Received: 5 July 2022 | Revised: 9 November 2022 | Accepted: 15 December 2022

IET Collaborative Intelligent Manufacturing

DOI: 10.1049/cim2.12072

The Institution of Engineering and Technology WILEY

REVIEW

A review on learning to solve combinatorial optimisation problems in manufacturing

Cong Zhang[1] | Yaoxin Wu[1] | Yining Ma[2] | Wen Song[3] | Zhang Le[4] | Zhiguang Cao[5] | Jie Zhang[1]

[1]School of Computer Science and Engineering, Nanyang Technological University, Singapore, Singapore

[2]Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore, Singapore

[3]Institute of Marine Science and Technology, Shandong University, Qingdao, China

[4]School of Information and Communication Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China

[5]Singapore Institute of Manufacturing Technology (SIMTech), A*STAR, Singapore, Singapore

Correspondence

Wen Song, Institute of Marine Science and Technology, Shandong University, Qingdao, China.
Email: wensong@email.sdu.edu.cn

Funding information

National Natural Science Foundation of China, Grant/Award Number: 62102228; Natural Science Foundation of Shandong Province, Grant/Award Number: ZR2021QF063; A*STAR Cyber-Physical Production System, CPPS – Towards Contextual and Intelligent Response Research Program, Grant/Award Number: A19C1a0018; Model Factory@SIMTech; A*Star Career Development Fund, Grant/Award Number: C222812027

Abstract

An efficient manufacturing system is key to maintaining a healthy economy today. With the rapid development of science and technology and the progress of human society, the modern manufacturing system is becoming increasingly complex, posing new challenges to both academia and industry. Ever since the beginning of industrialisation, leaps in manufacturing technology have always accompanied technological breakthroughs from other fields, for example, mechanics, physics, and computational science. Recently, machine learning (ML) technology, one of the crucial subjects of artificial intelligence, has made remarkable progress in many areas. This study thoroughly reviews how ML, specifically deep (reinforcement) learning, motivates new ideas for addressing challenging problems in manufacturing systems. We collect the literature targeting three aspects: scheduling, packing, and routing, which correspond to three pivotal cooperative production links of today's manufacturing system, that is, production, packing, and logistics respectively. For each aspect, we first present and discuss the state-of-the-art research. Then we summarise and analyse the development trends and point out future research opportunities and challenges.

KEYWORDS

bin packing, combinatorial optimisation, deep reinforcement learning, job shop scheduling, manufacturing systems, vehicle routing

1 | INTRODUCTION

Combinatorial optimisation problems (COPs), as one important branch of mathematical optimisation, have practical applications in many fields, such as communication, transportation, manufacturing and aroused broad research in industrial engineering, computer science, and operations research. Due to the NP (non-deterministic polynomial-time) hardness, finding their optimal solutions is challenging. In specific, the discrete solution space in COPs renders the optimisation less efficient, without the guidance of gradient as in continuous optimisation. Meanwhile, the complexity of searching the (near-)optimal solution(s) among feasible solutions could exponentially increase as the problem scale grows. Classic methods, including exact algorithms and (meta-)heuristics, generally depend on massive expertise and tuning work to solve specific problems. They are

Cong Zhang, Yaoxin Wu, and Yining Ma are equal contribution.

also subject to impractical runtime and inferior results, especially on large-scale or hard COPs.

To tackle the above issues, learning-based optimisation methods flourish in recent years. They learn heuristics or key policies in COP algorithms in a data driven way, so that the resulting learning assisted methods are able to achieve high-quality solutions. The automatically learned heuristics or policies have potentials to surpass their classic counterparts via the effective design of neural networks and training algorithms. The logic behind the effectiveness of deep learning for COPs are as follows [1, 2]: (1) a class of problem instances could share similar structures with only their data drawn from an underlying distribution; (2) the deep (reinforcement) learning is conducive to learning heuristics or policies automatically given sufficient instances with similar problem structures. The first point is often satisfied in practice given the fact that similar problems are repeatedly solved everyday in manufacturing and daily life, for example, machine scheduling and object packing in production; the second point has been witnessed in the revolution by deep (reinforcement) learning, which has realise intriguing achievements in diverse tasks in terms of images, texts, audios, graphs etc.

Despite very early attempts that couple machine learning (ML) with COPs [3–5], recent deep learning-based methods present promising results, which are comparable to classic optimisation methods [6–8]. Their success has inspired miscellaneous deep models to cope with specific COPs, for example, vehicle routing [9–13], scheduling [14–17], bin packing [18–21], or enhance performance in solving general COPs, for example, integer programming [22–28] and constraint programming [29–31]. The corresponding literature on the methods for specific COPs are reviewed respectively in refs [32–35]. In addition, some surveys from the methodological perspective overview existing learning-based methods for COPs, for example, on generic usage of ML [1, 36, 37], graph neural networks [38–40], reinforcement learning (RL) [41–47].

In this survey, we review the learning-for-optimisation literature from the perspective of manufacturing. Compared to the previous surveys, we only centre on the existing learning-based methods for the most concerned COPs in manufacturing, that is, scheduling, bin packing, and routing problems, since (1) they are nearly the most studied ones in learning-for-optimisation community; (2) neural architectures and training algorithms for these problems share some associations, for example, similar sequence-to-sequence deep models are often used in both routing and bin packing; (3) they correspond to key procedures in the manufacturing process from goods production to delivery, which directly influence the final throughput and revenue [45, 48, 49]; (4) All of the problems are solved for everyday manufacturing, where deep learning has the most potential to aid for raising their efficiencies. In specific, the effective scheduling of machines and crew is paramount for a wise usage of resources and thus could significantly reduce the cost; a high-quality operational solution in the packing procedure empowers a high usage rate of containers and finally increases goods supply. The routing procedure directly influences the transportation of materials or goods, which plays critical roles in various manufacturing activities. Optimisation for routing undoubtedly renders the whole manufacturing process more fluently and reliably. On the other hand, all the above procedures happen regularly in daily manufacturing. The corresponding COPs need to be tackled in every day, with the relatively fixed problem structures but different input data, for example, the delivery locations for routing are varying in days with distances between locations changed. Therefore, deep learning could be leveraged to discover efficacious methods underlying the data distribution. Despite the overview of methods for tackling the three concerned COPs, we also discuss the gap between existing deep models and their applications in real situations and potential challenges that could be considered to tackle in future research. We also note that there are already some surveys for the usage of classic ML in manufacturing [50–52]. In contrast, this paper presents cutting-edge deep learning techniques for typical COPs in manufacturing, which have achieved superior performance to classic optimisation methods. In summary, the contributions of this paper are as follows:

- In this paper, we survey and present recent learning-for-optimisation literature on three specific COPs, that is, scheduling, bin packing, and routing problems, which are paramount joints in manufacturing. Compared to existing survey papers, for example, [1, 41], this paper covers the works to date and very recent state-of-the-art deep models for each problem.
- While some survey papers review broad applications of deep learning in a relatively coarse-grained way, for example, [40, 44], this paper spreads the learning-for-optimisation literature on the three most studied problems in a fine-grained way. We believe such a detailed review could inspire the development of deep learning in the other problem domains.
- Meanwhile, we do not limit our scope to single problem styles, for example, manufacturing/machine scheduling or transportation [45–47], since there are indivisible associations between neural architectures or training algorithms in methods for the involved three COPs, which we believe are critical to stimulating more advanced methods in learning-for-optimisation community.
- In contrast to existing surveys from the methodological perspective, we comprehensively present related works for each problem, which are not constrained to specific methods, for example, RL [41, 44–47], graph neural networks [38–40]. We hope it could grasp the whole picture for each problem, which readers would be interested in.

The detailed discrepancies between this paper and existing art are as follows. Compared with [41], we focus more on learning to solve COPs sourced from the domain of manufacturing. Mazyavkina et al. [41] discussed more general COPs, such as mixed-integer linear programs, while failing to present fine-grained reviews on manufacturing-oriented problems, for example, the job shop scheduling problem

(JSSP). Therefore, our review is more related and beneficial to audiences in the manufacturing domain. Panzer et al. [44] review a range of relevant disciplines designated for a production system and show that deep reinforcement learning (DRL) can address various tasks. However, we regard production as one of the essential links of the manufacturing system with the other two major components, that is, packing and logistics, and provide a more systematic view of the whole system by covering each of the links via reviewing the state of the art research respectively. Although the detailed analysis of scheduling problems in the production system is given in refs [45, 46], the literature introduced is relatively outdated compared with those in this paper. Furthermore, we focus on algorithms based on GNN and attention-based neural network, which are demonstrated as more suitable and promising for COPs in the manufacturing system, for example, for vehicle routing problem (VRP) [9]. Finally, Farazi et al. [47] specifically reviews the comprehensive application of RL in the broad transportation domain. Nonetheless, some covered disciplines, such as maritime transpiration and energy-efficient driving, are too domain-specific to be informative for audiences from manufacturing systems. In contrast, we focus on the topics in transportation most related to manufacturing, that is, logistics optimisation.

The reminder of this survey are arranged as follows. In Section 2, we introduce problem definitions and basic notions for the three COPs. In Section 3, we review the learning-based methods for scheduling problems, with a bias to paradigms for JSSP, one basic and generic scheduling problem. In Section 4, we review existing deep models for offline and online bin packing problems (BPPs) respectively. In Section 5, we elaborate neural solvers for a variety of routing problems. We conclude this survey in Section 6 with some discussions.

2 | PRELIMINARIES

In this section, we first introduce DRL, a novel ML paradigm for solving job shop scheduling, packing, and VRPs. Then, we state the three problems in the form of graph representation, on top of which existing works develop miscellaneous deep models. Since existing models mostly resort to DRL as training paradigms, we further present typical MDP formulations used commonly in literature.

2.1 | Deep reinforcement learning

DRL is the combination of RL [53] and deep learning [54]. RL is a subject of ML that deals with sequential decision-making. The basic idea of RL is straightforward: An agent (usually controlled by some policy π) interacts with its environment by taking actions. The environment reacts to the agent's actions and provides it with numerical feedback (rewards). The goal of the agent is to learn good behaviour such that the expected cumulative feedback (formulated as the state value function V) is maximised, which is usually achieved by learning an optimal policy π^* or an optimal state value function V^* explicitly (the optimal policy can be retrieved later from V^*). The interaction procedure between the agent and the environment can be described as the close loop in Figure 1.

In most cases for DRL, either the policy π_θ or the state value function V_θ is parameterised with some deep neural network with parameters θ. The former case is classified as the policy-based method, which includes REINFORCE [55], A3C [56], and proximal policy optimisation (PPO) [57], while the latter case is called the value-based method covering Deep Q Network (DQN) [58], double DQN [59], and duelling DQN [60]. We refer the audiences to ref. [61] for a comprehensive review of DRL methods.

2.2 | MDP formulations

To better describe the three problems, we provide basic problem statements and corresponding MDPs respectively. Please note that all MDPs in this section are concluded from existing literature, and they have been widely applied in existing RL methods for solving the problems.

2.2.1 | Job shop scheduling problems

Problem statement: Among the scheduling problems in manufacturing, the JSSP [62] is one of the generic models that has been vastly studied. In JSSP, a series of jobs are to be processed by a set of machines. Each job contains several operations that should be processed in a predefined order (i.e. the precedent constraints). The goal is to find the best permutations (processing orders) of operations on machines to achieve particular objectives while not violating the precedent constraints, such as minimising total production time (i.e. the makespan) or the total tardiness. The MDP formulation for the JSSP often varies from method to method, depending on the genre of the framework. Here we introduce one of the MDP modellings based on a construction heuristic exploited by most of the existing research.

State: The problem instances and partial schedules encountered during solving are usually represented with the disjunctive graph [63], a sparse directed graphical model with operations as nodes and the directed arc between any two operations denoting the processing order from the source to the target.

Action: The action set for the current state (disjunctive graph) is the subset of feasible operations that can be assigned to their designated machines. This operation assignment action is usually called *dispatching*.

Transition: The transition from one state to the next state is a deterministic function that changes the disjunctive graph according to the dispatched operation to accommodate the effect of dispatching. It updates the topology of the disjunctive graph, for example, new arcs are introduced to connect the dispatched operation with its preceding neighbour on the same machine, or the numerical features of each node, for example, the starting time.

Reward: The reward is closely correlated with the objective we would like to optimise. For example, one can define the reward as the value difference of the makespan of the current state to the previous state, such that the sum is the total makespan we want to minimise.

Policy: The stochastic policy takes the disjunctive graph representation of the current state as input. It selects one of the feasible operations to dispatch to the machine till the end of the episode.

Note that the above MDP model is a general formulation. Some existing works strengthen this MDP modelling by augmenting the disjunctive graph. For example, ScheduleNet [64] enhances the expressiveness of the disjunctive graph by introducing artificial machine nodes that include more information from the perspective of the resources (machines).

2.2.2 | Bin packing problems

Problem statement: Given a set of cuboid items $I = \{i_1, \ldots, i_N\}$, with each item i_n featured by its length, width and height (l_n, w_n, h_n), they need to be put into a bin B with length, width and height (L, W, H). The basic constraints are (1) the total capacity/weight of items cannot surpasses the maximal allowed capacity/weight of the bin and (2) items cannot overlap with each other. In addition, other practical constraints (e.g. regrading stability, precedence) are often involved in different problems [21, 65]. The construction heuristics for such problem could be described as the MDP defined below:

State: Since the items are put into the bin one by one, the state $s_t \in \mathcal{S}$ at the tth step should reflect both the status of the bin with the packed items I_t^p and the set of unpacked items I_t^u. Regarding the former, it comprises the configuration of the packed items in the bin, for example, the positions, orientations and sizes of packed items. Also, the height map is often used to reflect both the current distribution of packed items and utilisation rate in the bin. Regarding the latter, the sizes and orientations of unpacked items should be considered.

Action: At each state s_t, the action could be selecting the next item to pack from the unpacked ones $a_t^i \in \mathcal{A}_i$, determining the position of the selected item in the bin $a_t^p \in \mathcal{A}_p$, and selecting its orientation $a_t^o \in \mathcal{A}_o$. In the literature, the agent could only focus on one (e.g. $a_t = a_t^i$) or two actions (e.g. $a_t = (a_t^i, a_t^p)$), with the other executed by heuristics in the environment.

Transition: Given the action a_t, the selected item a_t^i is put into the bin, following its position a_t^p and orientation a_t^o, so that the state is deterministically updated as s_{t+1}, with a new status of the bin with the packed items $I_{t+1}^p = I_t^p \cup a_t^i$ and the remaining unpacked items $I_{t+1}^u = I_t^u - a_t^i$. Specially, $I_0^p = \varnothing$ and $I_0^u = I$.

