SCIENCE CHINA
Mathematics

• REVIEWS •

October 2025 Vol. 68 No. 10: 2519–2537
https://doi.org/10.1007/s11425-023-2364-2

Combinatorial optimization: From deep learning to large language models

Peng Tao[1] & Luonan Chen[1,2,3,4,*,†]

[1]*Key Laboratory of Systems Health Science of Zhejiang Province, School of Life Science, Hangzhou Institute for Advanced Study, University of Chinese Academy of Sciences, Hangzhou* 310024*, China;*
[2]*Key Laboratory of Systems Biology, Shanghai Institute of Biochemistry and Cell Biology, Center for Excellence in Molecular Cell Science, Chinese Academy of Sciences, Shanghai* 200031*, China;*
[3]*Guangdong Institute of Intelligence Science and Technology, Zhuhai* 519031*, China;*
[4]*Pazhou Laboratory* (*Huangpu*)*, Guangzhou* 510555*, China*

Email: taopeng@ucas.ac.cn, lnchen@sjtu.edu.cn

Received November 26, 2023; accepted December 10, 2024; published online April 30, 2025

Abstract Traditional operational research methods have been the primary means of solving combinatorial optimization problems (COPs) for the past few decades. However, with the rapid increase in the scale of problems in real-world scenarios and the demand for online optimization, these methods face persistent challenges including computational complexity and optimality. In recent years, combinatorial optimization methods based on deep learning have rapidly evolved, progressing from tackling solely small-scale problems (e.g., the traveling salesman problem (TSP) with fewer than 100 cities) to swiftly delivering high-quality solutions for graphs containing up to a million nodes. Particularly, in the last two years, a multitude of studies has surfaced, demonstrating the ability to generalize learned models to large-scale problems with diverse distributions. This capability empowers deep learning-based methods to demonstrate robust competitiveness, even when challenged by professional solvers. Consequently, this review summarizes the methods employed in recent years for solving COPs through deep learning (including prompt learning), scrutinizes the strengths and weaknesses of these methods, and concludes by highlighting potential directions for mitigating these weaknesses.

Keywords combinatorial optimization, deep learning, prompt learning, traveling salesman problem, chaotic backpropagation, chaotic simulated annealing

MSC(2020) 90C11, 90C27

Citation: Tao P, Chen L N. Combinatorial optimization: From deep learning to large language models. Sci China Math, 2025, 68: 2519–2537, https://doi.org/10.1007/s11425-023-2364-2

1 Introduction

Combinatorial optimization problems (COPs) involve optimizing discrete variables, aiming to identify the optimal solution from a finite set of solutions. This is closely associated with numerous issues in scientific and industrial domains, such as logistics and transportation, circuit design and drug development [6, 22, 62, 80, 85, 136]. While the solution set of a COP is finite, the number of feasible solutions grows

† Current address: School of Mathematical Sciences and School of AI, Shanghai Jiao Tong University, Shanghai 200240, China
* Corresponding author

exponentially with the problem's scale. Consequently, when the problem is of significant scale, the solution set approaches infinity, making exhaustive enumeration ineffective in obtaining the optimal solution.

In recent decades, a plethora of combinatorial optimization methods have been developed. Exact methods such as branch-and-bound [59] and dynamic programming [7] decompose an original problem into several subproblems, solving them to obtain the solution to the original problem. While these algorithms can yield precise results, they are only suitable for small-scale COPs. As the scale increases, approximate methods such as greedy algorithms [119], local search methods [127], tabu search methods [26], stochastic simulated annealing [54], chaotic simulated annealing [54], evolutionary algorithms [82] and particle swarm optimization [49] can provide a relatively good solution within a feasible time. While approximate methods have been a focal point of research, the advent of the big data era has led to significantly large-scale COPs in the real world. Faced with these extensive practical issues, approximate methods still struggle to obtain high-quality feasible solutions within acceptable timescales.

