Learning to Solve Multiple-TSP With Time Window and Rejections via Deep Reinforcement Learning

Rongkai Zhang, Cong Zhang, Zhiguang Cao, Wen Song, Puay Siew Tan, Jie Zhang, Bihan Wen, *Member, IEEE*, and Justin Dauwels

***Abstract*— We propose a manager-worker framework (the implementation of our model is publically available at: https://github.com/zcaicaros/manager-worker-mtsptwr) based on deep reinforcement learning to tackle a hard yet nontrivial variant of Travelling Salesman Problem (TSP), *i.e.,* multiple-vehicle TSP with time window and rejections (mTSPTWR), where customers who cannot be served before the deadline are subject to rejections. Particularly, in the proposed framework, a manager agent learns to divide mTSPTWR into sub-routing tasks by assigning customers to each vehicle via a Graph Isomorphism Network (GIN) based policy network. A worker agent learns to solve sub-routing tasks by minimizing the cost in terms of both tour length and rejection rate for each vehicle, the maximum of which is then fed back to the manager agent to learn better assignments. Experimental results demonstrate that the proposed framework outperforms strong baselines in terms of higher solution quality and shorter computation time. More importantly, the trained agents also achieve competitive performance for solving unseen larger instances.**

***Index Terms*— Travelling salesman problem, graph neural network, deep reinforcement learning.**

I. Introduction

DEEP reinforcement learning (DRL) has been garnering considerable interests for solving various NP-hard combinatorial optimization problems (COPs) [1], [2], [3], [4]. Among them, due to its high practical value, the Travelling Salesman Problem (TSP, including the Vehicle Routing Problem) is receiving particularly increasing attention. As promising modern alternatives, the DRL based methods have the potential to not only consume shorter inference time compared with the exact algorithms while producing near-optimal solutions, but also circumvent hand-crafted rules in the conventional heuristics. By virtue of those desirable capabilities, recently proposed DRL-based methods achieved much success in solving the general TSP [5], [6].

Recently, a number of attempts have been performed to leverage DRL to tackle more challenging variants of TSP. An emerging trend is to incorporate temporal constraints such as time window [4], [7], [8], as customers always prefer to be served within a desirable time period, earlier or later than which may cause dissatisfaction. The other notable trend is to extend the single vehicle (or salesman) to multiple ones [9], [10], [11], given that it is more common to dispatch a fleet of vehicles to serve customers in real life. Although some seminal works have been delivered to address the two types of variants respectively, rare studies that exploit the DRL method to simultaneously cope with time window and multiple vehicles have been conducted. Note that, it is nontrivial to investigate the combination of the two variants with the DRL method. On one hand, this combined variant is more in line with the real-world scenarios compared with the separate ones. On the other hand, this combined variant is arduous to be solved, since simply integrating the respective DRL methods does not guarantee desirable performance, especially in light of the NP-hard nature of the problem.

In this paper, we aim to close this gap by proposing a manager-worker framework based on DRL to tackle this harder yet more practical variant of TSP, namely multiple-vehicle TSP with time window and rejections (mTSPTWR). Since time window might be spontaneously prescribed by customers without any negotiations, it is possible that some of those temporal constraints would not be satisfied in a sub-tour. In mTSPTWR, customers who cannot be served before the deadline are subject to rejections. The goal is to find a route for each vehicle such that a hybrid cost consisted of tour length and rejection rate is minimized for the worst sub-tour.

To solve this challenging problem, in our proposed framework, a *manager* agent learns to divide mTSPTWR into sub-routing tasks, and a *worker* agent learns to solve each resulting sub-routing task. Specifically, we propose a novel

Manuscript received 6 September 2021; revised 17 March 2022 and 30 August 2022; accepted 12 September 2022. This work was supported in part by the Agency for Science Technology and Research Career Development Fund under Grant 222D8235 and Grant C222812027; in part by the IEO Decentralized GAP Project of Continuous Last-Mile Logistics (CLML) at the Singapore Institute of Manufacturing Technology under Grant I22D1AG003; in part by the Rapid-Rich Object Search (ROSE) Laboratory, Nanyang Technological University, Singapore; in part by the Ministry of Education, Singapore, under its Academic Research Fund Tier 1 under Project RG61/22; in part by the Start-Up Grant; in part by the National Natural Science Foundation of China under Grant 62102228; and in part by the Shandong Provincial Natural Science Foundation under Grant ZR2021QF063. The Associate Editor for this article was S. M. Easa. *(Rongkai Zhang and Cong Zhang contributed equally to this work.) (Corresponding author: Wen Song.)*

Rongkai Zhang and Bihan Wen are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: rongkai002@e.ntu.edu.sg; bihan.wen@ntu.edu.sg).

Cong Zhang and Jie Zhang are with the School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798 (e-mail: cong030@e.ntu.edu.sg; zhangj@ntu.edu.sg).

Zhiguang Cao and Puay Siew Tan are with the Singapore Institute of Manufacturing Technology (SIMTech), Agency for Science Technology and Research (A*STAR), Singapore 138632 (e-mail: zhiguangcao@outlook.com; pstan@simtech.a-star.edu.sg).

Wen Song is with the Institute of Marine Science and Technology, Shandong University, Qingdao 266237, China (e-mail: wensong@email.sdu.edu.cn).

Justin Dauwels is with the Department of Microelectronics, Faculty EEMCS, Technische Universiteit Delft, 2628 CD Delft, The Netherlands (e-mail: j.h.g.dauwels@tudelft.nl).

Digital Object Identifier 10.1109/TITS.2022.3207011

Graph Isomorphism Network (GIN) as the policy network, which is leveraged by the manager agent to assign customers to different vehicles by exploiting the selected output from the worker agent, while the worker agent relies on a self-attention based encoder-decoder policy network to minimize the tour length and rejection rate for each vehicle in regards of the assigned customers and time window constraints. In doing so, multiple vehicles and time window constraints are naturally decoupled and handled by the respective agents. The extensive experimental results based on randomly generated instances reveal that our proposed method significantly outperforms three strong metaheuristic baselines and an adapted DRL baseline, all of which are carefully tuned. It is also demonstrated that our method generalizes considerably well to much larger instances that are unseen during training. In short, the contributions of this paper are summarized as follows:

- Targeting at more realistic properties, we present and study a practical yet challenging variant of routing problem, *i.e.,* mTSPTWR, which handles time windows, rejections, and multiple vehicles.
- We propose the first DRL-based manager-worker framework to solve mTSPTWR, which encompasses a management agent for customer assignment based on a Graph Neural Network (GNN), and a worker agent for routing based on a self-attention encoder-decoder.
- Extensive experiments confirm that our proposed method delivers superior performance in terms of higher solution quality and shorter computation time compared to the state-of-the-art baselines and can be generalized to large-scale instances. A detailed ablation study has also justified the effectiveness of our design.

II. Related Work

As a classical family of COPs, routing problems have been investigated for many decades. Various methods, including conventional (meta)heuristic methods and machine learning based methods, have been proposed. Among them, DRL based method is arousing growing interests due to its desirable generalization capability and independence of ground-truth label. Bello *et al.* [12] first formulate TSP as a Markov Decision Process (MDP) solved by a DRL algorithm, where the policy is parameterized using a Seq2Seq based encoder-decoder architecture. Most of its following works apply a similar architecture while designing new respective encoders or decoders to ameliorate the performance. For example, realizing that the input sequence should be irrelevant to the instance representation, Nazari *et al.* [13] replace the LSTM encoder with element-wise projections so that the updated embeddings after state changes can be effectively computed. Inspired by the Transformer model [14], Deudon *et al.* [15] and Kool *et al.* [16] propose two self-attention based frameworks for TSP co-currently, which both lead to significant performance boost. Other notable works include integrating local search with DRL [17] for TSP and learning to generalize to large TSP instances [18]. Although some of them have considered TSP with constraints, e.g., the capacities of vehicles [16], [17], they are relatively simple and usually handled by trivial rules.

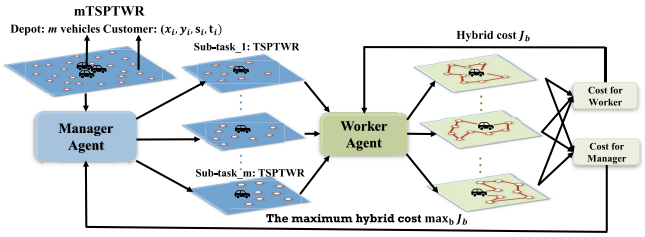


Fig. 1. **Manager-worker framework for solving mTSPTWR.** The manager agent learns to assign customers to different vehicles, and the worker agent learns to solve the resulting sub-routing tasks.

Very recently, some attempts based on deep (reinforcement) learning have started to tackle TSP (or VRP) with harder yet more realistic settings. Kaempfer and Wolf [9] introduce a supervised learning model for TSP with multiple vehicles. However, it requires sufficient labelled training data, which is computationally expensive to be acquired. To circumvent labelled training data, Hu *et al.* [10] propose a DRL based agent combined with heuristic solvers, e.g., Or-Tools [19] to solve TSP with multiple vehicles. Though the cooperation among the vehicles is considered, the usage of classical solver may limit its application to other variants. Besides multiple vehicles, several works focus on handling some other common yet hard constraints, such as time window. Ma *et al.* [20] explicitly consider the single-vehicle TSP with time windows, *i.e.,* TSPTW (first defined in [21]), by exploiting a hierarchical graph pointer network (GPN). Gao *et al.* [4] leverage DRL to learn local-search heuristics that iteratively improve the solution quality of VRP (Vehicle Routing Problem) and VRPTW (Vehicle Routing Problem with Time Window). Chen *et al.* [7] propose a DRL based framework to learn an efficient large neighbor searching heuristic for VRPTW. Different from the above works, Zhang *et al.* [22] consider a variant of TSPTW from a more practical view, which allows rejecting the nodes (customers), *i.e.,* TSPTWR. It is natural since the time window may conflict with each other, and a feasible solution that satisfies all customers may not exist. To solve this problem, they propose a DRL based framework combined with backtracking to post-process a solution to TSP in ways that it will become a desirable feasible solution to TSPTWR. To our knowledge, although some success has been achieved for DRL based works to tackle TSP with multiple vehicles or time window separately, none of them is able to effectively handle the combined variant, which is much harder yet nontrivial.