Reward: According to different objectives to optimise, the reward functions are defined diversely. Here, we take the utilisation rate as the objective, which is mostly studied in the present learning-based methods. The reward function can be defined as $r_t = l_t w_t h_t / LWH$, where (l_t, w_t, h_t) is the size of the selected item a_t^i. The reward indicates the change of the utilisation rate when the item a_t^i is packed at the tth step. The interaction between the agent and environment alternately extends until the termination by the capacity/weight constraint, and the cumulative reward from step t of the episode is $R_t = \sum_{k=t}^{T} \gamma^{k-t} r_k$, with the discount factor $\gamma \in [0, 1]$. When $t = 1$ and $\gamma = 1$, $R_1 = \sum_{k=1}^{T} l_k w_k h_k / LWH$. Therefore, the goal of RL is maximising the expected utilisation rate $\mathbb{E}[R_1]$ over the distribution of episodes.

Policy: The stochastic policy π is represented by a conditional probability distribution $\pi = P(a_t^i, a_t^p, a_t^o | s_t)$. The stochastic episodes are produced by solving a class of BPPs, with the policy iteratively picking the action given each sate s_t. The episodes are used to optimise the policy by RL algorithm. According to the concerned actions, the policy for only one or two actions are trained in most literature, for example, $\pi = P(a_t^i | s_t)$ and $\pi = P(a_t^i, a_t^p, | s_t)$.

Please note that the above problem description and MDP is for a basic offline 3D bin packing. The instantiations of the MDP have been successfully applied for specific packing problems in some works [20, 66]. Other variants of packing problems and their corresponding MDPs (e.g. packing with online settings, multiple bins or continuous positions) can be found in literature [18, 49, 67, 68].

2.2.3 | Vehicle routing problems

Problem statement: Given a graph $G = (V, E)$ where V contains N nodes and E contains the possible edges between each node $i, j \in V$, the general routing problems consider finding the shortest route to serve N customers under several constraints. For example, the Travelling Salesman Problem (TSP) considers finding the shortest Hamilton cycle, that is, a tour that starts with one node, visits other node once and only once, and finally returns to the starting node. The prevailing MDP formulations of DRL-based solvers for routing problems can be classified into two types, that is, *construction* MDP and *improvement* MDP. Below we introduce them in detail.

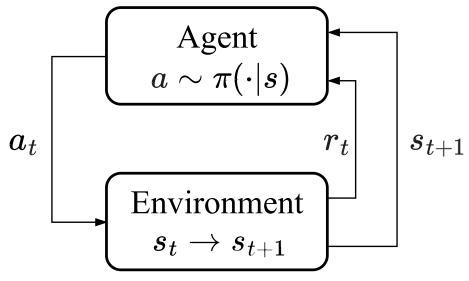

FIGURE 1 The close-loop interaction between the agent and the environment. For interaction at time step t, a_t, s_t, and r_t stand for the action, state, and reward respectively. s_{t+1} is the next state to which the environment transits. The agents' policy $\pi(\cdot|s)$ takes a state s as input and outputs an action a.

1) *Construction MDP*: Staring from an empty solution, the MDP describes a process of iteratively adding a node into the current partial solution until a valid solution is constructed.

State: The state $s_t \in S$ at time t reflects (1) node-level information for each customer node (i.e. coordinates, demand etc.), (2) global-level information about the route construction (e.g. the remaining capacity of the vehicle), and (3) the current partial solution $\{a_1, \ldots, a_{t-1}\}$ where a_i is the previously selected node (action) at time i. For initial state when $t = 0$, the partial solution is empty.

Action: At time t, the action is to choose a valid node in set V. However, we note that the action space is state-dependent, which means that not all nodes in V are available for selection given a particular state s_t. We mask the actions that lead to infeasible solutions so as to make sure all the constraints of the routing problems are satisfied. In the TSP example, we mask previously visited nodes to make sure that each node is only visited once.

Transition: The state transition rule is deterministic which adds the selected action a_t to the partial solution, that is, change the partial solution from $\{a_1, \ldots, a_{t-1}\}$ to $\{a_1, \ldots, a_{t-1}, a_t\}$ and then update the node-level and global-level information of the state accordingly.

Reward: In typical routing problems where the objective is to minimise tour length, the reward is commonly set to the negative value of the increase in tour length, such that the cumulative rewards would be the negative total tour length. The idea here is to ensure that maximising cumulative rewards is equivalent to minimising the tour length objective. This reward design is supposed to be updated for different VRP variants based on their specific objective function.

Policy: The policy π is usually parameterised by a deep model (i.e. a deep neural network). Given the input state s_t that describes the current partial solution and the information about the construction, the policy decides on an action a_t that selects a node. The entire episode concludes when all of the nodes are selected and a valid tour is constructed. In most cases, the policy is stochastic in the sense that it learns an action distribution for selecting each node.

2) *Improvement MDP*: Differently, this MDP describes a search process similar to the neighbourhood search [69] in traditional heuristics. Staring from a randomly generated solution, it iteratively performs rewriting operations (neighbourhood search) on the current solution to convert it to another ones, until near-optimal solutions are found. Below we give a possible MDP formulation for the above improvement process.

State: The state s_t of the MDP at time t would reflect: (1) node-level information for each customer node (similar to the construction MDP), (2) global-level information about the search (e.g. historical visited solution and its corresponding costs), and (3) the current solution δ_t. For the initial state s_0, the solution δ_0 can be any randomly generated one as long as it is feasible for the studied routing problem.

Action: The action a_t would be a specific operation that changes the current solution δ_t into another new one δ'_t. For example, according to refs [2, 70], the action is to specify two nodes (i, j) in V to perform a pairwise local search operation as shown in Figure 2. And according to refs [71, 72], the action is to control a ruin-and-repair operator to perform the neighbourhood search. This process is similar to the idea of neighbourhood search in the literature. Meanwhile, we note that masking is also applied to the action space in order to make sure the next solution is feasible.

Transition: In many existing works, the state transition rule is deterministic which will always accept the next proposed solution δ'_t as the solution of the next state. Meanwhile, the node-level and the global-level information of the state are updated accordingly.

Reward: The reward is typically set to be the immediate reduced objective value of the current best-so-far solution after taking the local search action, which ensures that the cumulative rewards would equal to the total improvement (i.e. reduced objective value) over the initial solution (for minimising optimisation).

Policy: The policy π is usually stochastic and parameterised by a deep model. Different from the above construction MDP, the time horizon in improvement MDP could be any user-specified values according to the time budget of users; thus a reward discount factor $\gamma < 1$ is usually needed. The best solution found throughout the whole improvement horizon was recognised as the final solution to the studied routing problem.

The instantiations of the above two MDPs have been successfully applied for specific routing problems in several representative works (e.g. in refs [2, 9, 10, 70, 72, 73]). Similar to scheduling and bin packing, we note that there are different MDPs used in the literature other than the above ones for routing problems. We refer to refs [73, 74, 75] as some examples.

3 | JOB SHOP SCHEDULING PROBLEMS

Fabrication is one of the essential procedures of modern manufacturing systems. Scheduling is a prerequisite for coordinating resources for tasks to increase productivity and reduce cost, critical for achieving efficient fabrication. The scheduling

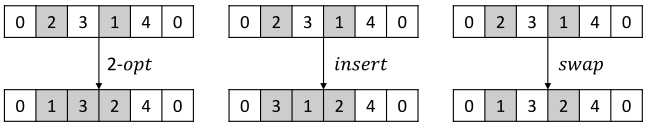

FIGURE 2 Three examples of pairwise neighbourhood search operators. This graph is retrieved from Ma et al. [70], where 2-opt reverses the segment between node i and node j; insert puts node i after node j; and swap exchanges the position of node i and node j.

problems are usually mathematically defined as COPs and thus can be solved precisely with exact methods such as mathematical programming and branch-and-bound [76]. However, most of the scheduling problems possess NP-hardness [77], making it too time-consuming to obtain the optimal solution for large-scale instances within a limited time for real-life scenarios in practice. Therefore, a wide range of so-called heuristic or metaheuristic methods [78] are developed, which trade off the solution quality with computational time. Although the heuristics and metaheuristics have improved to progressively mature since their emergence, the recent breakthrough of ML techniques and the increasing complexity of scheduling problems have aroused more interest in incorporating ML with traditional heuristic and metaheuristic methods for better performance [79, 80].

3.1 | Learning to solve JSSP

3.1.1 | Learning to generate PDRs for JSSP

Many algorithms have been proposed to solve JSSP. Among them, the priority dispatching rule (PDR) [81] is a class of frequently used heuristics that has extensive applications on JSSP due to its very fast computation and human interpretable expressions. A number of PDRs has been designed and implemented for JSSP instances with various production criterion and constraints [81]. However, manually designing PDRs can be tedious and mentally intensive since there are no guarantees that a single PDR or a subset is enough and effective to adapt for full spectrum of JSSP instances and objectives. From a data science point of view, the bright part is that the abundant JSSP instances with diverse sizes and distributions make it possible to utilise ML to generate PDRs automatically. Attributed to the generalisation ability of ML, machine learning models are expected to generate high-quality PDRs for a range of JSSP instances for given distributions, which alleviates the burden of human expert in designing PDRs manually.

In particular, L2D [14], GNNRL [82], and ScheduleNet [64] are three pioneering and representative works following this idea. L2Ds [14] model the solution-constructing procedure as a sequential decision-making problem and utilise DRL to solve it by learning to dispatch operations. Specifically, the JSSP instances and (partial) solutions are represented with disjunctive graphs [63], a sparse directed graphical model describing the processing orders among operations. Then, a DRL agent, usually parameterised by a GNN, learns the vectorised embeddings of the operations (nodes) in the disjunctive graph, based on which one of the feasible operations is selected and dispatched to its corresponding machine and make corresponding changes to the topology of the disjunctive graphs (states). Unlike the traditional PDRs that select operations based on a priority index computed by some rules, the DRL agent learns from the graph and outputs the operation in an end-to-end manner. Such learning ability allows the DRL agent to learn from copious instances and generalise to unseen ones, even those from other distributions. GNNRL [82] follows a similar strategy but with some differences. The states (disjunctive graphs) are static, and the architecture of GNN is different, which is a simple message-passing network. Compared with the other two works, ScheduleNet [64] is the first to propose introducing artificial machine nodes into the disjunctive graph to incorporate the work-in-progress (WIP) information for partial schedules. By learning the embedding of machine and operation nodes independently with a type-aware GNN model, ScheduleNet achieves state-of-the-art performance due to more expressive graph representation and more effective graph embeddings. JSSenv [83] is a delicately designed and well-optimised simulator for JSSP, extended from the OpenAI gym's environment suite [84]. Instead of employing the disjunctive graph, JSSenv models and represents the states of partial schedules with Gantt charts [85]. JSSenv then proposes a DRL agent to solve JSSP instances individually in an online fashion. However, it is an online method that performs training for each individual instance, hence is slower in computation compared with other works which can quickly infer the solutions to given instances after trained offline.

Prior to DRL, researchers employ genetic programming (GP), a kind of classic ML algorithm to learn to generate PDRs, for example, in refs [86, 87]. While showing promising results, GP-based methods generate a fix rule that applies to all states. In contrast, DRL-based methods generate policies that are adaptive to each scheduling states, hence have more potential in optimising the performance.

3.1.2 | Learning to select PDRs for JSSP

Instead of learning PDRs explicitly, several works tend to learn to select from a subset of PDRs. Lin et al. [88] propose a DQN [58] (one of the well-known DRL methods for discrete control)-based method, to select PDR from a pool of candidates for each machine. Each machine is now treated separately, and a centralised DQN agent learns to pick dispatching rules for each machine respectively. Since the rule set is predefined and fixed during learning and testing, the DQN agent's performance is highly dependent on the quality of the rules in the set. Compared with learning PDRs explicitly, which theoretically can search the whole PDR space, learning to select rules has limited exploration (only covering a subset of the space), and the performance is therefore relatively less competitive. However, while learning PDRs loses its interpretability as it is difficult to reason why a particular operation is dispatched since the action is generated with a deep neural network, learning to select PDRs is more understandable to human experts.

3.2 | Learning to solve JSSP with flexible machines

In order to be more in line with the actual situation of manufacturing systems in real life, some challenging but essential features and constraints are needed to be incorporated

when solving JSSP. For example, flexible machines and dynamical events are two classes of constraints often considered. The flexible machines (or parallel machines in some literature) are a group of identical machines for processing jobs. Compared with classical JSSP, where each job has its designated machines to process, the job with multiple flexible machines must be assigned to one of the eligible machines before deciding its processing order on the selected machine. The JSSP with flexible machines is termed as the flexible JSSP [89], or FJSSP in short. Recently, several works have been devoted to exploring FJSSP solving with ML techniques. In ref. [90], Müller et al. propose a ML algorithm for selecting constraint programming solvers for FJSSP. In their study, they find that the CPLEX from IBM and OR-Tools from Google are the two best solvers for solving FJSSP in terms of solution quality and computational time; however, their performance is complementary to each other. According to this observation, they propose two ML algorithms based on decision trees (traditional ML) and convolutional neural networks (CNNs) (deep learning) to select CPLEX or OR-Tools solver based on the instance features and parameters. The comprehensive experiments show that the solver-selection approach outperforms using any of the two solvers. In ref. [91], Song et al. propose a DRL-based method to automatically learn high-quality PDRs for FJSSP end-to-end, without relying on existing manual PDRs. They design a heterogeneous graph structure to represent machine and job status during the construction process, and propose a heterogeneous graph attention network to extract graph embeddings for decision making. Results on both synthetic and public benchmarks show that the learned policy performs better than traditional manual designed PDRs. Besides utilising deep architectures, the traditional ML algorithm has also demonstrated effectiveness in learning for solving FJSSP. RANFORS [92] proposes a random forest-based algorithm for extracting PDRs from the best schedules generated with mixed-integer linear programming, constraint programming, or hybrid genetic algorithm, with the aim of minimising the average total weighted tardiness of jobs. The experiment results show that the learned PDRs outperform the existing dispatching rules.

3.3 | Learning to solve JSSP with dynamic events

Accidents can happen anytime in such a complex system as the production system, such as new jobs arriving or machines malfunctioning, which poses more challenges for ML-based solvers. The JSSP with dynamic events is called dynamic JSSP (DJSSP). Some methods start considering various dynamic events to make the scheduler more robust to these disturbances. Zeng et al. [94] consider machine breakdown and different order requirements in JSSP, where a machine can break or the configuration of jobs may change. They formulate the DJSSP as an MDP with the disjunctive graphs as the states and a set of PDRs as the action space. A graph representation learning module based on the self-attention mechanism [97] extracts the embeddings of each operation for the current state. Unlike previous works with GNN as the backbone embedding networks that aggregate information from neighbours by following the topology of the disjunctive graph, they propose to exploit self-attention as the embedding operator where each node pays attention to every other node regardless of their connections in the disjunctive graph. Then a DRL agent based on double duelling DQN with prioritised replay and noisy networks (D3QPN) algorithm learns to pick rules from the action set. The experiment results show that D3QPN outperforms traditional PDRs and a genetic algorithm-based metaheuristic method. Their analysis of the effect of different graph representation learning methods, that is, matrix-based, GNN-based, and attention-based methods, shed some light on the representation learning for JSSP.

