HUAN ZHOU [M'14] (zhouhuan117@gmail.com) received his Ph. D. degree from the Department of Control Science and Engineering at Zhejiang University. He was a visiting scholar at the Temple University from Nov. 2012 to May, 2013, and a CSC supported postdoc fellow at the University of British Columbia from Nov. 2016 to Nov. 2017. Currently, he is a professor at the College of Computer, Northwestern Polytechnical University. His research interests include Mobile Social Networks, Opportunistic Mobile Networks, and Edge Computing.

HONG ZHU (zhuhong@cnis.ac.cn) received the Ph.D. degree from the School of Electronics Engineering and Computer Science, Peking University, Beijing, China, in 2009. She is currently an associate researcher of China National Institute of Standardization, Beijing, China. Her research interests include Standardization, Big Data, and Knowledge Management.

Zhi Liu [SM'19] (liu@ieee.org) received his Ph.D. in informatics in National Institute of Informatics. He is currently an Associate Professor at The University of Electro-Communications. His research interest includes video network transmission and mobile edge computing.