III. Proposed Framework for mTSPTWR

In this section, we first present the definition of mTSPTWR, then elaborate the rationale of the proposed manager-worker framework.

A. Problem Definition

In mTSPTWR, a fleet of vehicles are dispatched to serve customers scattered at different locations. As a common and practical property in the variants of TSPTW, customers have their respective time window for being served. However, the customers who cannot be served before the deadline

are subject to rejections in mTSPTWR. The general goal is to minimize the total tour length, while serving as many customers as possible. In our setting, an mTSPTWR instance I is characterized by a set of n customers $C = \{c_1, \ldots, c_n\}$ and m ($m > 0$) identical vehicles (or salesmen). Each customer $c_i \triangleq (x_i, y_i, s_i, t_i)$ is associated with a 2D location (x_i, y_i) and a time window (s_i, t_i), where s_i and t_i ($s_i \leq t_i$) denote the start and terminate of the time window, respectively. The depot (considered as a *dummy customer*) houses the m vehicles and is represented as $d \triangleq (x_d, y_d, 0, \infty)$. Departing from the depot, each vehicle serves a subset of customers in a single sub-tour and returns to the depot finally. Pertaining to customer c_i, arriving earlier than s_i causes waiting, and arriving later than t_i renders it rejected to be served.

We achieve the goal of mTSPTWR by minimizing a hybrid cost of length and rejection rate for the worst sub-tour, where the rejection rate for a sub-tour is defined as the quotient of the number of rejected customers and the number of assigned customers. Formally, let $r_b (1 \leq b \leq m)$ denote the sub-tour for vehicle b with length $l(r_b)$ and rejection rate $rej(r_b)$, and then the cost for vehicle b is denoted as $J_b \triangleq l(r_b) + \beta \cdot rej(r_b)$, where β is a *penalty coefficient* to control the rejection rate. The usage of a hybrid cost can mitigate the ambiguity of a sole cost. For instance, solely minimizing rejection rate may result in solutions with different tour length, while using a hybrid cost can further select the ones with less tour length out. Accordingly, the objective of mTSPTWR is defined as:

$$\min \left(\max_b J_b \right). \tag{1}$$

Note: Although we mainly optimize the above min max objective, we will demonstrate that our method is not restricted to particular objectives. As an example, we will show that our method can learn to minimize the combined overall rejection rate and tour length for all sub-tours rather than the combination of that of the worst sub-tour. Nevertheless, here we argue that the min max objective is more customer-friendly, as the level of service [23] of each vehicle can be guaranteed with a higher lower-bound. A more detailed discussion about different objectives and the experiment results for overall costs are presented in Appendix A. On the other hand, although mTSPTWR is partially related to multiple TSPTW (mTSPTW) and multiple prize-collection TSP (mPCTSP), they are still obviously different from each other. Firstly, in mTSPTW, the instances are generated with the assumption that all the customers [24] can be served feasibly and the goal is usually to minimize the tour length. It may distort the real-life scenarios where we have to reject a number of requests to pursue desirable balance between the rejection rate and the tour length. Secondly, although 'rejections' are also considered in mPCTSP, they are not triggered by the violation of the feasibility constraints, *i.e.,* time window.

B. Proposed Framework

To efficiently solve mTSPTWR, we propose a DRL based manager-worker framework that decouples the problem into upper-level and lower-level tasks, respectively, as depicted in Fig. 1. Regarding the upper-level task, a manager agent learns to divide customers into subgroups and then assign them to different vehicles. Regarding the lower-level task, a worker agent learns to route for each vehicle in consideration of the assigned customers and the hybrid cost. The (selected) outputs from the lower-level task are adopted to train both agents to help optimize the objective in Eq. (1). Algorithm 1 summarizes the whole inference process of our proposed framework for solving the mTSPTWR problem.

Algorithm 1 Inference Process of the Proposed Framework

Load: Manager_agent(), Worker_agent().
Input: One instance $\mathcal{G}_I$.
Output: The sub-tours $\{r_1, \ldots, r_m\}$ and the corresponding cost $\{J_1, \ldots, J_m\}$ for all m vehicles.

***Customer assignment process*:**
  $\{\mathcal{G}_1, \ldots, \mathcal{G}_m\}$ = Manager_agent($\mathcal{G}_I$).
***Subtour routing process*:**
for $b = 1 : m$ **do**
  r_b, J_b = Worker_agent($\mathcal{G}_b$).
end for

Pertaining to the manager agent, we leverage a Graph Neural Network (GNN) combined with self-attention to parameterize the policy, which explicitly models each vehicle based on learnt graph embeddings and then assigns each customer to a vehicle. Pertaining to the worker agent, we exploit a self-attention based encoder-decoder model that is similar to [22] to parameterize the policy. The details are as follows.

1) The Manager Agent: The process of customer assignment by the manager agent can be formulated as a one-step MDP [25]. Given a problem instance I sampled from certain distribution $p(I)$, *i.e.,* $I \sim p(I)$, we represent the unique *state* s_I as a fully connected graph $\mathcal{G}_I = (V_I, E_I)$ with nodes set $V_I = C \cup \{d\}$ and edges set $E_I = \{(v, u) | \forall v, u \in V_I\}$. The *action* a_{s_I} is to assign customers to different vehicles, thus the *action space* $\mathcal{A}(s_I)$ ($a_{s_I} \in \mathcal{A}(s_I)$) refers to all combinations of customer-vehicle matches $\mathcal{A}(s_I) \subseteq \mathcal{P}(C)$ with $\mathcal{P}(C)$ denoting the power set of all customers for I. The corresponding *reward* is defined as $R_M(a_{s_I}, s_I) = -\max_b(J_b), \forall 1 \leq b \leq m$, which is fed back by the worker agent and refers to the sub-tour with the maximum hybrid cost. At s_I, a stochastic policy $\pi(a_{s_I}|s_I)$ outputs a distribution over $\mathcal{A}(s_I)$, from which the action is sampled accordingly. Note that, the manager agent outputs the overall distribution in one shot rather than in a sequential fashion. During training, an instance will be discarded once exploited, and new instances will be sequentially generated on-the-fly to update the policy until converged.

The policy $\pi(a_{s_I}|s_I)$ for the manager agent is parameterized by $\pi_{\theta_M}(a_{s_I}|s_I)$ with trainable parameters θ_M. To learn informative representation and facilitate the matching between customers and vehicles, we design a policy network as depicted in Fig. 2. Firstly, we leverage GNN to embed the state s_I including customers and depot. Particular in this paper, we adopt GIN [26] for graph embedding which is known as a strong GNN variant with discriminative power to learn representation over graphs. The formulation of GIN layers,

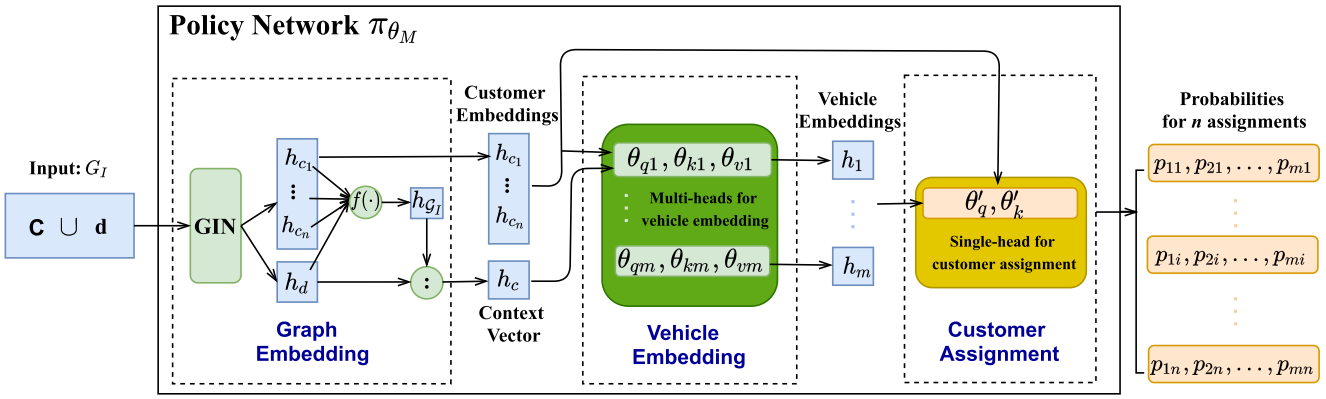


Fig. 2. **Architecture of manager agent policy network π_{θ_M}.** It mainly consists of three components, namely, *graph embedding*, *vehicle embedding*, and *customer assignment*, respectively. The output is the distribution for all customers c_i being assigned to different vehicles.

Algorithm 2 Customer Assignment Process

Input: $\mathcal{G}_I$ with feature $h^{(0)}(v) \in \mathbb{R}^{n_0}, \forall v \in V_I$.
Parameter: GIN parameters: $\theta_{t=0} \in \mathbb{R}^{n_0 \times n_G}, \theta_{t>0} \in \mathbb{R}^{n_G \times n_G}$; vehicle embedding network parameters: $\theta_{qb} \in \mathbb{R}^{2n_G \times n}$, $\theta_{kb}, \theta_{vb} \in \mathbb{R}^{n_G \times n}$; customer assignment network parameters: $\theta'_q \in \mathbb{R}^{n \times n'}, \theta'_k \in \mathbb{R}^{n_G \times n'}$.
Output: Matching between vehicles and customers.