3.4 | Other shop-level scheduling problem

In addition to JSSP, there are efforts to approach other types of shop-level scheduling problems with DRL. MatNet [95] considers a flexible flow shop problem (FFSP). In FFSP, a set of jobs consists of multiple stages, each of which is to be processed by a set of flexible machines. The problem instances are encoded by a series of matrices, each corresponding to a stage containing the processing time of each job on each flexible machine at that stage. Then a matrix encoding network (MatNet) based on self-attention mechanism [98] independently embeds each stage's matrix and outputs the permutation of jobs on each machine for that stage. The experiment results reveal that MatNet is better than the CPLEX solver and a particle swarm-based metaheuristic method in terms of speed and optimality gap. However, MatNet makes an assumption that the machines are not shareable across stages, which is less generic than JSSP. MGRO [15] learns a policy to aid iterated greedy, an improvement heuristic, to solve the hybrid flow shop scheduling problem. PFSPNet [96] solves a simpler variant of the scheduling problem called the permutation flow shop scheduling problem, where the solution of a problem instance is a permutation of all jobs. PFSPNet consists of an encoder and a decoder; all built based on the classical RNN network. By taking the problem instance as input and embedding it with the encoder, PFSPNet extracts the workflow for jobs as vectors, which are then fed into the decoder to decode a permutation of jobs. Large-scale problem instances are tested for up to 200 jobs and 20 machines. The experiment results are promising compared with other heuristic and metaheuristic baselines.

3.5 | Future directions

We have reviewed the state-of-the-art deep (reinforcement) learning-based methods for solving JSSP and its variants in previous sections, for which we summarise the information in Table 1. Nevertheless, the performance of existing learning-based methods can be enhanced with the advanced searching

TABLE 1 Deep (reinforcement) learning-based methods for JSSP and its variants in manufacturing systems

Method	Year	Venue	Problem	Constraints or dynamic events	Objective	Machine leaning algorithm	Problem encoding
L2D [14]	2020	NeurIPS	JSSP	NA	Makespan	PPO	Disjunctive graph
GNNRL [82]	2021	IJPR	JSSP	NA	Makespan	PPO	Disjunctive graph
ScheduleNet [64]	2022	Preprint	JSSP	NA	Makespan	REINFORCE	Disjunctive graph
JSSenv [83]	2021	ICAPS	JSSP	NA	Makespan	PPO	Gantt chart
Lin et al. [88]	2019	TII	JSSP	NA	Makespan	DQN	Matrix
EAS [13]	2022	ICLR	JSSP	NA	Makespan	PPO	Disjunctive graph
Iklassov et al. [93]	2022	Preprint	JSSP	NA	Makespan	Actor critic	Vector
Müller et al. [90]	2022	EJOR	FJSSP	Flexible machines	Makespan	Decision trees, supervised DL	Manually engineered features
Song et al. [91]	2022	TII	FJSSP	Flexible machines	Makespan	PPO	Heterogeneous graph
Zeng et al. [94]	2022	Preprint	DJSSP	Machine breakdown, different job requirement	Makespan	Double duelling DQN	Disjunctive graph
MatNet [95]	2021	NeurIPS	FFSP	Flexible machines	Makespan	REINFORCE	Vector
MGRO [15]	2021	SIGKDD	HFSP	NA	Makespan	PPO	Multiple graphs
PFSPNet [96]	2021	TETCI	PFSP	NA	Makespan	Actor critic	Matrix

Abbreviations: JSSP, job shop scheduling problem; PFSP, permutation flow shop scheduling problem.

strategy and generalisation power. Efficient active search (EAS) [13] proposes three EAS strategies to adjust either the problem embedding learned by the encoder, the residual layer added to the decoder, or the parameters of a lookup table to improve the performance of the neural COP solvers. In this way, EAS only adjusts a small subset of (model) parameters during search, which is faster than the original active search (AS) framework that updates the whole set of parameters. Experiments on a range of COP problems, such as vehicle routing and JSSP, demonstrate that EAS is a practical framework for dramatically improving the performance of existing neural solvers. Another direction deserving attention is to increase the generalisation ability. Iklassov et al. [93] come up with an adversarial curriculum learning (ACL) strategy for addressing the catastrophic forgetting when generalising between different problem sizes for JSSP. Specifically, ACL is trained starting from the most accessible problems (smallest problem size) and revisits the worst-solved instances whose difficulty (problem size) is proportional to the distribution of performance. When the model's performance is above a threshold for the current difficulty (problem size), the difficulty level is increased by one (i.e. the next problem size). ACL has shown outstanding performance improvement against L2D across the full range of problem sizes.

Generalisation ability is a challenging topic for ML research in general. Although it has been extensively investigated in other ML domains, it is less studied under the context of the COP problem, especially for intelligent scheduling systems. In contrast, the diversity of complex production systems eagerly craves a robust ML model that can handle various requirements and scenarios. Taking a step back, performance of the existing ML-based scheduling methods is less satisfying than mature traditional solvers, which may be attributed to the unsuitable problem representation and weak problem representation learning of existing models. Unlike VRPs for which self-attention [9] is a well-recognised backbone model, JSSP still lacks such a backbone model that can effectively learn problem representations for various scheduling problems. These are the current challenges for the future research of intelligent scheduling in manufacturing systems.

4 | BIN PACKING PROBLEMS

In the manufacturing industry, packing is critical in different scenarios such as container packing, truck loading, warehouse stacking etc. [99–101]. The improvement of efficiency in these joints could largely reduce the cost in manufacturing and promote the revenue for a company. Different taxonomies are applicable in the domain of bin packing according to item dimensions/shapes, objective functions etc. In this section, we review the learning-based methods in the aspect of offline/online BPPs, which associate with whether the upcoming items for packing are known in prior. It determines the packing action space, the use of historical data and training paradigms, which should be considered in the employment of learning techniques. We also summarise the hyper-heuristics for BPPs in the literature since they reflect the early mindsets to solve the problems with the aid of learning-based techniques. For your convenient retrieval, we present the taxonomy of deep learning-based methods for bin packing in Table 2. We elaborate them below.

TABLE 2 A taxonomy of deep learning-based literature for bin packing

Method	Year	Venue	Problem	Objective	Deep model	Policy to learn	Training algorithm
Hu et al. [19]	2017	Preprint	Offline 3D-BPP	Least surface area	Pointer net	Sequence	REINFORCE
Laterre et al. [102]	2018	AAAI workshop	Offline 2D~3D-BPP	Least surface area	Feed-forward network	Position & orientation	R2
Duan et al. [103]	2019	AAMAS	Offline 3D-BPP	Least surface area	Pointer net	Sequence & orientation	PPO + SL
Zhang et al. [104]	2021	Preprint	Offline & online 3D-BPP	Maximal utilisation rate	Modified transformer	Sequence & orientation	REINFORCE prioritised sampling
Hu et al. [67]	2020	ACM TOG	Offline 2D~3D-BPP	Compactness & Pyramidality & Stability	RNN + attention	Sequence & orientation	A2C
Li et al. [105]	2019	Preprint	Offline 2D~3D-BPP	Maximal utilisation rate	Modified transformer	Sequence & orientation & position	A2C + GAE
Jiang et al. [20]	2021	TCYB	Offline 3D-BPP	Maximal utilisation rate	Modified transformer + CNN	Sequence & orientation & position	A2C + GAE
Yang et al. [68]	2021	Hindawi MPE	Offline 1D~3D-BPP	Maximal utilisation rate	-	Sequence	Q learning
Duan et al. [49]	2022	Preprint	Offline 3D-BPP	Maximal utilisation rate	Modified Pointer net	Sequence	REINFORCE
Kundu et al. [106]	2019	IEEE RO-MAN	Online 2D-BPP	Maximal utilisation rate	CNN	Sequence	Double DQN
Verma et al. [65]	2020	Preprint	Online 3D-BPP	Maximal utilisation rate	Feed-forward network	Position	Double DQN
Zhao et al. [21]	2021	AAAI	Online 3D-BPP	Maximal utilisation rate	CNN	Position	ACKTR
Zhao et al. [18]	2021	ICLR	Online 3D-BPP	Maximal utilisation rate	GAT	Position	ACKTR
Yang et al. [107]	2021	IROS	Online 3D-BPP	Maximal utilisation rate	MLP	Position	PPO
Zhu et al. [108]	2021	ACM CIKM	Online 3D-BPP	Maximal utilisation rate	CNN	Position	SL

4.1 | Off-line BPPs

Inspired by Pointer Networks [6, 7], Hu et al. propose the first trial to solve a 3D BPP (3D-BPP) by deep learning [19]. They take the sizes of items as input to the encoder, and employ the decoder to output the sequence in which the items are packed into bins. Laterre et al. also attempt to determine the sequence of items for 2D-BPP and 3D-BPP by an adversarial self-play in single-player games, which is extended from two-player games (e.g. go and chess) [102]. They transform standard rewards in MDP according to if a reward surpasses a percentile of historical rewards, so that the agent always aim to achieve better performance than its latest version. Duan et al. learn the policy to select orientations for items in addition to the sequence policy, and adapt Pointer Network with the intra-attention which involves the information in previous decoding steps in the current decision [103]. The above three works minimise the surface area of a flexible rectangular bin, given soft packing materials. In contrast, various methods are proposed to maximise the utilisation rate of bins. Zhang et al. apply a transformer-style neural network to learn the sequence policy and an additional attention-based network to determine positions to put each item. They use prioritised oversampling to improve the training and evaluate their method for offline/online 3D-BPPs [104]. Hu et al. resort to an encoder-decoder neural architecture to learn the selection of the pair of an item and its orientation [67]. They consider a special case where the items are initialised by a spatial configuration, which leads to a preference constraint between items, for example, an item can only be selected to pack when all items on top of it have been packed. There are also some endeavours to simultaneously learn policies for selecting the item, its rotation and position. In specific, Li et al. decompose the actions for combinations of item, rotation and position into sub-actions for each dimension, and train the respective policies by an actor-critic algorithm with generalised advantage estimation [66, 105]. This method is evaluated on offline/online 2D and 3D-BPPs. Similarly, Jiang et al. design a sequence-to-sequence neural network with three decoders to be trained as three policies, and in the encoder a CNN is used for embedding the status of bins along with an attention mechanism for embedding the items [20]. In addition, Yang et al. focus on the objective of minimum bin slack in 1D~3D-BPP, in which Q-learning is used to train selection policy for items [68]. Duan et al. convert 3D-BPP into a set covering problem, considering to pack the current order with historical packing patterns [49]. In this way, practical and complex constraints can be readily handled, since they have been satisfied in the historical successful packs. The set covering problem is solved by the column generation algorithm, in which the authors further accelerate the solving of pricing subproblems, by selecting candidate packs with a modified Pointer Net.

4.2 | On-line BPPs

The recent study on on-line BPPs by deep learning may start from [106], in which a CNN-based Q network is used to learn the position policy in 2D-BPPs. Verma et al. claim a first trial to solve online 3D-BPPs with real-world physical constraints. They limit the candidate positions in a bin to put the items by a heuristic strategy, and then use DQN to select the action pair of the position and orientation [65]. This work also discusses multiple constraints and criteria that should be considered in online BPPs. Zhao et al. discretise the bottom of a bin as the grid, with grid points to be selected for locating the front-left-bottom corner of a box, and optimise space utilisation rate in 3D-BPP by an on-policy actor-critic algorithm [21, 109]. This method is limited to integer constrained box sizes and employ an additional neural network to predict infeasible grid points for pruning solution space. To handle continuous sizes of boxes, the authors also propose a tree search-based learning method, where leaf nodes represent candidate placements for the next item, and they use attention-based neural network to select a leaf node and spread the tree [18]. Yang et al. reflect the promising candidate actions (derived from heuristics) as MDP rewards for the agent, so as to guide the policy learning via PPO [107]. They develop a robotic system to evaluate their method on real-world packing tasks, which is shown effective to pack variable-sized products. Zhu et al. extend the online 3D packing into realistic settings with hundreds of items to pack and dozens of constraints considered [108]. The learning-based method is based on a tree search framework, where a CNN is trained to prune nodes (packing actions) that could deliver sub-optimal solutions.

4.3 | Hyper-heuristics

Except the above learning-based methods, some attempts that learn to solve BPPs are developed in the hyper-heuristic domain. Hyper-heuristics is a class of artificial intelligence techniques to assist the automatic design of heuristics [110]. Despite standard solution-oriented heuristics, hyper-heuristics are high-level heuristic-oriented methods, which aim to wisely use low-level heuristics to solve problems for better performance. There are already some hyper-heuristics applied to BPPs. Mao et al. employ a selective hyper-heuristic to solve 1D-BPPs, and use a multilayer perceptron (MLP) to predict the positioning heuristic from the candidates for a given instance [111]. López-Camacho et al. propose the representations of hyper-heuristics (combinations of low-level heuristics) by chromosomes and search the most promising hyper-heuristic by genetic algorithm for a class of instances in 1D-BPP or 2D-BPP [112]. Similarly, Duhart et al. search the hyper-heuristic for 2D-BPP with the ant colony optimisation (ACO) [113]. Silva-Gálvez et al. use k-means to cluster solving states of instances and statisticise scores of candidate positioning heuristics on these clusters, so that the final learned hyper-heuristic naturally selects the heuristic with highest score [114]. The above methods belong to selective hyper-heuristics and aim to select low-level heuristics according to instances or solving states of instances. On the other hand, generative hyper-heuristics automatically construct new heuristics from components of existing heuristics. A few trials use GP to

discover effective constructive heuristics for BPPs [115, 116]. All the above methods are designed for off-line BPPs and a few hyper-heuristics solve the online version with similar techniques [117–119]. Compared to deep learning, hyper-heuristics generally lack semantic or informative representations of instances or solving states, and are less efficient in training, especially with a large quantity of instances.

4.4 | Future directions

In our survey, we find the current learning-based methods have achieved appealing results on both offline/online BPPs, generally with superior performance to classic hand-crafted construction heuristics. However, there are still some challenges that should be well tackled in the future study. We state the related issues and the potential research directions in the following aspects.