On the other hand, artificial intelligence (AI) technologies, with deep learning as a representative, have been rapidly advancing in recent years and have achieved remarkable milestones in various fields, including disease diagnosis [95], time series prediction [13,91], machine translation [92], autonomous driving [60] and protein structure prediction [102]. In the field of combinatorial optimization, AI methods demonstrate various unique advantages, such as fast computational speed and strong generalization capabilities, in comparison with traditional methods. While there are already some reviews in this field [6,31,65,80,120], due to the rapid development in this field over the past two years, many significant advancements have not been covered such as combinatorial optimization methods based on unsupervised learning (UL) and prompt learning (PL). Therefore, this review further examines and summarizes deep learning-based combinatorial optimization methods, compares their advantages and disadvantages, and finally identifies the main challenges and potential solutions in this field.

2 Deep learning methods for combinatorial optimization

Methods for solving COPs based on deep learning can be classified according to multiple criteria. Based on the role of deep learning, they can be categorized as end-to-end and non-end-to-end, with the former primarily relying on deep learning, while the latter is often employed to enhance traditional operational research methods. According to the construction of solutions, they can be classified into learning constructive heuristics (LCH) and learning improvement heuristics (LIH) [9]. Additionally, classification can be based on the neural network models used in deep learning such as the Pointer network (Ptr-Net) and graph neural networks (GNNs). Below, we first classify these methods based on the COPs (mainly focused on the TSP, see Table 1), then based on the model architectures, and finally based on the training framework, i.e., supervised learning (SL), reinforcement learning (RL) and unsupervised learning, including the gradient dynamics-based algorithm and the chaotic dynamics-based algorithm. Considering space limitations, we see that this survey will not delve into detailed explanations of related fundamental concepts and methods. For those unfamiliar with deep learning (including reinforcement learning and GNNs), the references [27,110,129] are recommended. For combinatorial optimization, the reference [56] is suggested. It should be noted that the field of solving COPs with deep learning has experienced rapid development, and there is a plethora of relevant work. This review, due to space constraints, cannot cover all the work in this area. We apologize to authors who have made significant contributions but may have been omitted in this paper.

2.1 The traveling salesman problem (TSP)

2.1.1 *The Hopfield neural network* (*HNN*)

The use of neural networks to solve COPs can be traced back to the Hopfield neural network (HNN) in 1985 (see [38]). This network belongs to a single-layer fully connected recurrent network, where the state of neurons changes over time, and each neuron serves as both input and output. Since the HNN can be viewed as a nonlinear dynamical system, the concept of an energy function is introduced to assess the stability of network iterations. In this context, if the state of neurons encodes a solution to the TSP, then the network's energy is minimal precisely when it corresponds to the optimal TSP solution. However, the Hopfield network can only address exceedingly small TSP instances, and owing to the training process tending to converge to local minima, we see that the solutions obtained are frequently suboptimal. Inspired by the chaotic dynamics found in the real brain, Aihara et al. [2] introduced chaos dynamics into the Hopfield network, proposing the chaotic neural network (ACNN or Aihara network). Subsequently, Chen and Aihara [10] further introduced the transient chaotic neural network (TCNN or Chen network), which employs chaotic simulated annealing (CSA) to help the network escape from local minima. Due to the pseudorandom nature of chaotic dynamics and the ergodicity in the fractal space [11, 12], this approach significantly improves the quality of TSP solutions, compared with the other algorithms including stochastic simulated annealing (SSA). Although this approach has been gradually optimized [117, 122], it still can only handle small-scale COPs.