Graph Embedding using L GIN layers:
for $l = 1 : L$ **do**
  $h_v^{(l)} = \text{GIN}_{\theta_l}^{(l)}(h_v^{(l-1)})$.
end for
$h_v = \sum_{l=1}^{L} h_v^{(l)}$; $h_{\mathcal{G}_I} = f(h_v^{(l)}) = \frac{1}{|V|}\sum_{l=1}^{L}\sum_v h_v^{(l)}$.
Vehicles Embedding using multi-head self-attention:
for $b = 1 : m$ **do**
  $q_b = \theta_{qb} h_c$, where the context vector $h_c = (h_{G_I} : h_d)$ with (:) denoting the concatenation operator.
  for $c_i \in C$ **do**
    $k_{bi} = \theta_{kb} h_{c_i}$, $v_{bi} = \theta_{vb} h_{c_i}$.
  end for
  $h_b = \sum_i w_{bi} v_{bi}$, $w_{bi} = \frac{e^{u_{bi}}}{\sum_j e^{u_{bj}}}$, $u_{bi} = \frac{q_b^T k_{bi}}{\sqrt{n}}$.
end for
Customer Assignment using single-head self-attention:
for $c_i \in C$ **do**
  for $b = 1 : m$ **do**
    $q'_b = \theta'_q h_b$, $k'_{bi} = \theta'_k h_{c_i}$, $u'_{bi} = \frac{q_b'^T k'_{bi}}{\sqrt{n'}}$, $s_{bi} = \tanh u'_{bi}$, $p_{bi} = \frac{e^{s_{bi}}}{\sum_b e^{s_{bi}}}$.
  end for
  Assign c_i to $\hat{b}_{c_i}$ by sampling, *i.e.,* $\hat{b}_{c_i} \sim (p_{1i}, \ldots, p_{mi})$.
end for

i.e., $\text{GIN}(h_v^{(k-1)})$, is expressed as follows:

$$\text{MLP}_{\theta_k}^{(k)}\left(\left(1+\epsilon^{(k)}\right)\cdot h_v^{(k-1)} + \sum\nolimits_{u\in\mathcal{N}(v)} h_u^{(k-1)}\right), \quad (2)$$

where $h_v^{(k)}$ is the representation of node v at iteration k and $h_v^{(0)}$ refers to its raw features for input, $MLP_{\theta_k}^{(k)}$ is a Multi-Layer Perceptron (MLP) with parameter θ_k for iteration k followed by batch normalization [27], ϵ is an arbitrary number that can be learned, and $\mathcal{N}(v)$ is the neighbourhood of v. After K iterations of updates, a global representation for the entire graph can be obtained using a pooling function L that takes as input the embeddings of all nodes and outputs a p-dimensional vector $h_{\mathcal{G}} \in \mathbb{R}^p$ for $\mathcal{G}$. Here we use average pooling, *i.e.,* $h_{\mathcal{G}} = L(\{h_v^K : v \in V\}) = 1/|V| \sum_{v\in V} h_v^K$.

Secondly, we exploit a *multi-head* self-attention architecture [14] to embed vehicles where each head corresponds to a vehicle. We argue that, although m vehicles are identical, treating them homogeneously (via parameters sharing) may cause difficulty for managing them. To model this individuality of vehicles, we stipulate that the multiple heads are mutually independent, which is also justified in the ablation study. Finally, given customer and vehicle embeddings, a *single-head* self-attention architecture is employed to model a centralized assignment policy. The procedure of customer assignment by the manager agent is summarized as Algorithm 2.

Meanwhile, the policy $\pi_{\theta_M}(a_{s_I}|s_I)$ is trained using the REINFORCE [28] algorithm. Specifically, we maximize the expected reward obtained with $\pi_{\theta_M}(a_{s_I}|s_I)$ given any instance I as follows,

$$\mathbb{E}_{I\sim p(I)}[\sum_{a_{s_I}}(R_M(a_{s_I}, s_I) - B(I))\pi_{\theta_M}(a_{s_I}|s_I)], \quad (3)$$

where the reward $R_M(a_{s_I}, s_I)$ is acquired from the worker agent, and $B(I)$ is a baseline to reduce variance during training [16]. With such an objective, the manager agent will learn assignments that help reduce the hybrid cost of the worst sub-tour. In practice, we adopt a *batch* of instances from $p(I)$ and rollout π_{θ_M} on them to estimate the objective in Eq. (3), which is then used to update the policy.

2) The Worker Agent: Once decomposed by the manager agent, the mTSPTWR becomes a TSPTWR for the respective vehicle. Then, the worker agent should construct the route by finding a sub-tour for each vehicle such that the hybrid cost consisted of tour length and rejection rate is minimized for each sub-tour. In light of the recently reported competitive performance, DRL based method is more desirable for the worker agent to solve the TSPTWR [22]. However, the size of customers assigned to a vehicle always varies, and it would be substantially challenging to *jointly* train the manager agent and worker agent. To alleviate this issue, we train them in a two-stage manner. In the first stage, the worker agent is trained alone with its own cost. In the second stage, the well-trained

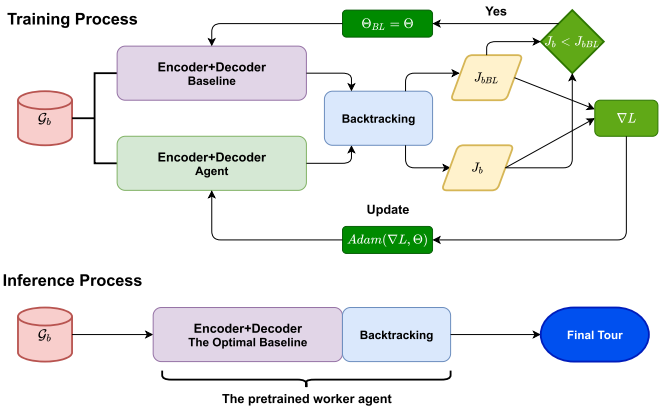


Fig. 3. The diagram of the worker agent [22].

Algorithm 3 Training Process of Worker Agent

Input: Instances S, Batch Size S_{bs}, Training Epoch E, Update Threshold $\alpha > 0$.
Parameter: Θ, Θ^{BL}.
Output: Θ^* for the optimal baseline/agent.

for epoch = 1:E **do**
  for all S **do**
    Sample S_{bs} instances from S
    for $S_i \in S_{bs}$ **do**
      $H_i = \text{Encoder}_{\Theta}(S_i)$,$H_{BLi} = \text{Encoder}_{\Theta^{BL}}(S_i)$.
      $r_i = \text{Decoder}_{\Theta}(H_i)$,$r_{BLi} = \text{Decoder}_{\Theta^{BL}}(H_i)$.
      $J_{bi} = \text{Backtracking}(r_i)$.
      $J_{bBLi} = \text{Backtracking}(r_{BLi})$.
    end for
    $\nabla L \propto \sum_1^{S_{bs}} (J_{bi} - J_{BLi}) \nabla_{\Theta} log\, P_i(r_{bi})$.
    $\Theta = \text{Adam}(\Theta, \nabla L)$.
    if $J_{bi} - J_{bBLi} < -\alpha$ **then**
      $\Theta^{BL} = \Theta_i$.
    end if
  end for
end for

worker agent is fixed and the output is adopted to calculate the cost for the manager agent to learn better assignments. In doing so, we not only guarantee stable training performance, but also enable flexibility on selecting routing algorithm that is deployed in a plug-and-play manner for the worker agent.

In our framework, we exploit a self-attention based encoder-decoder model that is similar to [22] to parameterize policy. Fig. 3 illustrates the training and inference process of the exploited model. We represent a TSPTWR instance for a single vehicle b as a fully connected graph $\mathcal{G}_b = (V_b, E_b)$, with $V_b = \{C_{\psi_b}\} \cup \{d\}$. Passing $\mathcal{G}_b$ through a self-attention based encoder-decoder policy network $\pi_{\Theta}(r|\mathcal{G}_b)$, a tour r (may be infeasible) for serving all the customers in $\mathcal{G}_b$ is generated. Then r is mapped to r_b (a guaranteed feasible solution) by a *backtracking* procedure [22]. The resulting hybrid cost for vehicle b is given as $J_b = l(r_b) + \beta \cdot rej(r_b)$. The negative hybrid cost $-J_b$ is applied as the *reward* for the worker agent. REINFORCE [28] with a rollout baseline is utilized to enable a faster training process with lower variance. During training, the parameter Θ^{BL} is updated to Θ, if J_θ is less than J_{BL}. This baseline is not only exploited to adjust Θ, but also able to reinforce the good solutions by increasing the probability. The final parameter Θ^* for inference is set to Θ^{BL}. Algorithm 3 summarizes the training process of the worker agent and Algorithm 4-6 detail the respective key components, *i.e., encoder*, *decoder*, and *backtracking* in the worker agent. All the algorithms are written as pseudo code in python style.

Algorithm 4 Encoder

Input: Features for nodes $x_i \in V_b$, $i = (1, \ldots, n)$, Normalization Constant d_k, Layers Number N.
Output: Embedding of the instance H=$\{h_i^{(N)}, \bar{h}^{(N)}\}$.

Embed to high dimension: $h_i^{(0)} = W^x x_i + b_x$.
for l = 1:N **do**
  Compute key k_i, value v_i and query q_i for each node: $k_i = W_l^K h_i^{(l-1)}$, $v_i = W_l^V h_i^{(l-1)}$, $q_i = W_l^Q h_i^{(l-1)}$.
  Compute the compatibilitires: $u_{ij} = \frac{q_i^T k_j}{\sqrt{d_k}}$, $i \neq j$.
  Compute the attention weights using softmax: $a_{ij} = \frac{e^{u_{ij}}}{\sum_j e^{u_{ij}}}$.
  Output from the multi head attention sublayer: $h_{i'}^{(l)} = \sum_j a_{ij} v_j$.
  Output from feed-froward sublayer: $h_{i''}^{(l)} = W_l^{ff_1} \text{ReLu}(W_l^{ff_0} h_i^{(l-1)} + b_l^{ff_0}) + b_l^{ff_1}$.
  Combine and Batch Normalize (BN): $h_i^{(l)} = \text{BN}(h_{i'}^{(l)} + h_{i''}^{(l)})$.
end for
Obtain $h_i^{(N)}$ and $\bar{h}^{(N)} = \frac{\sum_i h_i^{(N)}}{n}$.