Complete state representation: Almost all existing methods are developed based on DRL, which must learn state representation for the status of both the bin and items. Such representation could greatly influence the decision in packing process and thus the final performance. However, existing methods might only involve partial information when they learn single policy, for example, sequence or position policy. In fact, most literature admit the sequence, positions and orientations of items are intertwined, which could together influence the final packing performance. Hence, the aggregation of embeddings for these aspects could increase the final performance even if single policy is trained.

In addition, since the utilisation rate of bins at each step are greatly influenced by critical points of currently packed items, that is, the points with the most values along each dimension, it is natural to discover the spatial relation between them in the encoding, especially for 3D-BPP. The GNN could be an elegant choice, which we find is scarcely used in the literature.

Large (continuous) action space: Compared to scheduling and routing problems, the action space in BPP is the combinations of items, positions and orientations. However, very a few current works learn to decide on all three aspects [20, 66], since it will cause the issue of the large action space. This could be more intractable when continuous positions are considered [18]. From the perspective of DRL, such large action space could be potentially tackled by the algorithms for continuous action space, for example, DDPG, TD3, and decomposition techniques. We refer readers to representative works in these domains [120–123].

Practical constraints & criteria: Existing literature mostly focus on simple BPPs without the consideration of practical constraints. This issue is pointed out in recent works [49, 108]. Except the standard constraints in BPP, for example, capacity/weight constraints, overlap constraint, it could be a valuable direction to cope with realistic constraints in learning, so that the trained models could be used in real-world settings. The additional constraints are different in varying domains, for example, physical stability with robots [65], customer requirements in delivery [49], and preference constraints in the style of picking and packing [67]. Regarding the criteria in BPP, existing methods mostly focus on minimising the utilisation rate and bin number. Some other objectives are also critical to consider in the optimisation, for example, stability, compactness, pyramidality [67]. Thus it could be worthwhile to extend multi-objective learning techniques for solving BPPs to gain more reliable performance.

5 | VEHICLE ROUTING PROBLEMS

Another critical part of modern manufacturing is logistics, which assures the availability of raw materials before production and the efficacy of product distribution after manufacturing. Logistics services that are efficient, smooth, and low-cost may reduce operating expenses, boost profits, and eventually promote manufacturing development. For decades, the logistic scenarios are modelled and studied as a classic family of COP called VRPs, also known as the routing problems [124]. Due to the NP-hard nature, VRPs are considered unlikely to be solved optimally within running time polynomial in the problem size [125], making exact methods unfavourable in solving industrial-scale VRP instances. To this end, heuristic algorithms such as simulated annealing (SA) [126], neighbourhood search [127, 128], and evolutionary algorithms [129, 130], were proposed to find near-optimal solutions for VRPs with much shorter time, where hand-engineered designs and rules by human experts are leveraged to define the search process.

Recently, the concepts such as Logistics 4.0 [131] and smart logistics [132] have embarked in both industry and academia, which has garnered increased attention in designing learning-based solvers for routing problems. After years of development, these neural solvers have resulted in promising performance on numerous routing problems in terms of both inference time and solution quality [10, 13, 72, 133–135]. Overall, the main motivation is that the conventional hand-engineered heuristics can be effectively substituted by data-driven neural solvers through automatic ML, hopefully with several benefits: (1) the heuristic patterns learned by neural solvers straight from data may be more efficient than human designs; (2) the GPU-accelerated neural solvers tend to exhibit a faster speed than conventional CPU-based heuristics, especially when used to solve a large amount of instances in parallel; (3) human knowledge regarding the target routing problem is often reduced to a minimum while designing such neural solvers, allowing neural approaches to have potential to be swiftly adapted for learning to solve new VRP variant; in another word, neural approaches are some new general tools for solving routing problems.

The majority of early initiatives of neural solvers were centred on supervised learning (SL). In the attempts by refs [6, 136, 137], they successfully showed feasibility of learning faster neural solvers based on SL. However, SL requires proper training dataset (i.e. a set of VRP instances and their corresponding optimal solutions) and the presence of human

solvers, which prevents its direct application to hard-to-solve and large-scale VRPs. Nevertheless, even in some recent works (e.g. in refs [133, 135, 136, 138, 139]), SL is still the most efficient learning paradigm for training deep models for small-scale and fundamental VRP tasks (e.g. TSP). Another learning paradigm is unsupervised learning (UL), which was used in refs [134, 140, 141] to (help) learn deep models for VRPs. Besides SL and UL, more recent works are focused on the state-of-the-art DRL methods. DRL learns heuristics without human guidance and manually labelled data, which shows potential to learn powerful and efficient solvers even for complex, large-scale or currently unsolvable routing problems.

Consequently, this section mainly focuses on reviewing DRL-based approaches for solving routing problems. We also discuss SL in Section 5.2.3 since it is still essential for some recently developed neural solvers. Unsupervised learning is not highlighted since it was not actively used in this field. Meanwhile, we limit the review scope to research works that learn *neural heuristics* since these methods are more suitable for industrial-scale VRP problems in manufacturing we are concerned in this review. There existing some other works that leverage neural networks to assist *exact* approaches. For example, Furian et al. [172] proposed a ML-based branch and price framework which uses neural networks to predict the branching scores in assisting variable selection for branching. Since we will not cover this direction, we refer to a recent survey by Lodi et al. [173] for more details of *neural exact* approaches. In Table 3, we list a summary of recent DRL-based heuristic approaches for VRPs. Other ML assisted heuristic approaches for VRPs are depicted in Table 4. Below we summarise existing learning-based solvers from the aspects of their target VRP variants and types of learnt neural solvers, respectively, followed by some discussions on future directions.

5.1 | Target variants of routing problems

5.1.1 | Classical routing problems

We first discuss several classical routing problems of interest for manufacturing.

As the simplest routing problem, TSP was the first research focus in the literature. TSP considers to visit N nodes (locations) exactly once, and finally return to the original one, where the objective is to minimise the total travel distance of the route. Bello et al. [7] presented the first DRL-based solver for solving TSP. Using the negatives tour length as the reward signal of RL, their RL method has shown significant improvement over SL [6] and outperformed Google's solver OR-Tools [174] on TSP100 (TSP with $N = 100$ nodes). Dai et al. [142] then proposed S2V-DQN based on graph embedding network, which delivered better performance compared to hand-crafted heuristics such as 2-opt, nearest/farthest insertions etc. Since then, considerable works have focussed on increasing the optimality gaps of DRL-based neural solvers utilising three benchmark problems: TSP20, TSP50, and TSP100 (e.g. Refs [2, 8, 9, 10, 70, 146–148, 154, 163]). Besides the above works that focus on learning efficient TSP solvers from scratch, researchers are also interested in the scalability and the generalisation of these learned solvers on large-scale instances where conventional heuristics fail to address well. Drori et al. [151] showed that their learnt neural solver can solve TSP in approximately linear time complexity as the problem size increases. Fu et al. [138] demonstrated that they can generalise a pre-trained model on small-scale TSP (e.g. TSP50) to large TSP instances with up to 10,000 nodes using the proposed graph sampling and heat maps merging approaches.

Another representative routing problem is the Capacitated Vehicle Routing Problem (CVRP), which first appears in Dantzig et al. [175]. It considers multiple vehicles and a capacity constraint on the basis of TSP. In a CVRP solution, each vehicle needs to fulfil the demands of some customers by departing from the depot, visiting the customers, and finally returning back to the depot, where the cumulative demands of visited customers in one sub-route cannot exceed the given vehicle capacity. Existing works for solving CVRP are usually tested on CVRP20, CVRP50, and CVRP100. Nazari et al. [8] extended the work of Bello et al. [7] and became the first learning-based method for CVRP. In another concurrent work by Kool et al. [9], Google's OR-Tools has been significantly surpassed by the learning-based Attention Model (AM). Further integrated with multiple decoders proposed in Xin et al. [148], or POMO training [10], the performance of AM for CVRP was further improved. Specifically, POMO achieved an averaged gap of 0.32% to the hand-crafted LKH solver [176] on 10,000 CVRP100 instance with much shorter computational time (2 min vs. 12 h) [10]. Different from the above works that learn neural heuristics in a purely data-driven way (i.e. without any human experience), Lu et al. [73] proposed a framework called L2I that combined the learning-based methods with the traditional highly-optimised local search operators. Despite requiring much more inference time, L2I becomes the first DRL-based CVRP solver that beats classical strong solver LKH [176] in terms of the objective values. In a recent work by Li et al. [135], they provided the first neural approach for large-scale CVRP. By learning to decompose large CVRP instances into sub-problems, the proposed *learning-to-delegate* framework that might speed up any hand-crafted CVRP solver by 10× to 100× while maintaining competitive solution quality. We note that the *learning-to-delegate* may also be applied to the above learning-based CVRP solver.

Pickup and Delivery Problems (PDPs) are also common in manufacturing logistics. On the top of TSP, the pickup-and-delivery TSP (PDTSP) considers a set of service requests, each of which is defined by a pickup node, a corresponding delivery node, and a precedence constraint that requires pickup to be conducted before delivery. The PDTSP was tackled through DRL for the first time in ref. [155], where a novel heterogeneous attention mechanism was introduced to AM. In ref. [169], the multi-agent RL was explored to solve PDTSP with capacity constraint. In a recent work by Ma et al. [72], the

TABLE 3 Deep reinforcement learning-based heuristic approaches for VRPs

Method	Year	Venue	Problems	Problem size trained on	Heuristic type	Deep model	DRL algorithm	Generalisation (size)	Generalisation (benchmark)
Bello et al. [7]	2017	ICLR workshop	TSP	{20, 50, 100}	Construction + sampling Construction + active search	RNN (Pointer net)	A2C	NA	NA
S2V-DQN [142]	2017	NeurIPS	TSP	[50, 100]	Construction + greedy	GNN (graph embedding)	DQN	Up to 1200	TSPLIB
Nazari et al. [8]	2018	NeurIPS	TSP, CVRP	{20, 50, 100}	Construction + sampling	RNN (Pointer net) + Att	REINFORCE	NA	NA
AM [9]	2019	ICLR	TSP, CVRP, SDVRP, OP, (S)PCTSP	{20, 50, 100}	Construction + sampling	Modified transformer	REINFORCE	Up to 120	NA
NeuRewriter [11]	2019	NeurIPS	CVRP	{20, 50, 100}	Improvement (swap)	RNN + Att	A2C	{20, 50, 100}	NA
L2I [73]	2019	ICLR	CVRP	{20, 50, 100}	Improvement (hybrid)	Att	REINFORCE	NA	NA
MODRL/D-AM [143]	2019	ISICA	MOTSP	{20, 40}	Construction + sampling	Transformer (AM)	REINFORCE	Up to 200	TSPLIB
James et al. [144]	2019	TITS	DPDP	[6, 10]	Construction + sampling	RNN (S2V-DQN) + RNN (Pointer net)	A3C	Up to 200	Real data
NLNS [71]	2020	ECAI	CVRP, SDVRP	{20, 50, 100}	Improvement (ruin-and-repair)	Att	A2C	Up to 300	Benchmark [145]
d O Costa et al. [146]	2020	ACML	TSP	{20, 50, 100}	Improvement (2-opt)	RNN + Att	A2C	Up to 300	TSPLIB
Xin et al. [147]	2020	TII	TSP, CVRP	{20, 50, 100}	Construction + sampling	AM/Pointer net	REINFORCE	Up to 300	TSPLIB
MDAM [148]	2020	AAAI	TSP, CVRP, SDVRP, OP, (S)PCTSP	{20, 50, 100}	Construction + beam search	Transformer (AM)	REINFORCE	NA	NA
POMO [10]	2020	NeurIPS	TSP, CVRP	{20, 50, 100}	Construction + augmentation	Transformer (AM)	POMO	NA	NA
Zhao et al. [149]	2020	TITS	CVRP, VRPTW	{20, 50, 100}	Construction + local search	RNN + Att	A2C	NA	NA
MAAM [150]	2020	TRPC	VRPSTW	{20, 50, 100, 150}	Construction + sampling	Transformer (AM)	MARL	NA	NA
Drori et al. [151]	2020	ICMLA	TSP	{100}	Construction	GNN + Att	REINFORCE	Up to 1000	TSPLIB
DRL-MOA [152]	2020	TCYB	MOTSP	{40}	Construction + sampling	RNN (Pointer net) + Att	A2C	Up to 500	TSPLIB

(Continues)

TABLE 3 (Continued)

Method	Year	Venue	Problems	Problem size trained on	Heuristic type	Deep model	DRL algorithm	Generalisation (size)	Generalisation (benchmark)
DRLSA [153]	2020	ICAPS	DSVRP	Around 22	Online decision	MLP	DQN	NA	Real data
MRAM [154]	2021	TCYB	TSP, CVRP, SDVRP, OP, PCTSP	{20, 50, 100}	Construction + sampling	Transformer (AM)	REINFORCE	{20, 50, 100}	NA
Heter-AM [155]	2021	TITS	PDTSP	{20, 40, 80, 120}	Construction + sampling	Transformer (AM)	REINFORCE	{20, 40, 80,120}	NA
Lin et al. [156]	2021	TITS	EVRPTW	{10}	Construction + sampling	RNN (Pointer net) + Att	REINFORCE	Up to 100	NA
Wu el al. [2]	2021	TNNLS	TSP, CVRP	{20, 50, 100}	Improvement (2-opt, swap, insert)	Modified transformer	A2C	Up to 200	TSPLIB, CVRPLIB
Zhang et al. [157]	2021	TNNLS	DTSP, DPDP	Up to 100	Construction + sampling	Transformer (AM)	REINFORCE	NA	Real data
Li et al. [74]	2021	TCYB	HCVRP	{80, 100, 120, 140, 160}	Construction + sampling	Transformer (AM)	REINFORCE	{80, 100, 120,140, 160}	CVRPLIB
MODRL/D-EL [158]	2021	IJCNN	MOVRPTW	{50}	Construction + sampling	Transformer (AM)	REINFORCE + EL	Up to 100	Solomon
RNN-AM [159]	2021	SMC	TDTSPTW	{20, 40}	Construction + sampling	Transformer (AM)+ RNN	REINFORCE	NA	NA
SOLO [160]	2021	SOCS	Online CVRP	{20, 100}	Online decision	GNN	DQN	NA	NA
ST-DDGN [161]	2021	ACDE	DPDP	Up to 600+	Online decision	GNN + Att	Double DQN	NA	Real data
LCP [12]	2021	NeurIPS	TSP, CVRP, PCTSP	{20, 50, 100}	Construction + improvement	AM/Pointer net	REINFORCE	Up to 500	TSPLIB
MatNet [95]	2021	NeurIPS	ATSP	{20, 50, 100}	Construction + augmentation	Transformer (POMO)	POMO	NA	NA
Ma et al. [162]	2021	NeurIPS	DPDP	{50, 300, 1000}	Online decision	MLP + Att	DQN + REINFORCE	Up to 1000	Real data
DACT [70]	2021	NeurIPS	TSP, CVRP	{20, 50, 100}	Improvement (2-opt, swap, insert) + augmentation	Dual-aspect collaborative transformer (DACT)	PPO + CL	Up to 200	TSPLIB, CVRPLIB
Neural-3-opt [163]	2021	ACML	TSP	{20, 50, 100}	Improvement (3-opt)	RNN + Att	A2C	Up to 200	TSPLIB
Chen et al. [164]	2022	EJOR	SDDPVD	3 vehicle 10 drones	Online decision	MLP	DQN	4 vehicle 10 drones	NA
P-MOCO [165]	2022	ICLR	MOTSP, MOVRP	{20, 50, 100}	Construction + augmentation	Transformer (POMO)	Modified POMO	NA	Benchmark [166]