Table 1 The table presents a summary of the deep learning-based combinatorial optimization methods discussed in this review, including information on the combinatorial optimization problems (COPs) addressed, the network model and learning framework used, and the maximum number of variables ($n_{\max}$) tested for each method

COP	Model	Framework	Reference (Year)	$n_{\max}$
TSP	HNN	UL	Hopfield and Tank (1985) [38]	30
	HNN	UL	Chen and Aihara (1995) [10]	48
	Ptr-Net	SL	Vinyals et al. (2015) [121]	50
	Ptr-Net	RL	Bello et al. (2016) [5]	50
	Ptr-Net	RL	Nazari et al. (2018) [86]	100
	GNN	SL	Nowak et al. (2017) [87]	20
	GNN	SL	Joshi et al. (2019) [48]	100
	GNN	SL	Prates et al. (2019) [93]	80
	GNN	SL	Xin et al. (2021) [130]	10,000
	GNN	SL	Fu et al. (2021) [23]	10,000
	GNN	SL	Sun and Yang (2023) [107]	10,000
	GNN	SL	Li et al. (2023) [68]	1,000
	GNN	RL	Dai et al. (2017) [18]	1,200
	GNN	RL	Ma et al. (2019) [76]	1,000
	GNN	RL	Hudson et al. (2022) [43]	100
	GNN	RL	Jiang et al. (2022) [45]	200
	GNN	RL	Qiu et al. (2022) [94]	10,000
	GNN	RL	Ye et al. (2023) [133]	1,000
	CNN	SL	Graikos et al. (2022) [29]	200
	Transformer	SL	Luo et al. (2023) [75]	1,000
	Transformer	RL	Deudon et al. (2018) [19]	100
	Transformer	RL	Kool et al. (2018) [55]	100
	Transformer	RL	Kwon et al. (2020) [58]	100
	Transformer	RL	Wu et al. (2021) [128]	200
	Transformer	RL	Ma et al. (2021) [77]	200
	Transformer	RL	Kim and Park (2021) [50]	500
	Transformer	RL	Zheng et al. (2021) [140]	85,900
	Transformer	RL	Bi et al. (2022) [8]	$\leqslant 200$
	Transformer	RL	Kim et al. (2022) [51]	< 250
	Transformer	RL	Hottung et al. (2021) [40]	200
	Transformer	RL	Son et al. (2023) [105]	1,000
	Transformer	RL	Hou et al. (2023) [41]	7,000
	Transformer	RL	Pan et al. (2023) [89]	10,000
	Transformer	RL	Cheng et al. (2023) [15]	20,000
	Transformer	RL	Zhou et al. (2023) [141]	5,000
	LLM	PL	Wang et al. (2023) [123]	100
	LLM	PL	Yang et al. (2023) [131]	50
	LLM	PL	Liu et al. (2023) [72]	25
	LLM	PL	Liu et al. (2023) [71]	1,000

(*To be continued on the next page*)

(*Continued*)