IV. Experiments and Results

In this section, we conduct extensive experiments to evaluate our method. Firstly, we compare our method with three strong metaheuristics on problems of different sizes and difficulties. In particular, two types of mTSPTWR are selected, namely an easier version where 10 vehicles are available and a harder version where only 5 vehicles are available. The problem sizes range from 50 customers to 500 customers to indicate both small and large-scale problems in reality. Next, we conducted several ablation studies. We first show that the GIN based manager agent can well learn a customer assignment strategy for each vehicle to boost the performance of worker agent, which is superior to a multi layer perceptron (MLP) based one and a counterpart heuristic strategy, e.g., K-means. Afterwards we justify that multi-head architecture is a key component for the manager agent to learn meaningful vehicle embeddings. Finally, we show our algorithm is robust against different values of the penalty coefficient β, although (we show) different values of β will affect the solutions in terms of tour length and rejection rate. The configuration details of the baselines including K-means are given in appendix B.

A. Settings

Following the settings that are commonly used in the DRL based methods for solving routing problems [12], [15],

[16], [29], we randomly generate problem instances with different sizes of customers (n) and vehicles (m). Specifically, we consider $n = \{50, 100, 150\}$ as small sizes to train and test our method, and $n = \{200, 300, 400, 500\}$ as relatively large sizes to evaluate the generalization performance. For each customer size, we also consider m=5 and 10, respectively, where m=5 is considered as *harder* since fewer vehicles are available. We follow [16] to generate customer locations, *i.e.,* $\forall c_i, (x_i, y_i) \in U[0, 1] \times U[0, 1]$ with $U[\cdot]$ denoting the uniform distribution. The time window is unified for all sizes and provided as $\forall c_i, s_i \sim U[0, 3]$, $t_i = s_i + 3$. The location of depot can be randomly selected or predefined in the unit square, however, for an easier data generation, We fix the depot at location $[0.5, 0.5]$ with time window $[0, 10]$, where 10 is used to make sure all vehicles return to depot. Note that, the customer size n also includes the depot, which is considered as a *dummy customer* in our setting.

Algorithm 5 Decoder

Input: Overall embedding $\bar{h}^{(N)}$, Key for nodes K.
Output: Tour for all customers r.
Initialize two trainable place holders for step 0: v_f, v_l.
Compute initial decode context: $q_d = W^{Q_d}\text{Concate}(\text{Encoder}(\bar{h}^{(N)}, v_f, v_l))$. Concate() denotes concatenation operator
for i = 1:n **do**
  Compute the compatibilities: $u_{dj} = \frac{q_d^T k_j}{\sqrt{d_k}}$, $j = (1, \ldots, n)$.
  Mask visited nodes: $u_{dj} = -\infty$, if node j is visited.
  Compute probability of visiting node j: $p_{dj} = \frac{e^{u_{dj}}}{\sum_i e^{u_{dj}}}$.
  Sample a node based on the probability: $r_i = \text{Sample/GreedySample}(p_{dj})$. # GreedySample() is only used in inference.
  Update v_f,v_l: $v_f = h_{r_1}^{(N)}$,$v_l = h_{r_i}^{(N)}$.
  Update decode context: $q_d = W^{Q_d}\text{Concate}(\text{Encoder}(\bar{h}^{(N)}, v_f, v_l))$.
end for

Algorithm 6 Backtracking Function

Input: Tour r (for all customers), Instances S.
Output: Solution to TSPTWR r_b.
Initialize time $t = 0$,
for i = 0:$n - 1$ **do**
  $t = t + t_{r_i, r_{i+1}}$, where $t_{r_i, r_{i+1}}$ is the time needed from node r_i to r_{i+1}.
  if $t > t_{r_{i+1}}$(the termination time of node r_{i+1}) **then**
    $t = t - t_{i,i+1}$.
    delete r_{i+1} from r.
  end if
end for

We tune all hyperparameters on size n=50 with m=10 and 5, respectively, and fix them during training and evaluations for other sizes. Pertaining to the manager agent, we train the policy network for 10,000 iterations with 128 (batch size) instances generated on-the-fly at each iteration. The model is validated on 100 fixed instances every 100 training iterations. For GIN, we adopt 3 layers and set $\epsilon^{(t)} = 0$ for all layers following [17]. The $mlp_{\theta_t}^{(t)}$ for each layer has 2 hidden layers with dimension 32. We use mean neighborhood aggregation for each GIN layer and mean pooling as the readout function over the entire graph. For multi-head self-attention vehicle embedding network and single-head self-attention assignment network, we unify the hidden dimensions of all parameters, *i.e.,* $\theta_{qb}, \theta'_q, \theta'_k \in \mathbb{R}^{64\times64}$ and $\theta_{kb}, \theta_{vb}, \in \mathbb{R}^{32\times64}$. Pertaining to the worker agent, we mainly follow configurations in [22]. The pretraining sizes related to solving TSPTWR are set as $\lceil n/m \rceil$ (*i.e.,* $\lceil\cdot\rceil$ refers to the rounding up operation). For example, regarding the mTSPTWR instances with $m = 5$ and $n = 150$, the pretraining size is $\lceil 150/5 \rceil = 30$, and it also can solve TSPTWR with customers more or less than 30. Accordingly, we adopt pretraining sizes of $\{5, 10, 15, 20, 30\}$ for the worker agent to train the manager agent on small sizes of mTSPTWR, while $\{40, 50, 60, 80, 100\}$ to evaluate the generalization on large sizes. The penalty coefficient β for rejection rate is set to 100. We use Adam optimizer with constant learning rate $lr = 1 \times 10^{-4}$ for both the manager and worker agents during training. Other parameters follow the default settings in PyTorch [30].

As reviewed in Section II, due to the complexity, the mTSPTWR has been rarely studied with DRL based methods. To better benchmark our method, we adapt AM [16] to mTSPTWR as a baseline, which is known as the state-of-the-art DRL method for TSP. Besides, we adopt three competitive and representative conventional metaheuristic algorithms including tabu search (TS) [31], simulated annealing (SA) [32] and bees algorithm (BA) [33] as additional baselines. Moreover, we conduct detailed ablation studies on the components in our proposed framework to verify their effectiveness. The hardware we use is a PC with Intel Core i9-10940X CPU and 251GB memory. The specific configurations of baselines are carefully tuned for the same objective, *i.e.,* minimizing the hybrid cost of length and rejection rate for the worst sub-tour, and given in the appendix B.

B. Results and Analysis

1) Comparison Studies on Easier and Harder mTSPTWR: We first train and test AM and our method on small sizes of mTSPTWR with n = 50, 100,[1] and 150. Then we evaluate the generalization performance on large sizes of mTSPTWR without re-training, where n = 200, 300, 400, and 500. For each customer size, we consider two different sizes of vehicles, namely, *easier* mTSPTWR with 10 vehicles (m=10) and *harder* mTSPTWR with 5 vehicles (m=5). We infer 100 unseen randomly generated instances for each size of customer and vehicle, and explicitly report the averaged tour length, rejection rate, hybrid cost and computation time. Note that during inference, both manager and worker agent greedily choose respective actions according to computed probability,

[1]The biggest size for AM is $n = 100$ due to the heavy computation [16], and bigger sizes than $n = 100$ cause out of memory.

TABLE I

OUR METHOD VS. BASELINES ON MTSPTWR WITH M=10. "REJ. RATE": REJECTION RATE. COMPUTATION TIME IS COUNTED IN SECONDS. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**

Size		Ours-50	Ours-100	Ours-150	AM-50	AM-100	TS	SA	BA
50	Length	3.56	3.57	3.69	3.60	3.91	3.70	**3.42**	**3.42**
	Rej. Rate	**0.00%**	**0.00%**	0.46%	**0.00%**	0.50%	3.89%	**0.00%**	**0.00%**
	Hybrid Cost	3.56	3.57	4.15	3.60	4.41	7.59	**3.42**	**3.42**
	Time(s)	**0.08**	**0.08**	**0.08**	0.09	0.10	⩾1800	⩾1800	⩾1800
100	Length	3.88	3.82	3.91	6.41	4.03	3.71	**3.47**	3.95
	Rej. Rate	0.13%	**0.00%**	**0.00%**	64.30%	0.12%	**0.00%**	**0.00%**	**0.00%**
	Hybrid Cost	4.01	3.82	3.91	70.71	4.15	3.71	**3.47**	3.95
	Time(s)	0.14	**0.13**	0.14	0.14	0.15	⩾1800	⩾1800	⩾1800
150	Length	4.56	4.33	4.22	6.43	5.78	4.17	**3.63**	5.01
	Rej. Rate	0.89%	0.03%	**0.00%**	79.23%	53.80%	0.40%	0.45%	5.10%
	Hybrid Cost	5.64	4.36	4.22	85.66	59.58	4.57	**4.09**	10.11
	Time(s)	0.20	**0.19**	**0.19**	0.21	0.25	⩾1800	⩾1800	⩾1800
200	Length	4.93	**4.61**	4.82	6.48	6.00	4.86	4.59	5.40
	Rej. Rate	4.62%	2.99%	**0.63%**	85.00%	74.67%	2.60%	2.97%	14.53%
	Hybrid Cost	9.56	7.60	**5.44**	91.48	80.67	7.46	7.56	19.93
	Time(s)	0.26	0.26	**0.25**	0.27	0.28	⩾1800	⩾1800	⩾1800
300	Length	5.21	**5.17**	5.41	6.53	6.04	5.69	5.57	5.75
	Rej. Rate	8.10%	**0.99%**	0.84%	90.43%	86.96%	25.69%	7.14%	33.18%
	Hybrid Cost	13.30	**6.17**	6.25	96.96	93.00	31.38	12.71	38.93
	Time(s)	0.42	**0.41**	**0.41**	0.51	0.52	⩾1800	⩾1800	⩾1800
400	Length	5.27	**5.16**	5.22	6.54	6.06	5.95	5.99	5.89
	Rej. Rate	18.46%	7.22%	**6.18%**	92.72%	91.02%	43.24%	14.60%	47.62%
	Hybrid Cost	23.73	12.38	**11.41**	99.26	97.08	49.19	20.59	53.51
	Time(s)	0.63	**0.61**	**0.61**	0.64	0.64	⩾1800	⩾1800	⩾1800
500	Length	**5.44**	5.50	5.47	6.55	6.07	6.03	6.01	5.98
	Rej. Rate	25.86%	12.40%	**9.61%**	93.63%	93.44%	56.97%	22.71%	58.13%
	Hybrid Cost	31.30	17.89	**15.08**	100.18	99.51	63.00	28.72	64.11
	Time(s)	**0.80**	0.81	0.83	0.84	0.88	⩾1800	⩾1800	⩾1800