TABLE 3 (Continued)

Method	Year	Venue	Problems	Problem size trained on	Heuristic type	Deep model	DRL algorithm	Generalisation (size)	Generalisation (benchmark)
MLDRL [167]	2022	TNNLS	MOTSP, MOVRPTW	{20, 50}	Construction + sampling	Transformer (AM)	REINFORCE + meta learning	Up to 200	TSPLIB
Zhang et al. [168]	2022	TITS	mTSPTWR	{50, 100, 150}	Construction + sampling	Transformer (AM) + GNN	REINFORCE	Up to 500	NA
EAS [13]	2022	ICLR	TSP, CVRP	{100}	Construction + active search	Transformer (POMO)	POMO + IL	{125, 150, 200}	NA
MAPDP [169]	2022	AAAI	PDP	{20, 50, 100}	Construction + sampling	Transformer (AM)	Cooperative A2C	NA	Real data
N2S [72]	2022	IJCAI	PDTSP, PDTSP-LIFO	{20, 50, 100}	Improvement (ruin-and-repair) + augmentation	Transformer (modified DACT)	PPO + CL	Up to 200	Benchmark [170, 171]

neural neighbourhood search (N2S) framework was proposed which was the first learning-based solver that outstrips the well-known LKH3 solver [176] on solving PDTSP and its variant with Last-In-Fist-Out (LIFO) constraint on the synthesised dataset.

Besides the precedence covariant in PDPs, the time window (TW) constraint is also critical in real-world logistics. The VRPTW was studied in Zhao et al. [149], in which the AM model is combined with a local search post-processing procedure to outstrip traditional heuristics such as the ACO. In Lin et al. [156], the model of Nazari et al [8] was adapted to solve the Electric VRPTW where the charging of electric vehicles are considered in the problem formulation. By proposing a novel Multi-Agent Attention Model, Zhang et al. [150] tackled the VRP with Soft Time Windows (VRPSTW) and outperformed OR-Tools with shorter computation time. In Zhang et al. [168], a more practical TSPTW variant called multiple-vehicle TSP with time window and rejections was studied, where customers can be rejected due to TW constraint.

Many other variants of routing problems are also studied via DRL-based methods. In refs [9, 148, 154], their models were also tested on Split Delivery Routing Problems (SDVRP), Orienteering Problem [177], Prize Collecting TSP (PCTSP) [178] and Stochastic PCTSP. The Split Delivery VRP (SDVRP) was studied in ref. [71] with a proposed Neural Large Neighbourhood Search (NLNS) framework which outperformed the strong hand-crafted solver SplitILS [179]. Li et al. [74] investigated the DRL-based solver for Heterogeneous CVRP, which takes into account vehicles of various capacity, using both min-sum and min-max objective settings. Kwon et al. [180] presented the first neural solver MatNet for solving the Asymmetric TSP based on the POMO [10] and achieved far superior performance than other neural approaches.

5.1.2 | Multi-objective routing problems

In real-world manufacturing, multiple (conflicting) objectives may exist in a single optimisation task, for example, minimising route length while minimising the number of vehicles. Since it is unlikely to optimise all the objectives with only one solution, the major purpose of multi-objective optimisation is to discover a group of *Pareto* optimal solutions. The Pareto optimal means that any solution in the Pareto set can dominate others in terms of some objectives but not all objectives. Thus, the Pareto optimal solutions are various trade-offs between the different objectives. From the literature, several representative hand-crafted solvers for multi-objective optimisation have been proposed such as the non-dominate sorting-based NSGA-II [181], the decomposition-based evolutionary algorithms MOEA/D [182], and the niching techniques based evolutionary algorithms [183].

Li et al. [152] were among the first attempts to solve multi-objective TSP (MOTSP) using DRL and showed better performance than the NSGA-II solver [181]. In their proposed DRL-MOA method, multiple Pointer Network [6]-based deep models are learnt together, each of which

TABLE 4 Other Machine Learning-assisted heuristic approaches for VRPs

Method	Year	Venue	Problems	Problem size	ML assisted heuristic type	Deep model
GCN-BS [136]	2019	Preprint	TSP	{20, 50, 100}	SL + beam search	GNN
Fu et al. [138]	2021	AAAI	TSP	Up to 10,000	SL + Monte Carlo tree search	GNN + Att
CVAE-Opt [141]	2021	ICLR	TSP, CVRP	Up to 150	UL (conditional variational autoencoder) + EA (differential evolution)	RNN
NeuroLKH [134]	2021	NeurIPS	TSP, CVRP, PDP, CVRPTW	Up to 1000	SL + UL + Lin-Kernighan-Helsgaun Heuristic	GNN
Learning-to-delegate [135]	2021	NeurIPS	CVRP, CVRPTW, VRPMPD	Up to 3000	SL + any VRP solver	Modified transformer
DPDP [133]	2022	CPAIOR	TSP, CVRP, TSPTW	{100}	SL + dynamic programming	GNN
GNN-GLS [139]	2022	ICLR	TSP	Up to 200	SL + guided local search	GNN

optimises a sub-problem that considers optimising the weighted-sum of multiple objectives with various different weight configurations. The obtained solutions by all models are then processed to find the Pareto solutions with a non-dominating sorting strategy similar to NSGA-II [181]. In Wu et al. [143], the authors further upgraded the Pointer Network in DRL-MOA into the more powerful AM model [9] and solved the multi-objective VRPTW variant for the first time, leading to the MODRL/D-AM method. Later on, MODRL/D-AM was further refined with additional techniques for better performance such as evolutionary learning [158] and meta learning [167]. However, the above methods all need to learn a set of multiple models and require additional traditional techniques to find Pareto solutions which may not be efficient. Moving against that, a recent approach P-MOCO [165] proposed to directly output the approximate Pareto solutions by learning one single preference-conditioned model to predict all the possible preferences. Experimental results on MOTSP and MOVRP showed that their method significantly outperformed other baselines in terms of solution quality and the search efficiency.

5.1.3 | Online/dynamic routing problems

The online VRPs contemplate a planning process in which information regarding the VRP instance is provided progressively, that is, in a real-time manner (see a survey [184]). This type of VRP is also connected to dynamic VRPs, which take into account information changes throughout the operational horizon (see a survey [185]). Such changes could be, for example, newly arrived or left orders, changes in service time, and time-dependent traffic conditions. Because the dispatch system in most online or dynamic VRP settings must respond to dynamic events in a timely manner, learning-based solver might be a promising option because it is often considerably quicker than hand-crafted heuristics. Meanwhile, the above online or dynamic events typically have underlying distributions or patterns, making data-driven and learning-based solvers, more ideal for tackling them.

Based on the deep model of graph embedding [142] and the Pointer Network [6], James et al. [144] was among the earliest works of applying DRL to solve dynamic VRPs where newly arrived pickup-delivery orders and dynamic traffic conditions are simulated based on real-world data. Later on, Zhang et al. [157] tackled the similar dynamic TSP and dynamic PDP (DPDP) by ameliorating the AM in Kool at al. [9] into an upgraded version in order to process dynamic environment changes. In Wu et al. [159], the AM [9] was adapted together with some RNN units to solve the time-dependent TSPTW variant. In Oren et al. [160], the deep Q-learning was explored together with an improved Monte Carlo Tree Search (MCTS) to solve online CVRP. However, the problem scales of the aforementioned works remain rather small. Li et al. [161] introduced Spatial-Temporal Aided Double Deep Graph Network (ST-DDGN), the first learning-based solver to tackle industry-scale DPDP with up to 600+ orders and 150+

vehicles. They exploited a ST-DDGN to estimate delivery demand and regulate the vehicle assignment for new orders. A hierarchical optimisation framework for DPDP was proposed in Ma et al. [162] to tackle large-scale DPDP with up to 1000 orders. In their methods, the upper-level agent could dynamically partition the DPDP into a series of sub-problems and the lower-level agent could solve the sub-problem using a neighbourhood search solver (could also be learning-based one). Aside from the above solvers that leveraged large and powerful deep models, there is another line of research works that used some basic neural networks such as the Multi-Layer Perceptron (MLP) [186] (that are much shallower than the ones such as the Pointer Network [6] and the AM [9]) to aid the decision making. Joe and Lau [153] studied the dynamic CVRP problem with newly arrived customers through a hybrid deep Q-learning and SA algorithm. Chen et al. [164] investigated a new online routing scenario in which vehicles and drones collaborate to serve dynamic customers and used RL to learn to allocate new customers to either drones or vehicles. In a recent work by Basso et al. [187], the safe RL (without neural networks) was explored to address the dynamic stochastic electric VRP. When compared to human solutions, these learning-based solvers for online/dynamic VRPs could generate much better results within shorter response time.

5.2 | Types of learnt neural solvers

As mentioned in Section 2, DRL-based works have mainly exploited two types of MDP formulations, resulting in two types of learnt neural solvers. In this section, we explain them in depth, followed by a short review of other learning-based solvers that do not employ DRL.

5.2.1 | Neural construction solvers

The first type is to learn a *construction* heuristic. Staring from an empty solution, it learns to add an customer node into the partial solution to construct the complete one. In general, neural construction solvers for routing problems consume extremely short inference time, which makes it more suitable for online/dynamic settings that need rapid response. However, for NP-hard routing problems, constructing solutions can easily fall into the local minimum, which renders the needs of combining other post-processing techniques, such as: (1) sampling (e.g. in refs [8, 9, 143, 144, 150, 152, 154–159, 167]) that repeats the stochastic construction process for multiple times, (2) beam search (e.g. in refs [148]) that performs an additional best-first search process, (3) AS (e.g. in refs [7, 13]) that performs a few extra gradient updates during inference for each specific instance, (4) data augmentation (e.g. in refs [10, 95, 165]) that samples multiple solutions by exploiting the equivalence and invariant transformations for routing problems, and (5) other hybrid construction and local search methods (e.g. in refs [12, 149]). Among them, we highlight the data augmentation technique, which was initially suggested in POMO [10] and is the most effective and efficient way for boosting the performance of neural construction solvers. In specific, the authors proposed that by flipping the node coordinates, a given TSP or CVRP instance in Euclidean space might be augmented into eight distinct ones with the same optimal solution. Though these instances are equivalent for the routing problem, the learned deep models may regard them as distinct, which would enable more diverse construction processes. We also highlight the EAS by Hottung et al. [13]. As also mentioned in Section 3.5, EAS performs AS for a particular instance during inference by only adjusting a small subset of parameters of the learnt deep model to achieve superior performance. And it could be generic to boost most of the above neural approaches. For example, by applying EAS to POMO on solving CVRP, EAS achieved a even better performance than the strong hand-crafted heuristic LKH3 [176], leading to the state-of-the-art performance on learning data-driven CVRP solvers [13].

5.2.2 | Neural improvement solvers

Aiming at directly learning the search process, the second type exploits the neural *improvement* heuristics that learn to perform N2S. Staring from a complete initial solution, it learns to iteratively transform the current solution into another one until a given time limit is achieved. Since this type of neural solvers learn to improve a poor random solution, they are in general slower than the neural construction heuristics (but can be still faster than conventional hand-crafted heuristics in most of the cases). These improvement solvers are motivated by the idea that if the available solving time is longer, they may find much higher quality solutions than neural construction solvers. As the first neural improvement heuristic, NeuRewriter [11] learns to pick local search operations and rewrite the local components of the current solution. It successfully outperformed the neural construction solver AM [9] and Nazari et al. [8] on solving CVRP. The L2I framework proposed in Lu et al. [73] explored the hybrid of learning framework and conventional methods. It used a learning-based neural network to select local search operators from a candidate pool containing several professional local search operators and became the first neural solver to outperform hand-crafted LKH3 solver [176] on solving CVRP instances in terms of the solution quality (though required longer computation time). By exploring SA heuristics as well as the idea of the large neighbourhood search (LNS) heuristics which considers applying destroy and repair operators to current solutions, the neural LNS (NLNS) was proposed in Hottung et al. [71] and also showed superior performance over AM [9]. Wu et al. [2] proposed a Transformer-based improvement solver for TSP and CVRP. By learning to perform the pair-wise operators (e.g. 2-opt, reinsert, or swap as illustrated in Figure 2), it has shown a promising result over NeuRewriter [11] and AM [9]. The method of Wu et al. [2] was then improved to dual-aspect

collaborative transformer (DACT) method by Ma et al. [70] based on a upgraded deep model and a CL strategy to enhance the RL. Though DACT displayed a significantly better inference and generalisation performance than Wu et al. [2], its computational efficiency is still not satisfactory when compared to the latest neural construction solvers (e.g. POMO [10]). In a more recent work, DACT was further modified to N2S [72] to solve the pickup and the delivery problems (PDPs). By further equipped with a designed data augmentation scheme similar to POMO [10], N2S significantly outperformed DACT, Heter-AM [155], and POMO on solving PDPs and became the first learning-based solver to outstrip the well-known hand-crafted LKH3 solver [176] on synthesised PDP datasets.

5.2.3 | Other ML-assisted solvers

Furthermore, there are studies that employed ML techniques other than DRL to learn neural solvers. In CVAE-Opt solver [141], the authors employed the unsupervised conditional variational autoencoder approach to learn a latent search space where each point in the space corresponds to a VRP solution. As a result, the optimisation of constrained routing problems was transformed into an unconstrained optimisation problem in the latent space, allowing high-quality solutions to be found via powerful unconstrained continuous optimisation solvers like differential evolution algorithm. Another interesting idea in the literature is the combination of SL with other conventional approaches. As discussed previously, SL can effectively train deep models only when the training data are available. Thus, SL is often prohibited for large-scale or complex VRPs. Recently, researchers discovered that models learnt through SL may also be utilised to effectively address challenging routing problems by integrating some post-processing techniques, such as beam search [136], MCTS [138], dynamic programming [133], and guided local search [139]. For example, in Joshi et al. [136], the authors made the first effort to learn a model using SL to predict a heat map that indicates the likelihood of edges belonging to the optimal TSP solutions. High-quality solutions are then searched by the beam search heuristics in such heat map. To address the limitation that SL only works on small-scale TSP, Fu et al. [138] suggested that we may construct heat maps for large-scale TSPs by (1) dividing large-scale TSPs into smaller sub-problems, (2) predicting heat-maps using models learnt on small-scale TSPs, and (3) merging these heat maps. Further combined with the MCTS, their method can be generalised to solving TSP with arbitrarily larger sizes. Meanwhile, SL was also used to train some decision-making models to assist a particular solver. In Xin et al. [134], SL was leveraged together with UL to create more efficient edge candidate sets that guides and improves the well-known LKH solver. In Li et al. [135], the classic SL technique regression was exploited to learn to select sub-problems of large-scale VRP instances and learn to delegate the optimisation of the sub-problem to existing VRP solvers.