TABLE II

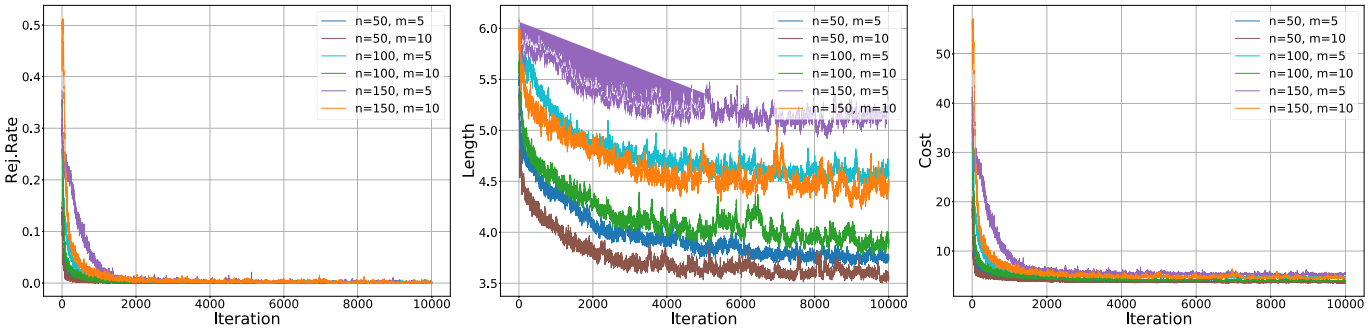
OUR METHOD VS. BASELINES ON MTSPTWR WITH M=5. "REJ. RATE": REJECTION RATE. COMPUTATION TIME IS COUNTED IN SECONDS. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**

Size		Ours-50	Ours-100	Ours-150	AM-50	AM-100	TS	SA	BA
50	Length	3.75	3.85	3.77	3.74	4.00	3.60	**3.44**	3.53
	Rej. Rate	**0.00%**	0.04%	0.04%	0.06%	0.20%	**0.00%**	**0.00%**	**0.00%**
	Hybrid Cost	3.75	3.89	3.80	3.80	4.20	3.80	**3.44**	3.53
	Time(s)	**0.07**	0.08	**0.07**	0.11	0.10	⩾1800	⩾1800	⩾1800
100	Length	4.77	4.53	4.65	6.00	4.94	4.60	**4.13**	5.22
	Rej. Rate	0.83%	0.12%	**0.00%**	73.39%	0.86%	1.89%	1.42%	4.94%
	Hybrid Cost	5.59	**4.65**	**4.65**	79.39	5.81	6.49	5.55	10.16
	Time(s)	0.14	**0.13**	**0.13**	0.14	0.15	⩾1800	⩾1800	⩾1800
150	Length	5.58	5.33	**4.99**	6.20	6.26	5.48	5.20	5.59
	Rej. Rate	2.17%	0.34%	**0.00%**	86.15%	74.59%	13.24%	5.20%	18.91%
	Hybrid Cost	7.74	5.67	**4.99**	92.35	80.85	18.72	10.40	24.50
	Time(s)	**0.19**	**0.19**	0.20	0.21	0.22	⩾1800	⩾1800	⩾1800
200	Length	5.87	**5.36**	5.37	6.27	6.34	5.66	5.69	5.71
	Rej. Rate	8.69%	2.32%	**0.17%**	90.61%	85.02%	19.49%	8.98%	31.98%
	Hybrid Cost	14.55	7.68	**5.55**	96.88	91.36	25.15	14.67	37.69
	Time(s)	0.27	**0.26**	0.27	0.29	0.28	⩾1800	⩾1800	⩾1800
300	Length	5.99	5.73	**5.64**	6.35	6.40	5.97	6.00	5.97
	Rej. Rate	17.67%	14.03%	**8.34%**	94.68%	91.43%	48.97%	18.60%	51.36%
	Hybrid Cost	23.66	19.75	**13.98**	101.03	97.83	54.94	24.60	57.32
	Time(s)	0.45	0.45	**0.44**	0.45	0.47	⩾1800	⩾1800	⩾1800
400	Length	6.16	**5.59**	5.97	6.40	6.45	6.03	6.10	6.08
	Rej. Rate	26.99%	20.13%	**15.14%**	96.33%	93.87%	62.79%	27.54%	63.32%
	Hybrid Cost	33.15	25.72	**21.11**	102.73	100.32	68.82	33.64	69.40
	Time(s)	0.66	**0.63**	0.66	0.72	0.71	⩾1800	⩾1800	⩾1800
500	Length	6.20	**5.79**	6.19	6.43	6.48	6.09	6.10	6.11
	Rej. Rate	36.70%	27.61%	**22.96%**	97.18%	95.40%	70.63%	33.79%	70.85%
	Hybrid Cost	42.90	33.40	**29.15**	103.61	101.88	76.72	39.89	76.97
	Time(s)	1.11	**1.10**	**1.10**	1.26	1.25	⩾1800	⩾1800	⩾1800

i.e., the action with the highest probability will be selected. Although sampling multiple solutions may yield better results, greedy policy is faster for inference while still producing satisfactory results (see [17]).

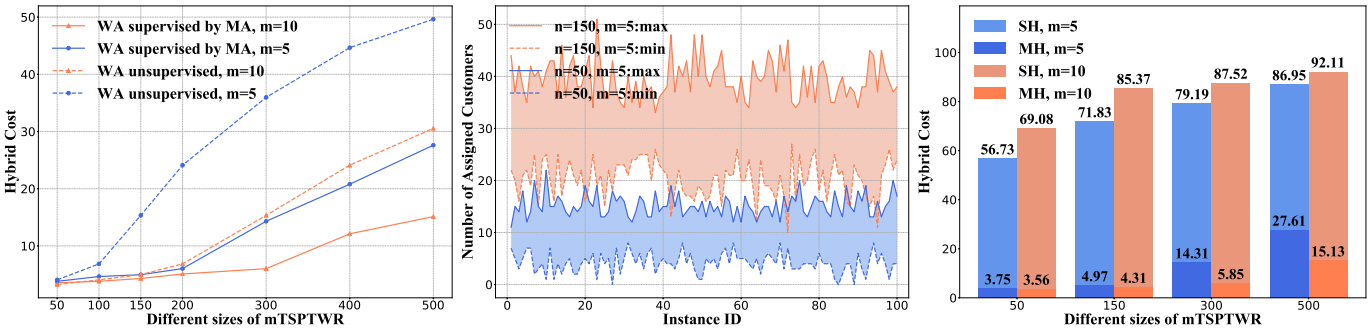
Table I demonstrates that for easier mTSPTWR, when the problem sizes are small, the conventional metaheuristics can produce solutions with high qualities, e.g., SA achieves the lowest hybrid cost with customer sizes of 50, 100, and 150. AM exhibits acceptable performance for testing sizes smaller than the training ones, but fails to generalize to bigger sizes. Our proposed method achieves competitive performance compared to SA with slightly higher hybrid cost, and generalizes well to different sizes. It should be emphasized that, thanks to the careful tuning of hyperparameters and small solution space, conventional metaheuristics could achieve desirable performance where SA attains slightly better objective values than ours by solving instances individually. However, its computation time is far longer as SA needs to search from scratch for each instance, especially given that ours only requires about 0.08s to solve an instance. This significantly limits the scalability of conventional metaheuristics. On larger sizes, the performance of all the baselines drops dramatically, because it is hard for the conventional metaheuristics to conduct an efficient search in such large solution space, while AM cannot be generalized. Our method outperforms all the baselines in terms of the solution quality and computation time, even though our models are trained on instances that are much smaller than the inferred ones. The overall trend is that the closer testing sizes are to the training ones, the better the performance is. Moreover, it should be noticed that in all these instances our method outputs lower rejection rates than the baselines. It means that our learnt policy is able to reduce unnecessary rejections with minor sacrifices in the hybrid cost. Generally, the results reveal that our model can be generalized well to unseen instances of the same or larger sizes, while maintaining competitive performance. Concerning real-life problems whose sizes may vary, our method has the desirable potential to be trained for a specific size but applied to handle various sizes with short inference time.

Table II shows that other than the fast inference, our method is also robust for solving harder mTSPTWR. In comparison with the results in Table I, both the rejection rate and tour length increase for all methods in Table II, which is natural since there are fewer available vehicles for the customers to choose from. However, the increments in hybrid cost for the conventional metaheuristics are considerably large, especially as the number of customers becomes higher than 100. In contrast, the increments for our method is relatively low across all sizes. Again, in our method, the closer testing sizes are to the training ones, the better the performance is. The superiority of our method comes from the fact that it is always able to yield lower rejection rates. Although increasing the number of available vehicles could decrease the rejection rates, how to efficiently serve more customers in the case of a limited number of vehicles would be more desirable. Remarkably, our method seems having a strong capability to capture this key feature of mTSPTWR and learn competitive policies accordingly. In addition, the training curves for all sizes of problems given in Fig. 4 demonstrate that our algorithm performs stably across all sizes of problems, and converges quickly as well.



(a) Training curve for rejection rate (b) Training curve for length (c) Training curve for hybrid cost

Fig. 4. Training curves of rejection rate, length, and hybrid cost for the worst sub-tour of each problem. The hybrid cost curve has the similar shape as that of rejection rate, and this is because we set $\beta = 100$ in our experiment, rendering rejection rate the relatively dominant term in the formula $J_b \triangleq l(r_b) + 100 \cdot rej(r_b)$.



(a) (b) (c)

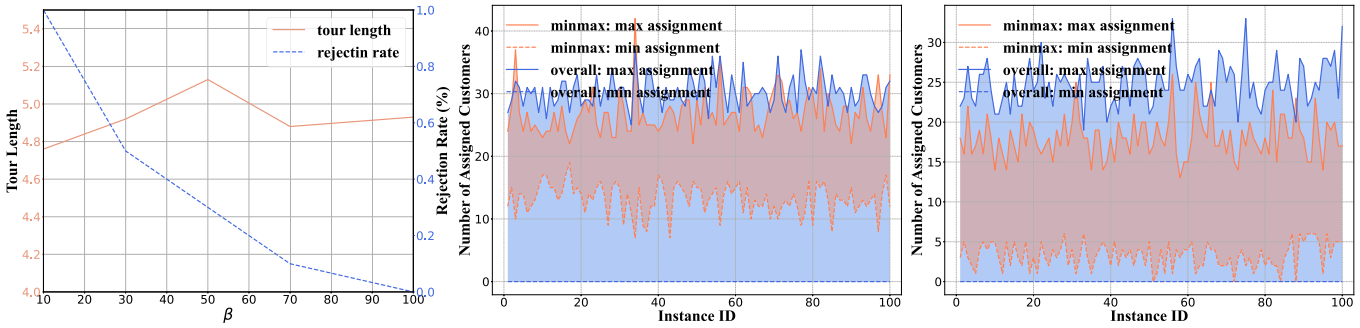
Fig. 5. (a) Comparison between worker agent alone and with manager agent; (b) The maximum and minimum number of customers assigned to vehicles in an instance; (c) Comparison between single-head (SH) and multi-head (MH) self-attention for vehicle embedding.

2) Compensation Study: We further verify the superiority of our method by analyzing the compensation from the manager agent to the worker agent. In fact, the worker agent is pre-trained with the customer size $\lceil n/m \rceil$, and intuitively, it should achieve the best performance when handling TSPTWR with about $\lceil n/m \rceil$ customers. However, we show that with the assignment by the manager agent, the worker agent can further reduce hybrid cost even if the pretrained model is applied to solve TSPTWR of other sizes during inferring an mTSPTWR instance.

As depicted in Fig. 5(a), we show the average hybrid cost of TSPTWR by the sole worker agent (the dashed lines) and the corresponding mTSPTWR by the manager-agent framework (the solid lines), respectively, where the former model is trained and tested with TSPTWR of $\lceil n/m \rceil$ customers. Note that, the hybrid cost for mTSPTWR still refers to the maximum one among the m vehicles. We observe that with the supervision by the manager agent, the hybrid cost of mTSPTWR is much lower than that of the corresponding TSPTWR. This superiority becomes more significant as the customer size increases, which well reflects that the manager agent could compensate the sole work agent by learning more effective customer assignment for it. As depicted in Fig. 5(b), we present the maximum and minimum numbers of assigned customers to the vehicles regarding the involved mTSPTWR instances. We can obviously see that the manager agent always assign various sizes of customers to vehicles rather than the ones close to $\lceil n/m \rceil$.

3) Ablation Study for Assignment Strategy: We compare the assignment strategy learnt by our manager agent against K-means heuristic (a clustering method commonly applied in the two-stage routing problem [34]) and the one learnt by a 3-layer MLP. In particular, regarding the former baseline, the customers are partitioned into m clusters by K-means based on spatial information (locations) and temporal information (time windows), after which the same worker agent is applied for each cluster. Regarding the latter baseline, we mainly replace the GIN with MLP while keeping the remaining parts of the whole framework unchanged. Both ours and MLP are trained for mTSPTWR-150 with $m = 5/10$ and generalized to the other sizes. Table III summarizes the performance of our manager agent, MLP and K-means. It can be observed that, although K-means produces shorter route lengths on some instances, its rejection rate and the hybrid cost are much inferior to our manager agent and MLP in most cases. It implies that K-means fails to generate desirable clusters (that would lead to lower hybrid cost) by mining the given spatial and temporal information. Moreover, compared with MLP, our GIN based manager agent can successfully learn a more effective assignment policy from these heterogeneous raw features yielding better performance for most of the cases.

4) Ablation Study for Multi-Head Architecture: To justify the effectiveness of the multi-head attention for the manager agent to identify and differentiate m identical vehicles, we compare it with a single-head self-attention where all m vehicles share parameters. Specifically, we compute their



(a) The ablation study for β. (b) $n = 100$, $m = 5$. (c) $n = 100$, $m = 10$.

Fig. 6. (a) The ablation study for different values of β with size ($n = 150, m = 5$); (b) The maximum and minimum number of customers assigned to vehicles in ($n = 100, m = 5$) instances for objectives "min max" and "overall", respectively; (c) The maximum and minimum number of customers assigned to vehicles in ($n = 100, m = 10$) instances for objectives "min max" and "overall", respectively.

TABLE III

THE PERFORMANCE OF OUR MODEL COMPARED WITH K-MEANS HEURISTIC WHERE THE NUMBER OF VEHICLES m SERVES AS THE NUMBER OF CLUSTERS AND THE ASSIGNMENT STRATEGY LEARNT BY AN MLP. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**

Size		m=5			m=10		
		Ours-150	MLP-150	Kmeans	Ours-150	MLP-150	Kmeans
50	Length	3.77	**3.73**	4.95	**3.69**	3.71	4.21
	Rej. Rate	0.04%	**0.00%**	3.56%	0.46%	**0.00%**	**0.00%**
	Hybrid Cost	3.80	**3.73**	8.41	4.15	**3.71**	4.21
	Time(s)	0.07	**0.08**	0.10	**0.08**	0.10	0.15
100	Length	**4.65**	4.76	4.75	**3.91**	4.16	4.80
	Rej. Rate	**0.00%**	0.22%	20.83%	**0.00%**	0.18%	1.77%
	Hybrid Cost	**4.65**	4.98	25.58	**3.91**	4.34	6.57
	Time(s)	**0.13**	0.15	0.17	**0.14**	0.16	0.21
150	Length	4.99	**4.90**	4.93	**4.30**	4.85	4.79
	Rej. Rate	**0.00%**	**0.00%**	33.36%	**0.04%**	0.68%	10.58%
	Hybrid Cost	4.99	**4.90**	38.29	**4.34**	5.53	15.37
	Time(s)	**0.20**	0.22	0.24	**0.19**	0.22	0.29
200	Length	5.37	5.56	**5.18**	4.82	5.32	**4.79**
	Rej. Rate	**0.17%**	7.00%	39.42%	**0.63%**	1.12%	19.49%
	Hybrid Cost	**5.55**	12.56	44.61	**5.44**	6.44	24.28
	Time(s)	**0.27**	**0.27**	0.31	**0.25**	0.27	0.36
300	Length	5.64	5.83	**5.05**	5.41	5.86	**5.05**
	Rej. Rate	**8.34%**	13.06%	49.04%	**0.84%**	3.46%	33.18%
	Hybrid Cost	**13.98**	18.89	54.09	**6.25**	9.32	38.23
	Time(s)	0.44	**0.43**	0.51	0.41	**0.40**	0.55
400	Length	5.97	6.17	**3.95**	5.22	5.77	**5.03**
	Rej. Rate	**15.14%**	26.85%	73.58%	**6.18%**	12.49%	38.15%
	Hybrid Cost	**21.11**	33.02	77.53	**11.41**	18.25	43.19
	Time(s)	0.66	**0.56**	0.73	**0.61**	**0.61**	0.77
500	Length	6.19	6.26	**3.97**	5.47	6.16	**4.75**
	Rej. Rate	**22.96%**	36.66%	74.40%	**9.61%**	21.56%	43.80%
	Hybrid Cost	**29.15**	42.92	78.37	**15.08**	27.72	48.55
	Time(s)	1.10	**0.91**	0.97	**0.83**	0.90	1.01

$query$, $keys$, and $values$ as $q_b = \theta_q h_c$, $k_{bi} = \theta_k h_i$, and $v_{bi} = \theta_v h_i$ for $\forall c_i \in C$, where θ_q, θ_k and θ_v are shared trainable parameters. All other configurations of the model remain the same as in subsection IV-A. In Fig. 5(c), we observe that compared with sharing parameters, embedding each vehicle separately via independent network can learn more effective representation for each vehicle. In contrast, with the single-head architecture, the resulting assignment policy for the manager agent seemingly degenerates into a random allocation.