5.3 | Future directions

5.3.1 | More powerful deep models

The representation capability of deep models ensures a good performance of learning-based neural solvers. Thereafter, we observe that many researchers have focussed on proposing more advanced models which could often lead to significant improvement on the performance of the learnt solver. For instance, on improving the performance of the neural construction solver AM [9], the researchers have explored different architecture designs such as multiple decoders [148], multiple relational attention [154], heterogeneous attention [155], hybrid of recurrent neural network [159], matrix encoding [180] etc. Regarding to the neural improvement solvers, we also see that the dual-aspect attention [70] and the Synthesis attention [72] lead to much better performance than the previous model in Wu et al. [2]. Despite the above successes, we argue that the development of deep models for routing problems are still immature. On the one hand, most of the existing techniques such as the Transformer is originally designed for the tasks in Natural Language Processing (NLP), which means that such design may not be optimal for VRP tasks. For example, as suggested by Ma et al. [70], the positional encoding of the original Transformer is designed for linear sequences (e.g. sentences), which makes it problematic to reflect the circularity and symmetry of a route sequence in VRPs. They thus proposed a novel cyclic positional encoding method which significantly enhanced the generalisation performance in terms of different problem sizes. In this regard, we believe that more investigations should be made to study deep models that are more powerful and suited for VRP tasks. On the other hand, although the current neural models are usually generic to solve a family of routing problems based on the same architecture, the training of the models for different variants need to be conducted independently from scratch. In other words, we do not have a powerful enough and universal deep routing model that can be trained only once and then quickly adapted to address different VRP variants. There are examples of this kind of universal models in the field of NLP. For example, the large, pre-trained language model BERT [188] could be trained to handle a variety of different NLP tasks by pre-training and then fine-tuning on the individual task. And we expect more research works in the future on studying such large, pre-trained deep models for routing problems.

5.3.2 | Enhancing scalability capability

The scalability often measures the efficiency of a learnt heuristic for large-scale instances. Although there are already some efforts on improving the scalability of the learnt solvers (e.g. [135, 138, 151]), the proposed methods are often limited to the TSP and CVRP, and at the same time are not generic enough to boost most of the solvers listed in Table 3. Meanwhile, the performance of DRL-based methods on large-scale optimisation are still not good enough as we can see most of the solvers

in Table 3 only focussed on problem sizes up to hundreds of nodes. This might because the computation complexity of the self-attention used in most of Transformer-based neural solvers would grows at least in quadratic as the size of the problem grows. More advanced techniques such as sparse attention [189] and kernel-based linear Transformers [190] can be considered in the future.

5.3.3 | Enhancing generalisation capability

More future studies should pay attention to improving the generalisation of the trained DRL models across different distributions. For current neural solvers, the majority of the training data (i.e. VRP instances containing the coordinates of the customer nodes) are generated at random using a certain distribution (mostly the Uniform distribution). Real-world VRP instances, however, could adhere to varied or even arbitrary unknown distributions. Unfortunately, ML-based solvers usually suffer from a serious generalisation performance, making these learnt models less effectiveness when used to solve these out-of-distribution instances. As pointed out by Joshi et al. [191], zero-shot generalisation of the current neural solvers *'requires rethinking the entire neural combinatorial optimisation pipeline, from network layers and learning paradigms to evaluation protocols'*. Recently, some attempts have already concentrated on improving the generalisation of the learnt models across other distributions. Among these attempts, most of them focussed on leveraging more diverse instances from different distributions to the training data, which is supposed to force the model to learn more robust and generalisable features. Such augmentation of training data may be achieved by adversarial robustness [192], group distributionally robust optimisation [193], and hardness-adaptive CL [194]. Different from the above methods that learn to augment the training dataset, Bi et al. [195] proposed to tackle the cross-distribution generalisation issue by knowledge distillation. They first train teacher models on several exemplar distributions, and then learn a light-weight yet generalist student model through the proposed AMDKD framework. Despite the success of the above attempts, we note that the upgraded generalisation ability of the neural solvers is still not that satisfactory, especially when compared to the traditional heuristics. We expect more efficient approaches in the future to boost the generalisation of these learning-based solvers.

5.3.4 | Solving more constrained routing problems

Existing works concentrate solely on searching in the feasible areas of the whole searching space, which is often accomplished by masking out invalid actions that produce infeasible solutions for both construction-based and improvement-based neural solvers. However, such design might not be effective, since these feasibility masks must be calculated for every action in the MDP, which is even more crucial for constrained problems. Unfortunately, there is currently no attempt that tries to improve the DRL framework for constrained problem. And there is also no analysis of the effect of such masking scheme on the final performance of learnt heuristics. Therefore, we expect future research to put efforts on evaluating the effectiveness of the current invalid action masking scheme and exploring more effective approaches to deal with constraints for learning-based solvers.

6 | CONCLUSION AND DISCUSSION

This paper summarised and analysed the state-of-the-art research in applying ML for optimisation problems in manufacturing from three aspects, that is, production, packing, and logistics. By citing the job shop scheduling problems, the BPPs, and the routing problems as concrete examples, we show that DRL is a ML paradigm favoured by the community. We comprehensively discussed the challenges each aspect faces. First and foremost, we hope this survey will help researchers from related areas overlook the problems being studied for the topic of learning for manufacturing and the progress so that more advances can be made in this field. In addition, pointing out the limitation of current research, we hope to highlight the challenges and provide an analysis of the potential reasons to motivate future works and push intelligent manufacturing to life.

AUTHOR CONTRIBUTIONS

Cong Zhang: Conceptualisation; Formal analysis; Methodology; Validation; Writing – original draft; Writing – review & editing. **Yaoxin Wu**: Conceptualisation; Formal analysis; Methodology; Validation; Writing – original draft; Writing – review & editing. **Yining Ma**: Conceptualisation; Formal analysis; Methodology; Validation; Writing – original draft; Writing – review & editing. **Wen Song**: Conceptualisation; Formal analysis; Funding acquisition; Methodology; Validation; Writing – original draft; Writing – review & editing. **Zhang Le**: Conceptualisation; Formal analysis; Methodology; Validation; Writing – original draft; Writing – review & editing. **Zhiguang Cao**: Conceptualisation; Formal analysis; Methodology; Validation; Writing – original draft; Writing – review & editing. **Jie Zhang**: Conceptualisation; Formal analysis; Methodology; Validation; Writing – original draft; Writing – review & editing.

ACKNOWLEDGEMENTS

This work was supported in part by the National Natural Science Foundation of China under Grant 62102228 and the Shandong Provincial Natural Science Foundation under Grant ZR2021QF063. This work was also supported by the A*STAR Cyber-Physical Production System (CPPS) – Towards Contextual and Intelligent Response Research Program, under the RIE2020 IAF-PP Grant A19C1a0018, and Model Factory@SIMTech. Finally, this work is supported in part by the A*Star Career Development Fund under Grant C222812027.

CONFLICT OF INTEREST

The authors confirm there is no conflict of interest to declare.

DATA AVAILABILITY STATEMENT

Data sharing is not applicable to this article as no new data were created or analysed in this study.

ORCID

Cong Zhang https://orcid.org/0000-0002-8434-1181
Yining Ma https://orcid.org/0000-0002-6639-8547

REFERENCES

1. Bengio, Y., Lodi, A., Prouvost, A.: Machine learning for combinatorial optimization: a methodological tour d'horizon. Eur. J. Oper. Res. 290(2), 405–421 (2021). https://doi.org/10.1016/j.ejor.2020.07.063
2. Wu, Y., et al.: Learning improvement heuristics for solving routing problems. IEEE Transact. Neural Networks Learn. Syst. 33(9), 5057–5069 (2021). https://doi.org/10.1109/tnnls.2021.3068828
3. Smith, K.A.: Neural networks for combinatorial optimization: a review of more than a decade of research. Inf. J. Comput. 11(1), 15–34 (1999). https://doi.org/10.1287/ijoc.11.1.15
4. Baluja, S., Davies, S.: Using optimal dependency-trees for combinational optimization. In: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 30–38 (1997)
5. Moll, R., et al.: Learning instance-independent value functions to enhance local search. In: Advances in Neural Information Processing Systems, pp. 1017–1023 (1999)
6. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Advances in Neural Information Processing Systems, pp. 2692–2700 (2015)
7. Bello, I., et al.: Neural combinatorial optimization with reinforcement learning. In: International Conference on Learning Representations (Workshop) (2017)
8. Nazari, M., et al.: Reinforcement learning for solving the vehicle routing problem. In: Advances in Neural Information Processing Systems, pp. 9861–9871 (2018)
9. Kool, W., van Hoof, H., Welling, M.: Attention, learn to solve routing problems. In: International Conference on Learning Representations (2019)
10. Kwon, Y.D., et al.: POMO: policy optimization with multiple optima for reinforcement learning. In: Larochelle, H., et al. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 21188–21198. Curran Associates, Inc., San Diego (2020)
11. Chen, X., Tian, Y.: Learning to perform local rewriting for combinatorial optimization. Adv. Neural Inf. Process. Syst. 32, 6281–6292 (2019)
12. Kim, M., Park, J., Kim, J.: Learning collaborative policies to solve NP-hard routing problems. Adv. Neural Inf. Process. Syst. 34, 10418–10430 (2021)
13. Hottung, A., Kwon, Y.D., Tierney, K.: Efficient active search for combinatorial optimization problems. In: International Conference on Learning Representations (2022)
14. Zhang, C., et al.: Learning to dispatch for job shop scheduling via deep reinforcement learning. In: Larochelle, H., et al. (eds.) Advances in Neural Information Processing Systems, vol. 33, pp. 1621–1632. Curran Associates, Inc., San Diego (2020)
15. Ni, F., et al.: A multi-graph attributed reinforcement learning based optimization algorithm for large-scale hybrid flow shop scheduling problem. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 3441–3451 (2021)
16. Wang, R., et al.: A bi-level framework for learning to solve combinatorial optimization on graphs. Adv. Neural Inf. Process. Syst. 34 (2021)
17. Yang, S., Xu, Z.: Intelligent scheduling and reconfiguration via deep reinforcement learning in smart manufacturing. Int. J. Prod. Res. 60(16), 1–18 (2021). https://doi.org/10.1080/00207543.2021.1943037
18. Zhao, H., Yu, Y., Xu, K.: Learning efficient online 3D bin packing on packing configuration trees. In: Proceedings of the 10th International Conference on Learning Representations (2021)
19. Hu, H., et al.: Solving a new 3D bin packing problem with deep reinforcement learning method. arXiv preprint arXiv:170805930 (2017)
20. Jiang, Y., Cao, Z., Zhang, J.: Learning to solve 3-D bin packing problem via deep reinforcement learning and constraint programming. IEEE Trans. Cybern., 1–12 (2021). https://doi.org/10.1109/tcyb.2021.3121542
21. Zhao, H., et al.: Online 3D bin packing with constrained deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, pp. 741–749 (2021)
22. Gasse, M., et al.: Exact combinatorial optimization with graph convolutional neural networks. In: Advances in Neural Information Processing Systems (2019)
23. Wu, Y., et al.: Learning large neighborhood search policy for integer programming. Adv. Neural Inf. Process. Syst. 34, 6278–6289 (2021)
24. Gupta, P., et al.: Hybrid models for learning to branch. Adv. Neural Inf. Process. Syst. 33, 18087–18097 (2020)
25. Khalil, E.B., Morris, C., Lodi, A.: MIP-GNN: a data-driven framework for guiding combinatorial solvers. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence (2022)
26. Song, J., et al.: A general large neighborhood search framework for solving integer linear programs. In: Advances in Neural Information Processing Systems (2020)
27. Wu, Y., et al.: Learning scenario representation for solving two-stage stochastic integer programs. In: International Conference on Learning Representations (2021)
28. Wu, Y., et al.: Graph learning assisted multi-objective integer programming. In: Advances in Neural Information Processing Systems (2022)
29. Cappart, Q., et al.: Combining reinforcement learning and constraint programming for combinatorial optimization. Proc. AAAI Conf. Artif. Intell. 35(5), 3677–3687 (2021). https://doi.org/10.1609/aaai.v35i5.16484
30. Chalumeau, F., et al.: Seapearl: a constraint programming solver guided by reinforcement learning. In: Stuckey, P.J. (eds.) International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research, pp. 392–409. Springer, Cham (2021)
31. Song, W., et al.: Learning variable ordering heuristics for solving constraint satisfaction problems. Eng. Appl. Artif. Intell. 109, 104603 (2022). https://doi.org/10.1016/j.engappai.2021.104603
32. Li, S., et al.: Machine learning-based scheduling: a bibliometric perspective. IET Collab. Intell. Manuf. 3(2), 131–146 (2021). https://doi.org/10.1049/cim2.12004
33. Bogyrbayeva, A., et al.: Learning to solve vehicle routing problems: a survey. arXiv preprint arXiv:220502453 (2022)
34. Zhang, J., et al.: A survey for solving mixed integer programming via machine learning. arXiv preprint arXiv:220302878 (2022)
35. Ali, S., et al.: On-line three-dimensional packing problems: a review of off-line and on-line solution approaches. Comput. Ind. Eng. 168, 108122 (2022). https://doi.org/10.1016/j.cie.2022.108122
36. Kotary, J., et al.: End-to-end constrained optimization learning: a survey. arXiv preprint arXiv:210316378 (2021)
37. Garmendia, A.I., Ceberio, J., Mendiburu, A.: Neural combinatorial optimization: a new player in the field. arXiv preprint arXiv:220501356 (2022)
38. Vesselinova, N., et al.: Learning combinatorial optimization on graphs: a survey with applications to networking. IEEE Access 8, 120388–120416 (2020). https://doi.org/10.1109/access.2020.3004964
39. Cappart, Q., et al.: Combinatorial optimization and reasoning with graph neural networks. arXiv preprint arXiv:210209544 (2021)
40. Peng, Y., Choi, B., Xu, J.: Graph learning for combinatorial optimization: a survey of state-of-the-art. Data Sci. Eng. 6(2), 119–141 (2021). https://doi.org/10.1007/s41019-021-00155-3
41. Mazyavkina, N., et al.: Reinforcement learning for combinatorial optimization: a survey. Comput. Oper. Res. 134, 105400 (2021). https://doi.org/10.1016/j.cor.2021.105400
42. Wang, Q., Tang, C.: Deep reinforcement learning for transportation network combinatorial optimization: a survey. Knowl. Base Syst. 233, 107526 (2021). https://doi.org/10.1016/j.knosys.2021.107526

43. Yang, Y., Whinston, A.: A survey on reinforcement learning for combinatorial optimization. arXiv preprint arXiv:200812248 (2020)
44. Panzer, M., Bender, B.: Deep reinforcement learning in production systems: a systematic literature review. Int. J. Prod. Res. 60(13), 4316–4341 (2022). https://doi.org/10.1080/00207543.2021.1973138
45. Wang, L., Pan, Z., Wang, J.: A review of reinforcement learning based intelligent optimization for manufacturing scheduling. Complex Syst. Model. Simulat. 1(4), 257–270 (2021). https://doi.org/10.23919/csms.2021.0027
46. Kayhan, B.M., Yildiz, G.: Reinforcement learning applications to machine scheduling problems: a comprehensive literature review. J. Intell. Manuf., 1–25 (2021). https://doi.org/10.1007/s10845-021-01847-3
47. Farazi, N.P., et al.: Deep reinforcement learning in transportation research: a review. Transp. Res. Interdiscip. Perspect. 11, 100425 (2021). https://doi.org/10.1016/j.trip.2021.100425
48. Aydemir, E., Karagul, K.: Solving a periodic capacitated vehicle routing problem using simulated annealing algorithm for a manufacturing company. Braz. J. Oper. Prod. Manag. 17(1), 1–13 (2020). https://doi.org/10.14488/bjopm.2020.011
49. Duan, J., et al.: A data-driven column generation algorithm for bin packing problem in manufacturing industry. arXiv preprint arXiv:220212466 (2022)
50. Sharp, M., Ak, R., Hedberg, T., Jr.: A survey of the advancing use and development of machine learning in smart manufacturing. J. Manuf. Syst. 48, 170–179 (2018). https://doi.org/10.1016/j.jmsy.2018.02.004
51. Weichert, D., et al.: A review of machine learning for the optimization of production processes. Int. J. Adv. Manuf. Technol. 104(5), 1889–1902 (2019). https://doi.org/10.1007/s00170-019-03988-5
52. Dogan, A., Birant, D.: Machine learning and data mining in manufacturing. Expert Syst. Appl. 166, 114060 (2021). https://doi.org/10.1016/j.eswa.2020.114060
53. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press, Cambridge (2018)
54. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature 521(7553), 436–444 (2015). https://doi.org/10.1038/nature14539
55. Williams, R.J.: Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach. Learn. 8(3), 229–256 (1992). https://doi.org/10.1007/bf00992696
56. Mnih, V., et al.: Asynchronous methods for deep reinforcement learning. In: Balcan, M.F., Weinberger, K.Q. (eds.) International Conference on Machine Learning, pp. 1928–1937. PMLR, New York (2016)
57. Schulman, J., et al.: Proximal policy optimization algorithms. arXiv preprint arXiv:170706347 (2017)
58. Mnih, V., et al.: Human-level control through deep reinforcement learning. Nature 518(7540), 529–533 (2015). https://doi.org/10.1038/nature14236
59. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 30. (2016)
60. Wang, Z., et al.: Dueling network architectures for deep reinforcement learning. In: Balcan, M.F., Weinberger, K.Q. (eds.) International Conference on Machine Learning, pp. 1995–2003. PMLR, New York (2016)
61. François. Lavet, V., et al.: An introduction to deep reinforcement learning. Found. Trends Mach. Learn. 11(3-4), 219–354 (2018). https://doi.org/10.1561/2200000071
62. Kan, A.R.: Machine Scheduling Problems: Classification, Complexity and Computations. Springer Science & Business Media, New York (2012)
63. Błażewicz, J., Pesch, E., Sterna, M.: The disjunctive graph machine representation of the job shop scheduling problem. Eur. J. Oper. Res. 127(2), 317–331 (2000). https://doi.org/10.1016/s0377-2217(99)00486-5
64. Park, J., Bakhtiyar, S., Park, J.: Schedulenet: learn to solve multi-agent scheduling problems with reinforcement learning. arXiv preprint arXiv:210603051 (2021)
65. Verma, R., et al.: A generalized reinforcement learning algorithm for online 3D bin-packing. arXiv preprint arXiv:200700463 (2020)
66. Li, D., et al.: One model packs thousands of items with recurrent conditional query learning. Knowl. Base Syst. 235, 107683 (2022). https://doi.org/10.1016/j.knosys.2021.107683
67. Hu, R., et al.: Tap-net: transport-and-pack using reinforcement learning. ACM Trans. Graph. 39(6), 1–15 (2020). https://doi.org/10.1145/3414685.3417796
68. Yang, T., et al.: A flexible reinforced bin packing framework with automatic slack selection. Math. Probl. Eng. 2021, 1–15 (2021). https://doi.org/10.1155/2021/6653586
69. Hansen, P., Mladenović, N., Pérez, J.A.M.: Variable neighbourhood search: methods and applications. Ann. Oper. Res. 175(1), 367–407 (2010). https://doi.org/10.1007/s10479-009-0657-6
70. Ma, Y., et al.: Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In: Ranzato, M., et al. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 11096–11107. Curran Associates, Inc., San Diego (2021)
71. Hottung, A., Tierney, K.: Neural large neighborhood search for the capacitated vehicle routing problem. In: European Conference on Artificial Intelligence (2020)
72. Ma, Y., et al.: Efficient neural neighborhood search for pickup and delivery problems. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (IJCAI). International Joint Conferences on Artificial Intelligence Organization (2022)
73. Lu, H., Zhang, X., Yang, S.: A learning-based iterative method for solving vehicle routing problems. In: International Conference on Learning Representations (2019)
74. Li, J., et al.: Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. IEEE Trans. Cybern. 52(12), 13572–13585 (2021). https://doi.org/10.1109/tcyb.2021.3111082
75. Ulmer, M.W., et al.: On modeling stochastic dynamic vehicle routing problems. EURO J. Transport. Logist. 9(2), 100008 (2020). https://doi.org/10.1016/j.ejtl.2020.100008
76. Manne, A.S.: On the job-shop scheduling problem. Oper. Res. 8(2), 219–223 (1960). https://doi.org/10.1287/opre.8.2.219
77. Balas, E.: An additive algorithm for solving linear programs with zero-one variables. Oper. Res. 13(4), 517–546 (1965). https://doi.org/10.1287/opre.13.4.517
78. Osman, I.H., Kelly, J.P.: Meta-heuristics theory and applications. J. Oper. Res. Soc. 48(6), 657 (1997) https://doi.org/10.1038/sj.jors.2600781
79. Tamura, T., Ohsaki, M., Takagi, J.: Machine learning for combinatorial optimization of brace placement of steel frames. Japan Architect. Rev. 1(4), 419–430 (2018). https://doi.org/10.1002/2475-8876.12059
80. Rinciog, A., Meyer, A.: Towards standardising reinforcement learning approaches for production scheduling problems. Procedia CIRP 107, 1112–1119 (2022). https://doi.org/10.1016/j.procir.2022.05.117
81. Sels, V., Gheysen, N., Vanhoucke, M.: A comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions. Int. J. Prod. Res. 50(15), 4255–4270 (2012). https://doi.org/10.1080/00207543.2011.611539
82. Park, J., et al.: Learning to schedule job-shop problems: representation and policy learning using graph neural network and reinforcement learning. Int. J. Prod. Res. 59(11), 3360–3377 (2021). https://doi.org/10.1080/00207543.2020.1870013
83. Tassel, P., Gebser, M., Schekotihin, K.: A reinforcement learning environment for job-shop scheduling. arXiv preprint arXiv:210403760 (2021)
84. Brockman, G., et al.: Openai gym. arXiv preprint arXiv:160601540 (2016)
85. Jain, A.S., Meeran, S.: Deterministic job-shop scheduling: past, present and future. Eur. J. Oper. Res. 113(2), 390–434 (1999). https://doi.org/10.1016/s0377-2217(98)00113-1
86. Mei, Y., Zhang, M.: A comprehensive analysis on reusability of GP-evolved job shop dispatching rules. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 3590–3597. IEEE (2016)
87. Zhang, F., et al.: Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. IEEE Trans. Evol. Comput. 25(4), 651–665 (2021). https://doi.org/10.1109/tevc.2021.3065707

88. Lin, C.C., et al.: Smart manufacturing scheduling with edge computing using multiclass deep Q network. IEEE Trans. Ind. Inf. 15(7), 4276–4284 (2019). https://doi.org/10.1109/tii.2019.2908210
89. Chaudhry, I.A., Khan, A.A.: A research survey: review of flexible job shop scheduling techniques. Int. Trans. Oper. Res. 23(3), 551–591 (2016). https://doi.org/10.1111/itor.12199
90. Müller, D., et al.: An algorithm selection approach for the flexible job shop scheduling problem: choosing constraint programming solvers through machine learning. Eur. J. Oper. Res. 302(3), 874–891 (2022). https://doi.org/10.1016/j.ejor.2022.01.034
91. Song, W., et al.: Flexible job shop scheduling via graph neural network and deep reinforcement learning. IEEE Trans. Ind. Inf. 19(2), 1600–1610 (2022). https://doi.org/10.1109/tii.2022.3189725
92. Jun, S., Lee, S., Chun, H.: Learning dispatching rules using random forest in flexible job shop scheduling problems. Int. J. Prod. Res. 57(10), 3290–3310 (2019). https://doi.org/10.1080/00207543.2019.1581954
93. Iklassov, Z., et al.: Learning to generalize dispatching rules on the job shop scheduling. arXiv preprint arXiv:220604423 (2022)
94. Zeng, Y., et al.: Hybrid intelligence for dynamic job-shop scheduling with deep reinforcement learning and attention mechanism. arXiv preprint arXiv:220100548 (2022)
95. Kwon, Y.D., et al.: Matrix encoding networks for neural combinatorial optimization. In: Ranzato, M., et al. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 5138–5149. Curran Associates, Inc., San Diego (2021)
96. Pan, Z., et al.: Deep reinforcement learning based optimization algorithm for permutation flow-shop scheduling. IEEE Trans. Emerg. Top. Comput. Intell., 1–12 (2021). https://doi.org/10.1109/tetci.2021.3098354
97. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al.: Attention is all you need. In: Guyon, I., et al., (eds.) Advances in Neural Information Processing Systems, vol. 30, pp. 5998–6008. Curran Associates, Inc., San Diego (2017)
98. Vaswani, A., et al.: Attention is all you need. Adv. Neural Inf. Process. Syst. 30, 6000–6010 (2017)
99. Józefowska, J., et al.: Fast truck-packing of 3D boxes. Eng. Manag. Prod. Serv. 10(2), 29–40 (2018). https://doi.org/10.2478/emj-2018-0009
100. Liu, W., Deng, T., Li, J.: Product packing and stacking under uncertainty: a robust approach. Eur. J. Oper. Res. 277(3), 903–917 (2019). https://doi.org/10.1016/j.ejor.2019.03.040
101. Lazarev, D., Kuzyurin, N.N.: On online algorithms for bin, strip, and box packing, and their worst-case and average-case analysis. Program. Comput. Software 45(8), 448–457 (2019). https://doi.org/10.1134/s0361768819080036
102. Laterre, A., et al.: Ranked reward: enabling self-play reinforcement learning for combinatorial optimization. arXiv preprint arXiv:180701672 (2018)
103. Duan, L., et al.: A multi-task selected learning approach for solving 3D flexible bin packing problem. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, pp. 1386–1394 (2019)
104. Zhang, J., Zi, B., Ge, X.: Attend2pack: bin packing through deep reinforcement learning with attention. arXiv preprint arXiv:210704333 (2021)
105. Li, D., et al.: Solving packing problems by conditional query learning (2019)
106. Kundu, O., Dutta, S., Kumar, S.: Deep-pack: a vision-based 2D online bin packing algorithm with deep reinforcement learning. In: 2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), pp. 1–7. IEEE (2019)
107. Yang, Z., et al.: Packerbot: variable-sized product packing with heuristic deep reinforcement learning. In: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5002–5008. IEEE (2021)
108. Zhu, Q., et al.: Learning to pack: a data-driven tree search algorithm for large-scale 3D bin packing problem. In: Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 4393–4402 (2021)
109. Zhao, H., et al.: Learning practically feasible policies for online 3D bin packing. Sci. China Inf. Sci. 65(1), 1–17 (2022). https://doi.org/10.1007/s11432-021-3348-6
110. Burke, E., et al.: Hyper-heuristics: an emerging direction in modern search technology. In: Glover, F., Kochenberger, G.A. (eds.) Handbook of Metaheuristics, pp. 457–474. Springer, Boston (2003)
111. Mao, F., et al.: Small boxes big data: a deep learning approach to optimize variable sized bin packing. In: 2017 IEEE Third International Conference on Big Data Computing Service and Applications (BigDataService), pp. 80–89. IEEE (2017)
112. López. Camacho, E., et al.: A unified hyper-heuristic framework for solving bin packing problems. Expert Syst. Appl. 41(15), 6876–6889 (2014). https://doi.org/10.1016/j.eswa.2014.04.043
113. Duhart, B., et al.: An experimental study on ant colony optimization hyper-heuristics for solving the knapsack problem. In: Mexican Conference on Pattern Recognition, pp. 62–71. Springer (2018)
114. Silva-Gálvez, A., et al.: Discovering action regions for solving the bin packing problem through hyper-heuristics. In: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 822–828. IEEE (2020)
115. Burke, E.K., et al.: Automating the packing heuristic design process with genetic programming. Evol. Comput. 20(1), 63–89 (2012). https://doi.org/10.1162/evco_a_00044
116. Drake, J.H., et al.: A genetic programming hyper-heuristic for the multidimensional knapsack problem. Kybernetes 43(9/10), 1500–1511 (2014). https://doi.org/10.1108/k-09-2013-0201
117. Burke, E.K., et al.: Exploring hyper-heuristic methodologies with genetic programming. In: Computational Intelligence, pp. 177–201. Springer (2009)
118. Asta, S., Özcan, E., Parkes, A.J.: Champ: creating heuristics via many parameters for online bin packing. Expert Syst. Appl. 63, 208–221 (2016). https://doi.org/10.1016/j.eswa.2016.07.005
119. Burke, E.K., et al.: The scalability of evolved on line bin packing heuristics. In: 2007 IEEE Congress on Evolutionary Computation, pp. 2530–2537. IEEE (2007)
120. Dulac-Arnold, G., et al.: Deep reinforcement learning in large discrete action spaces. arXiv preprint arXiv:151207679 (2015)
121. Tavakoli, A., Pardo, F., Kormushev, P.: Action branching architectures for deep reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32 (2018)
122. Chandak, Y., et al.: Learning action representations for reinforcement learning. In: Proceedings of the 36th International Conference on Machine Learning (ICML), pp. 941–950 (2019)
123. Pazis, J., Parr, R.: Generalized value functions for large action sets. In: Proceedings of the 28th International Conference on Machine Learning (ICML), pp. 1185–1192 (2011)
124. Cattaruzza, D., et al.: Vehicle routing problems for city logistics. EURO J. Transport. Logist. 6(1), 51–79 (2017). https://doi.org/10.1007/s13676-014-0074-0
125. Lenstra, J.K., Kan, A.R.: Complexity of vehicle routing and scheduling problems. Networks 11(2), 221–227 (1981). https://doi.org/10.1002/net.3230110211
126. Wei, L., et al.: A simulated annealing algorithm for the capacitated vehicle routing problem with two-dimensional loading constraints. Eur. J. Oper. Res. 265(3), 843–859 (2018). https://doi.org/10.1016/j.ejor.2017.08.035
127. Hemmelmayr, V.C., Doerner, K.F., Hartl, R.F.: A variable neighborhood search heuristic for periodic routing problems. Eur. J. Oper. Res. 195(3), 791–802 (2009). https://doi.org/10.1016/j.ejor.2007.08.048
128. Veenstra, M., et al.: The pickup and delivery traveling salesman problem with handling costs. Eur. J. Oper. Res. 257(1), 118–132 (2017). https://doi.org/10.1016/j.ejor.2016.07.009
129. Prins, C.: A simple and effective evolutionary algorithm for the vehicle routing problem. Comput. Oper. Res. 31(12), 1985–2002 (2004). https://doi.org/10.1016/s0305-0548(03)00158-8