5) Ablation Study for Different Penalty Coefficient (β): In reality, the respective magnitude of rejection rate and sub-tour length in hybrid cost may vary under different circumstances. For example, when some customers are VIPs, the vehicles may have to serve them regardless of the cost. In this situation, the manager agent should pay more attention to optimize rejection rate while adjusting their service strategies. On the other hand, some customers may have flexible schedules which makes an alternative visit possible. In this case, vehicles may postpone services to them to make the sub-tour length small for saving cost. All these different purposes can be achieved via adjusting the penalty coefficient β in the objective formula $J_b \triangleq l(r_b) + \beta \cdot rej(r_b)$, *i.e.,* paying more attention on either optimizing rejection rate or sub-tour length. Previously we have evaluated our method with $\beta = 100$. Here we continue

TABLE IV

THE PERFORMANCE OF OUR MODEL FOR $\beta = 10$ AND $m = 5$. THE NUMBER IN THE BRACKETS ARE DIFFERENCES TO THAT OF $\beta = 100$. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**

size		Ours-50	Ours-100	Ours-150	SA
50	length	3.66 (-0.09)	3.71 (-0.14)	3.87 (0.10)	**3.47** (0.03)
	Rej. Rate	0.00% (0.00)	0.00% (-0.04)	0.08% (0.04)	**0.00** % (0.00)
	Hybrid Cost	3.66	3.71	3.87	**3.47**
	Time(s)	**0.08**	**0.08**	**0.08**	⩾1800
150	length	4.79 (-0.78)	4.84 (-0.49)	4.76 (-0.23)	**4.13** (-1.07)
	Rej. Rate	1.46% (-0.71)	1.21% (0.80)	**1.02**% (1.02)	7.78% (2.58)
	Hybrid Cost	4.94	4.96	**4.86**	4.91
	Time(s)	**0.21**	**0.21**	0.24	⩾1800
300	length	5.46 (-0.52)	5.56 (-0.17)	5.54 (-0.10)	**4.95** (-1.05)
	Rej. Rate	19.14% (1.47)	14.84% (0.81)	**10.82**% (2.48)	21.10% (2.60)
	Hybrid Cost	7.38	7.04	**6.62**	7.06
	Time(s)	0.20	**0.19**	0.24	⩾1800
500	length	6.11 (-0.09)	6.12 (0.34)	6.04 (-0.15)	**5.17** (-0.93)
	Rej. Rate	39.90% (3.19)	37.47% (9.86)	**33.44%** (10.48)	35.61% (1.82)
	Hybrid Cost	10.09	9.87	9.38	**8.74**
	Time(s)	0.92	**0.89**	0.92	⩾1800

TABLE V

THE PERFORMANCE OF OUR MODEL FOR $\beta = 10$ AND $m = 10$. THE NUMBER IN THE BRACKETS ARE DIFFERENCES TO THAT OF $\beta = 100$. THE BEST RESULTS ARE HIGHLIGHTED IN **BOLD**

size		Ours-50	Ours-100	Ours-150	SA
50	length	3.58 (0.02)	3.67 (0.11)	3.85 (0.16)	**3.38** (-0.04)
	Rej. Rate	0.00% (0.00)	0.04% (0.04)	0.26% (-0.20)	**0.00**% (0.00)
	Hybrid Cost	3.58	3.68	3.87	**3.38**
	Time(s)	**0.09**	**0.09**	**0.09**	⩾1800
150	length	**4.15** (-0.42)	4.21 (-0.12)	4.23 (-0.08)	5.06 (1.43)
	Rej. Rate	**0.21**% (-0.69)	0.60% (0.57)	0.30% (0.30)	1.58% (1.13)
	Hybrid Cost	**4.17**	4.21	4.26	5.22
	Time(s)	**0.21**	0.22	0.22	⩾1800
300	length	4.73 (-0.47)	5.09 (-0.08)	**4.04** (-0.47)	11.35 (5.78)
	Rej. Rate	2.47% (-5.36)	2.52% (1.52)	**1.21**% (0.37)	6.86% (-0.28)
	Hybrid Cost	**5.01**	5.34	5.06	12.04
	Time(s)	0.48	0.48	**0.47**	⩾1800
500	length	5.20 (-0.24)	5.54 (0.04)	5.44 (-0.02)	**5.01** (-1)
	Rej. Rate	16.00% (-9.86)	**12.48%** (0.08)	13.05% (3.44)	26.46% (3.75)
	Hybrid Cost	6.80	6.78	**6.75**	7.65
	Time(s)	0.88	0.89	**0.87**	⩾1800

to show our algorithm learns adjusted policies for another case when $\beta = 10$. Pertaining to the baseline, we only compare with the simulated annealing (SA) since it demonstrated relatively superior performance to others previously. For simplification, we only consider small size (50), median size (150), and large sizes (300, 500). The results for easier and harder mTSPTWR are recorded in Table V and Table IV, respectively. We observe that our algorithm still preserves better performance against baseline in the case of $\beta = 10$. Interestingly, when compared with results for $\beta = 100$, our algorithm seems able to adjust learnt policies to capture the discrepancies. For instance, regarding the same problem instances, the learnt policies should yield a smaller sub-tour length but a larger rejection rate, since $\beta = 10$ is much smaller compared with $\beta = 100$. This expectation is basically reflected by the difference values in the brackets. Furthermore, we illustrate how different β values affect the tour length and rejections for mTSPTWR with $m = 5, n = 150$ in Fig. 6(a), which also reveals the general trend.

V. Conclusion and Future Work

In this paper, we present a DRL based manager-worker framework to tackle a challenging yet practical variant of TSP with (1) multiple vehicles, (2) time window and rejections, *i.e.,* mTSPTWR. By dividing mTSPTWR into manager-level and worker-level tasks, the respective agents can learn strong policies to produce high-quality solutions to the original problem in short computation time. Extensive experiments confirm the superiority of our method against the metaheuristic baselines and also show that the proposed framework can generalize well to larger instances unseen during training. In the future, besides extending our framework to other multi-vehicle routing problems and real world datasets, we also would like to investigate, (1) how to enable the two agents to feed back to each other and train them jointly; (2) how to optimize the number of vehicles without exhausting them all; (3) how to automatically decide the penalty coefficient β for different scenarios.

TABLE VI

The Comparison of Our Model With SA w.r.t the Overall Cost. The "Length" and the "Rej. Rate" Are the Average Distance and Rejection Rate for All Vehicles, Respectively. The "Hybrid Cost" Is Calculated as "Length" + β · "Rej. Rate" Where $\beta = 100$. The Best Results Are Highlighted in **Bold**

Size		m=5		m=10	
		Ours	SA	Ours	SA
100	Length	**3.48**	3.57	2.08	**2.04**
	Rej. Rate	0.07%	**0.03%**	**0.01%**	**0.01%**
	Hybrid Cost	**3.55**	3.60	2.09	**2.05**
	Time(s)	**0.16**	$\geqslant$ 1800	**0.15**	$\geqslant$ 1800

TABLE VII

Hyperparameters for Baselines

Baseline	Hyperparameters	Value
TS	number of total possible actions	n^2
	tabu length	$\frac{n^2}{2}$
	termination threshold	10^{-6}
	initial solution	greedy
SA	population size	100
	sub-iterations	10
	number of moves	15
	mutation rate	$\frac{1}{n}$
	mutation step size	0.49
	mutation step size damp rate	1
	initial temperature	10000
	final temperature	1
BA	population size	500
	selected sites ratio	0.9
	selected sites bee count ratio	0.1
	elite sites ratio	0.2
	elite sites bee count ratio	2
	bees dance radius	1
	bees dance radius damp rate	0.99

Appendix A
Discussion on Different Objectives for mTSPTWR

Besides the objective we considered in Eq. (1), a more straightforward objective for mTSPTWR is directly minimizing the overall statistics, *i.e.,* the average tour length and overall rejection rate. We retrain our proposed framework for this overall objective and use SA with the same objective as a baseline. Using mTSPTWR with $m = 5/10$, $n = 100$ as an example, we demonstrate that our framework can also work well for the overall objective, the results of which are summarized in Table VI. However, since the sub-tours are not (explicitly) assessed in the overall objective, it may lead to extremely "unbalanced" solutions, where some vehicles are not deployed and some vehicles are assigned with too many customers. This may not make the best use of a fleet and consequently harm the level of service. In contrast, our objective encourages the manager agent to assign customers in a more balanced way, which can fulfill practical demands from the customer's perspective. Particularly, Fig. 6(b) and Fig. 6(c) display the minimum and maximum number of customers assigned to vehicles, which also justified the favorable property of our objective.

Appendix B
Hyperparameters for Baselines and Configuration of K-Means

We present detailed configurations for our baselines, namely tabu search (TS), simulated annealing (SA) and bees algorithm (BA). For each method, the objective in Eq. (1) with $\beta = 100$ is adopted as the evaluation function. TS is implemented with python. SA and BA are implemented in Matlab using YPEA [35] toolbox. All the solutions can be manipulated by swapping, reversing, and inserting to generate neighbors. The hyperparameters are carefully tuned, and we select the ones that can deliver satisfactory overall performance on all sizes of the problems. As a common hyperparameter, the

number of iterations for each method is set to 1000. Other hyperparameters are listed in Table VII for each respective baseline, where n denotes the number of customers in the corresponding problem instances. For K-means, we use `scikit-learn` [36] machine learning package with the following hyper-parameters, *i.e.,* `max_iter`=1000 (Maximum iteration number of the K-means algorithm for a single run). All other parameters follow the default setting.

REFERENCES

[1] Z. Li, Q. Chen, and V. Koltun, "Combinatorial optimization with graph convolutional networks and guided tree search," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 539–548.

[2] E. Yolcu and B. Póczos, "Learning local search heuristics for Boolean satisfiability," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 7992–8003.

[3] J. Li, L. Xin, Z. Cao, A. Lim, W. Song, and J. Zhang, "Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2306–2315, Mar. 2021.

[4] Z.-H. Fu, K.-B. Qiu, and H. Zha, "Generalize a small pre-trained model to arbitrarily large TSP instances," 2020, *arXiv:2012.10658*.

[5] L. Xin, W. Song, Z. Cao, and J. Zhang, "Step-wise deep learning models for solving routing problems," *IEEE Trans. Ind. Informat.*, vol. 17, no. 7, pp. 4861–4871, Oct. 2020.

[6] L. Xin, W. Song, Z. Cao, and J. Zhang, "Neurolkh: Combining deep learning model with Lin-Kernighan–Helsgaun heuristic for solving the traveling salesman problem," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 1–12.