130. Vidal, T., et al.: A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Oper. Res. 60(3), 611–624 (2012). https://doi.org/10.1287/opre.1120.1048
131. Bigliardi, B., Casella, G., Bottani, E.: Industry 4.0 in the logistics field: a bibliometric analysis. IET Collab. Intell. Manuf. 3(1), 4–12 (2021). https://doi.org/10.1049/cim2.12015
132. Woschank, M., Rauch, E., Zsifkovits, H.: A review of further directions for artificial intelligence, machine learning, and deep learning in smart logistics. Sustainability 12(9), 3760 (2020). https://doi.org/10.3390/su12093760
133. Kool, W., et al.: Deep policy dynamic programming for vehicle routing problems. In: International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research, pp. 190–213. Springer (2022)
134. Xin, L., et al.: Neurolkh: combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. In: Ranzato, M., et al. (eds.) Advances in Neural Information Processing Systems, vol. 34, pp. 7472–7483. Curran Associates, Inc. (2021)
135. Li, S., Yan, Z., Wu, C.: Learning to delegate for large-scale vehicle routing. Adv. Neural Inf. Process. Syst. 34, 26198–26211 (2021)
136. Joshi, C.K., Laurent, T., Bresson, X.: An efficient graph convolutional network technique for the travelling salesman problem (2019)
137. Prates, M., et al.: Learning to solve NP-complete problems: a graph neural network for decision TSP. In: AAAI Conference on Artificial Intelligence (2019)
138. Fu, Z.H., Qiu, K.B., Zha, H.: Generalize a small pre-trained model to arbitrarily large TSP instances. In: AAAI Conference on Artificial Intelligence (2021)
139. Hudson, B., et al.: Graph neural network guided local search for the traveling salesperson problem. In: International Conference on Learning Representations (2022)
140. Karalias, N., Loukas, A.: Erdos goes neural: an unsupervised learning framework for combinatorial optimization on graphs. Adv. Neural Inf. Process. Syst. 33, 6659–6672 (2020)
141. Hottung, A., Bhandari, B., Tierney, K.: Learning a latent search space for routing problems using variational autoencoders. In: International Conference on Learning Representations (2021)
142. Dai, H., et al.: Learning combinatorial optimization algorithms over graphs. In: Advances in Neural Information Processing Systems, pp. 6351–6361 (2017)
143. Wu, H., Wang, J., Zhang, Z.: MODRL/D-AM: multiobjective deep reinforcement learning algorithm using decomposition and attention model for multiobjective optimization. In: International Symposium on Intelligence Computation and Applications, pp. 575–589. Springer (2019)
144. James, J., Yu, W., Gu, J.: Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Trans. Intell. Transport. Syst. 20(10), 3806–3817 (2019). https://doi.org/10.1109/tits.2019.2909109
145. Belenguer, J.M., Martinez, M., Mota, E.: A lower bound for the split delivery vehicle routing problem. Oper. Res. 48(5), 801–810 (2000). https://doi.org/10.1287/opre.48.5.801.12407
146. da Costa, P.R.D.O., et al.: Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In: Asian Conference on Machine Learning, pp. 465–480 (2020)
147. Xin, L., et al.: Step-wise deep learning models for solving routing problems. IEEE Trans. Ind. Inf. 17(7), 4861–4871 (2020). https://doi.org/10.1109/tii.2020.3031409
148. Xin, L., et al.: Multi-decoder attention model with embedding glimpse for solving vehicle routing problems. In: AAAI Conference on Artificial Intelligence (2020)
149. Zhao, J., et al.: A hybrid of deep reinforcement learning and local search for the vehicle routing problems. IEEE Trans. Intell. Transport. Syst. 22(11), 7208–7218 (2020). https://doi.org/10.1109/tits.2020.3003163
150. Zhang, K., et al.: Multi-vehicle routing problems with soft time windows: a multi-agent reinforcement learning approach. Transport. Res. C Emerg. Technol. 121, 102861 (2020). https://doi.org/10.1016/j.trc.2020.102861
151. Drori, I., et al.: Learning to solve combinatorial optimization problems on real-world graphs in linear time. In: 2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA), pp. 19–24. IEEE (2020)
152. Li, K., Zhang, T., Wang, R.: Deep reinforcement learning for multiobjective optimization. IEEE Trans. Cybern. 51(6), 3103–3114 (2020). https://doi.org/10.1109/tcyb.2020.2977661
153. Joe, W., Lau, H.C.: Deep reinforcement learning approach to solve dynamic vehicle routing problem with stochastic customers. Proc. Int. Conf. Autom. Plan. Sched. 30, 394–402 (2020). https://doi.org/10.1609/icaps.v30i1.6685
154. Xu, Y., et al.: Reinforcement learning with multiple relational attention for solving vehicle routing problems. IEEE Trans. Cybern. 52(10), 11107–11120 (2021). https://doi.org/10.1109/tcyb.2021.3089179
155. Li, J., et al.: Heterogeneous attentions for solving pickup and delivery problem via deep reinforcement learning. IEEE Trans. Intell. Transport. Syst. 23(3), 2306–2315 (2021). https://doi.org/10.1109/tits.2021.3056120
156. Lin, B., Ghaddar, B., Nathwani, J.: Deep reinforcement learning for the electric vehicle routing problem with time windows. IEEE Trans. Intell. Transport. Syst. 23(8), 11528–11538 (2021). https://doi.org/10.1109/tits.2021.3105232
157. Zhang, Z., et al.: Solving dynamic traveling salesman problems with deep reinforcement learning. IEEE Transact. Neural Networks Learn. Syst., 1–14 (2021). https://doi.org/10.1109/tnnls.2021.3105905
158. Zhang, Y., et al.: MODRL/D-EL: multiobjective deep reinforcement learning with evolutionary learning for multiobjective optimization. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–8. IEEE (2021)
159. Wu, G., et al.: Solving time-dependent traveling salesman problem with time windows with deep reinforcement learning. In: 2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 558–563. IEEE (2021)
160. Oren, J., et al.: SOLO: search online, learn offline for combinatorial optimization problems. Proc. Int. Symp. Comb. Search 12(1), 97–105 (2021). https://doi.org/10.1609/socs.v12i1.18556
161. Li, X., et al.: Learning to optimize industry-scale dynamic pickup and delivery problems. In: 2021 IEEE 37th International Conference on Data Engineering (ICDE), pp. 2511–2522. IEEE (2021)
162. Ma, Y., et al.: A hierarchical reinforcement learning based optimization framework for large-scale dynamic pickup and delivery problems. Adv. Neural Inf. Process. Syst. 34, 23609–23620 (2021)
163. Sui, J., et al.: Learning 3-opt heuristics for traveling salesman problem via deep reinforcement learning. In: Asian Conference on Machine Learning, pp. 1301–1316. PMLR (2021)
164. Chen, X., Ulmer, M.W., Thomas, B.W.: Deep Q-learning for same-day delivery with vehicles and drones. Eur. J. Oper. Res. 298(3), 939–952 (2022). https://doi.org/10.1016/j.ejor.2021.06.021
165. Lin, X., Yang, Z., Zhang, Q.: Pareto set learning for neural multiobjective combinatorial optimization. In: International Conference on Learning Representations (2022)
166. Fonseca, C.M., Paquete, L., LópezIbánez, M.: An improved dimension-sweep algorithm for the hypervolume indicator. In: 2006 IEEE International Conference on Evolutionary Computation, pp. 1157–1163. IEEE (2006)
167. Zhang, Z., et al.: Meta-learning-based deep reinforcement learning for multiobjective optimization problems. IEEE Transact. Neural Networks Learn. Syst., 1–14 (2022). https://doi.org/10.1109/tnnls.2022.3148435
168. Zhang, R., et al.: Learning to solve multiple-TSP with time window and rejections via deep reinforcement learning. IEEE Trans. Intell. Transport. Syst., 1–12 (2022). https://doi.org/10.1109/tits.2022.3207011
169. Zong, Z., et al.: MAPDP: cooperative multi-agent reinforcement learning to solve pickup and delivery problems. Proc. AAAI Conf. Artif. Intell. 36(9), 9980–9988 (2022). https://doi.org/10.1609/aaai.v36i9.21236

170. Carrabs, F., Cordeau, J.F., Laporte, G.: Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. Inf. J. Comput. 19(4), 618–632 (2007). https://doi.org/10.1287/ijoc.1060.0202
171. Renaud, J., Boctor, F.F., Laporte, G.: Perturbation heuristics for the pickup and delivery traveling salesman problem. Comput. Oper. Res. 29(9), 1129–1141 (2002). https://doi.org/10.1016/s0305-0548(00)00109-x
172. Furian, N., et al.: A machine learning-based branch and price algorithm for a sampled vehicle routing problem. OR Spectr. 43(3), 1–40 (2021). https://doi.org/10.1007/s00291-020-00615-8
173. Lodi, A., Zarpellon, G.: On learning and branching: a survey. Top 25(2), 207–236 (2017). https://doi.org/10.1007/s11750-017-0451-6
174. Perron, L., Furnon, V.: OR-Tools (version 7.2). https://developers.google.com/optimization/ (2019). Accessed 15 Jan 2021
175. Dantzig, G.B., Ramser, J.H.: The truck dispatching problem. Manag. Sci. 6(1), 80–91 (1959). https://doi.org/10.1287/mnsc.6.1.80
176. Helsgaun, K.: An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems. Roskilde University, Roskilde (2017)
177. Golden, B.L., Levy, L., Vohra, R.: The orienteering problem. Nav. Res. Logist. 34(3), 307–318 (1987). https://doi.org/10.1002/1520-6750(198706)34:3<307::aid-nav3220340302>3.0.co;2-d
178. Balas, E.: The prize collecting traveling salesman problem. Networks 19(6), 621–636 (1989). https://doi.org/10.1002/net.3230190602
179. Silva, M.M., Subramanian, A., Ochi, L.S.: An iterated local search heuristic for the split delivery vehicle routing problem. Comput. Oper. Res. 53, 234–249 (2015). https://doi.org/10.1016/j.cor.2014.08.005
180. Kwon, Y.D., et al.: Matrix encoding networks for neural combinatorial optimization. Adv. Neural Inf. Process. Syst. 34, 5138–5149 (2021)
181. Deb, K., et al.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. 6(2), 182–197 (2002). https://doi.org/10.1109/4235.996017
182. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans. Evol. Comput. 11(6), 712–731 (2007). https://doi.org/10.1109/tevc.2007.892759
183. Liang, J.J., Yue, C., Qu, B.Y.: Multimodal multi-objective optimization: a preliminary study. In: 2016 IEEE Congress on Evolutionary Computation (CEC), pp. 2454–2461. IEEE (2016)
184. Jaillet, P., Wagner, M.R.: Online vehicle routing problems: a survey. In: The Vehicle Routing Problem: Latest Advances and New Challenges, pp. 221–237. Springer (2008)
185. Rios, B.H.O., et al.: Recent dynamic vehicle routing problems: a survey. Comput. Ind. Eng. 160, 107604 (2021). https://doi.org/10.1016/j.cie.2021.107604
186. Haykin, S.: Neural Networks: A Comprehensive Foundation. Prentice Hall PTR, Hoboken (1994)
187. Basso, R., et al.: Dynamic stochastic electric vehicle routing with safe reinforcement learning. Transport. Res. E Logist. Transport. Rev. 157, 102496 (2022). https://doi.org/10.1016/j.tre.2021.102496
188. Devlin, J., et al.: Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805 (2018)
189. Zaheer, M., et al.: Big bird: transformers for longer sequences. In: Advances in Neural Information Processing Systems (2020)
190. Choromanski, K., et al.: Rethinking attention with performers. arXiv preprint arXiv:200914794 (2020)
191. Joshi, C.K., et al.: Learning TSP requires rethinking generalization. arXiv preprint arXiv:200607054 (2020)
192. Geisler, S., et al.: Generalization of neural combinatorial solvers through the lens of adversarial robustness. In: International Conference on Learning Representations (2022)
193. Jiang, Y., et al.: Learning to solve routing problems via distributionally robust optimization. In: AAAI Conference on Artificial Intelligence (2022)
194. Zhang, Z., et al.: Learning to solve travelling salesman problem with hardness-adaptive curriculum. In: AAAI Conference on Artificial Intelligence (2022)
195. Bi, J., et al.: Learning generalizable models for vehicle routing problems via knowledge distillation. In: Advances in Neural Information Processing Systems (2022)

How to cite this article: Zhang, C., et al.: A review on learning to solve combinatorial optimisation problems in manufacturing. IET Collab. Intell. Manuf. e12072 (2023). https://doi.org/10.1049/cim2.12072