[7] M. Chen, L. Gao, Q. Chen, and Z. Liu, "Dynamic partial removal: A neural network heuristic for large neighborhood search," 2020, *arXiv:2005.09330*.

[8] J. Zhao, M. Mao, X. Zhao, and J. Zou, "A hybrid of deep reinforcement learning and local search for the vehicle routing problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 7208–7218, Nov. 2020.

[9] Y. Kaempfer and L. Wolf, "Learning the multiple traveling salesmen problem with permutation invariant pooling networks," 2018, *arXiv:1803.09621*.

[10] Y. Hu, Y. Yao, and W. S. Lee, "A reinforcement learning approach for optimizing multiple traveling salesman problems over graphs," *Knowl.-Based Syst.*, vol. 204, pp. 106–244, Sep. 2020.

[11] J. Li *et al.*, "Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem," *IEEE Trans. Cybern.*, early access, Sep. 23, 2021, doi: 10.1109/TCYB.2021.3111082.

[12] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 28, 2015, pp. 2692–2700.

[13] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takác, "Reinforcement learning for solving the vehicle routing problem," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9839–9849.

[14] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[15] M. Deudon, P. Cournut, A. Lacoste, Y. Adulyasak, and L.-M. Rousseau, "Learning heuristics for the TSP by policy gradient," in *Proc. Int. Conf. Integr. Constraint Program., Artif. Intell., Oper. Res.*, 2018, pp. 170–181.

[16] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems!" in *Proc. Int. Conf. Learn. Represent.*, 2019, pp. 1–25.

[17] Y. Wu, W. Song, Z. Cao, J. Zhang, and A. Lim, "Learning improvement heuristics for solving routing problems," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 9, pp. 5057–5069, 2021.

[18] Z.-H. Fu, K.-B. Qiu, and H. Zha, "Generalize a small pre-trained model to arbitrarily large TSP instances," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 8, 2021, pp. 7474–7482.

[19] L. Perron and V. Furnon. (2019). *Or-Tools*. Google. [Online]. Available: https://developers.google.com/optimization/

[20] Q. Ma, S. Ge, D. He, D. Thaker, and I. Drori, "Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning," 2019, *arXiv:1911.04936*.

[21] E. K. Baker, "An exact algorithm for the time-constrained traveling salesman problem," *Oper. Res.*, vol. 31, no. 5, pp. 938–945, 1983.

[22] R. Zhang, A. Prokhorchuk, and J. Dauwels, "Deep reinforcement learning for traveling salesman problem with time windows and rejections," in *Proc. Int. Joint Conf. Neural Netw.*, Jul. 2020, pp. 1–8.

[23] C. J. Malmborg, "A genetic algorithm for service level based vehicle scheduling," *Eur. J. Oper. Res.*, vol. 93, no. 1, pp. 121–134, 1996.

[24] Q. Cappart, T. Moisan, L.-M. Rousseau, I. Prémont-Schwarz, and A. A. Cire, "Combining reinforcement learning and constraint programming for combinatorial optimization," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 5, 2021, pp. 3677–3687.

[25] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 2018.

[26] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–17.

[27] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 448–456.

[28] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Mach. Learn.*, vol. 8, nos. 3–4, pp. 229–256, 1992, doi: 10.1007/BF00992696.

[29] W. Jinzhan, H. Yanmei, Z. Zhaomin, and Z. Liucun, "An novel shortest path algorithm based on spatial relations," in *Proc. 4th Int. Conf. Electron. Inf. Technol. Comput. Eng.*, 2020, pp. 1024–1028.

[30] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 8026–8037.

[31] S. Aminzadegan, M. Tamannaei, and M. Fazeli, "An integrated production and transportation scheduling problem with order acceptance and resource allocation decisions," *Appl. Soft Comput.*, vol. 112, Nov. 2021, Art. no. 107770.

[32] A.-H. Zhou *et al.*, "Traveling-salesman-problem algorithm based on simulated annealing and gene-expression programming," *Inf.*, vol. 10, no. 1, p. 7, 2019.

[33] W. A. Hussein, S. Sahran, and S. N. H. S. Abdullah, "The variants of the bees algorithm (ba): A survey," *Artif. Intell. Rev.*, vol. 47, no. 1, pp. 67–121, 2017.

[34] S. C. Ho, W. Szeto, Y.-H. Kuo, J. M. Y. Leung, M. Petering, and T. W. Tou, "A survey of dial—A-ride problems: Literature review and recent developments," *Transp. Res. B, Methodol.*, vol. 111, pp. 395–421, May 2018.

[35] M. K. Heris, "YPEA: Yarpiz evolutionary algorithms," 2019. [Online]. Available: https://yarpiz.com/477/ypea-yarpiz-evolutionary-algorithms

[36] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011.

Rongkai Zhang received the B.Eng. degree from the Beijing Institute of Technology, China, in 2017, and the M.Sc. degree from Nanyang Technological University, Singapore, in 2018, where he is currently pursuing the Ph.D. degree. His research interests include deep reinforcement learning and its applications.

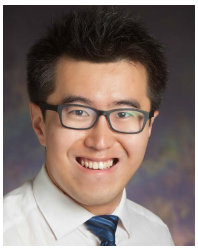
Cong Zhang received the B.Sc. degree from the Department of Mathematical Sciences, University of Liverpool, U.K., and the Department of Applied Mathematics, Xi'an Jiaotong-Liverpool University, China, in 2015, respectively, and the M.Sc. degree from the Department of Computing, Imperial College London, U.K., in 2016. He is currently pursuing the Ph.D. degree with Nanyang Technological University, Singapore, awarded the Singapore International Graduate Award (SINGA). His research interests include deep reinforcement learning, graph neural networks, job-shop scheduling, and intelligent vehicle routing.

Zhiguang Cao received the B.Eng. degree in automation from the Guangdong University of Technology, Guangzhou, China, the M.Sc. degree in signal processing from Nanyang Technological University (NTU), Singapore, and the Ph.D. degree from the Interdisciplinary Graduate School, Nanyang Technological University. He was a Research Assistant Professor with the Department of Industrial Systems Engineering and Management, National University of Singapore, and a Research Fellow with the Future Mobility Research Laboratory, NTU. He is currently a Scientist with the Singapore Institute of Manufacturing Technology (SIMTech), Agency for Science Technology and Research (A*STAR), Singapore. His research interests include neural combinatorial optimization.

Wen Song received the B.S. degree in automation and the M.S. degree in control science and engineering from Shandong University, Jinan, China, in 2011 and 2014, respectively, and the Ph.D. degree in computer science from Nanyang Technological University, Singapore, in 2018. He was a Research Fellow with the Singtel Cognitive and Artificial Intelligence Laboratory for Enterprises (SCALE@NTU). He is currently an Associate Research Fellow with the Institute of Marine Science and Technology, Shandong University. His current research interests include artificial intelligence, planning and scheduling, multi-agent systems, and operations research.

Puay Siew Tan received the Ph.D. degree in computer science from the School of Computer Engineering, Nanyang Technological University, Singapore. She is currently an Adjunct Associate Professor with the School of Computer Science and Engineering, Nanyang Technological University, and also the Co-Director of the SIMTECH-NTU Joint Laboratory on Complex Systems. In her full-time job at the Singapore Institute of Manufacturing Technology (SIMTech), she leads the Manufacturing Control TowerTM (MCTTM) as a Program Manager. She is also the Deputy Division Director of the Manufacturing System Division. Her research interests include cross-field disciplines of computer science and operations research for virtual enterprise collaboration, in particular sustainable complex manufacturing and supply chain operations in the era of industry 4.0.

Jie Zhang received the Ph.D. degree from the Cheriton School of Computer Science, University of Waterloo, Canada, in 2009. He is currently a Full Professor with the School of Computer Science and Engineering, Nanyang Technological University, Singapore. He is also an Adjunct Fellow of the Singapore Institute of Manufacturing Technology. His papers have been published by top journals and conferences and received several best paper awards. He is also active in serving research communities. During his Ph.D. study, he held the prestigious NSERC Alexander Graham Bell Canada Graduate Scholarship rewarded for top Ph.D. students across Canada. He was a recipient of the Alumni Gold Medal at the 2009 Convocation Ceremony. The Gold Medal is awarded once a year to honor the top Ph.D. graduate from the University of Waterloo.



Bihan Wen (Member, IEEE) received the B.Eng. degree in electrical and electronic engineering from Nanyang Technological University, Singapore, in 2012, and the M.S. and Ph.D. degrees in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2015 and 2018, respectively. He is currently a Nanyang Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University. His research interests include machine learning, computational imaging, computer vision, image and video processing, and big data applications. He was a recipient of the 2016 Yee Fellowship and the 2012 Professional Engineers Board Gold Medal. He was also a recipient of the Best Paper Runner Up Award at the IEEE International Conference on Multimedia and Expo in 2020. He has been an Associate Editor of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY since 2022, and an Associate Editor of *Micromachines* (MDPI) since 2021. He has also served as a Guest Editor for *IEEE Signal Processing Magazine* in 2022.

Justin Dauwels received the Ph.D. degree in electrical engineering from the Swiss Polytechnical Institute of Technology (ETH), Zurich, in December 2005. Moreover, he was a Post-Doctoral Fellow at the RIKEN Brain Science Institute (2006–2007) and a Research Scientist at the Massachusetts Institute of Technology (2008–2010). He is currently an Associate Professor at the Circuits and Systems Group, Department of Microelectronics, TU Delft. His academic laboratory has spawned four startups across a range of industries, ranging from AI for healthcare to autonomous vehicles. His research interests include data analytics with applications to intelligent transportation systems, autonomous systems, and analysis of human behavior and physiology. He serves as an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING since 2018, an Associate Editor of the *Signal Processing* journal (Elsevier) since 2021, a member of the Editorial Advisory Board of the *International Journal of Neural Systems*, and an organizer of IEEE conferences and special sessions. His research team has won several best paper awards at international conferences and journals.