COP	Model	Framework	Reference (Year)	$n_{\max}$
VRP	Ptr-Net	RL	Nazari et al. (2018) [86]	100
	GNN	SL	Xin et al. (2021) [130]	10,000
	GNN	RL	Gao et al. (2020) [24]	400
	GNN	RL	Jiang et al. (2022) [45]	200
	GNN	RL	Ye et al. (2023) [133]	1,000
	Transformer	SL	Li et al. (2021) [66]	3,000
	Transformer	SL	Luo et al. (2023) [75]	1,000
	Transformer	RL	Kool et al. (2018) [55]	100
	Transformer	RL	Chen and Tian (2019) [14]	100
	Transformer	RL	Lu et al. (2019) [74]	100
	Transformer	RL	Kwon et al. (2020) [58]	100
	Transformer	RL	Wu et al. (2021) [128]	200
	Transformer	RL	Ma et al. (2021) [77]	200
	Transformer	RL	Kim and Park (2021) [50]	500
	Transformer	RL	Bi et al. (2022) [8]	$\leqslant 200$
	Transformer	RL	Kim et al. (2022) [51]	< 250
	Transformer	RL	Hottung et al. (2021) [40]	200
	Transformer	RL	Son et al. (2023) [105]	1,000
	Transformer	RL	Hou et al. (2023) [41]	7,000
	Transformer	RL	Zhou et al. (2023) [141]	5,000
MIS	GNN	SL	Li et al. (2018) [69]	$\sim$ 100,000
	GNN	RL	Sun and Yang (2023) [107]	10,000
	GNN	RL	Li et al. (2023) [68]	1,000
	GNN	RL	Qiu et al. (2022) [94]	10,000
	GNN	UL	Schuetz et al. (2022) [98]	1,000,000
	GNN	UL	Tao et al. (2024) [114]	1,000,000
	RNN	UL	Toenshoff et al. (2021) [116]	5,000
MCut	GNN	RL	Abe et al. (2019) [1]	$< 5{,}000$
	GNN	RL	Barrett et al. (2020) [4]	2,000
	GNN	UL	Yao et al. (2019) [132]	500
	GNN	UL	Schuetz et al. (2022) [98]	1,000,000
	GNN	UL	Tao et al. (2024) [114]	1,000,000
	RNN	UL	Toenshoff et al. (2021) [116]	5,000
MVC	GNN	SL	Li et al. (2018) [69]	$\sim$100,000
	GNN	RL	Dai et al. (2017) [18]	1,200
	GNN	RL	Abe et al. (2019) [1]	$< 5{,}000$
	GNN	RL	Manchanda et al. (2020) [78]	20,000
GC	GNN	SL	Lemos et al. (2019) [61]	561
	GNN	UL	Schuetz et al. (2022) [98]	19,717
	GNN	UL	Li et al. (2022) [67]	19,717
	GNN	UL	Tao et al. (2024) [114]	19,717
SAT	GNN	SL	Li et al. (2018) [69]	$\sim$100,000
	GNN	SL	Selsam et al. (2018) [101]	200
	GNN	SL	Zhang et al. (2020) [138]	2,000
	GNN	SL	Li and Si (2022) [70]	600
	GNN	RL	Yolcu and Póczos (2019) [135]	100
	GNN	RL	Kurin et al. (2020) [57]	600
MILP	GNN	SL	Ding et al. (2020) [20]	–
	GNN	SL	Gupta et al. (2020) [33]	2,000
	GNN	RL	Gasse et al. (2019) [25]	2,000
	GNN	RL	Nair et al. (2020) [84]	1,000,000
	GNN	RL	Tang et al. (2020) [112]	121
	GNN	RL	Paulus et al. (2022) [90]	66
	Ptr-Net	RL	Wang et al. (2023) [124]	61,000

2.1.2 *The pointer network (Ptr-Net)*

A prominent work utilizing a supervised learning strategy to solve COPs is Ptr-Net proposed by Vinyals et al. [121] for solving the traveling salesman problem (TSP). Its core is an encoder-decoder structure-based seq2seq model. As depicted in Figure 1(a), for a TSP with n cities, Ptr-Net initially transforms the two-dimensional coordinates of each city, $(x_i, y_i), i \in [1, 2, \dots, n]$ into high-dimensional representation vectors $\boldsymbol{c}_i$. Subsequently, it utilizes an encoder to encode the input representation vector sequence, resulting in a feature vector $\boldsymbol{e}_i$. The decoder then decodes the output vectors $\boldsymbol{d}_t, t = 1, 2, \dots$ in an autoregressive manner. Finally, the attention mechanism is employed to calculate the probability u_i^t of selecting the i-th city at the t-th step, where the city with the maximum probability at each step forms the final solution. Here, the calculation formula for u_i^t is given by

$$u_i^t = \boldsymbol{v}^{\mathrm{T}} \tanh(\boldsymbol{W}_1 \boldsymbol{e}_i + \boldsymbol{W}_2 \boldsymbol{d}_t), \tag{2.1}$$

where $\boldsymbol{W}$ and $\boldsymbol{v}$ represent the parameters of the neural network. In contrast to supervised learning, reinforcement learning excels in sequential decision-making, aligning well with the variable selection process in COPs. Additionally, it overcomes the dependence on high-quality training samples, a challenge in supervised learning. Therefore, leveraging reinforcement learning to address COPs is highly appropriate and currently a focal point of research. Reinforcement learning was initially employed by Bello et al. [5]

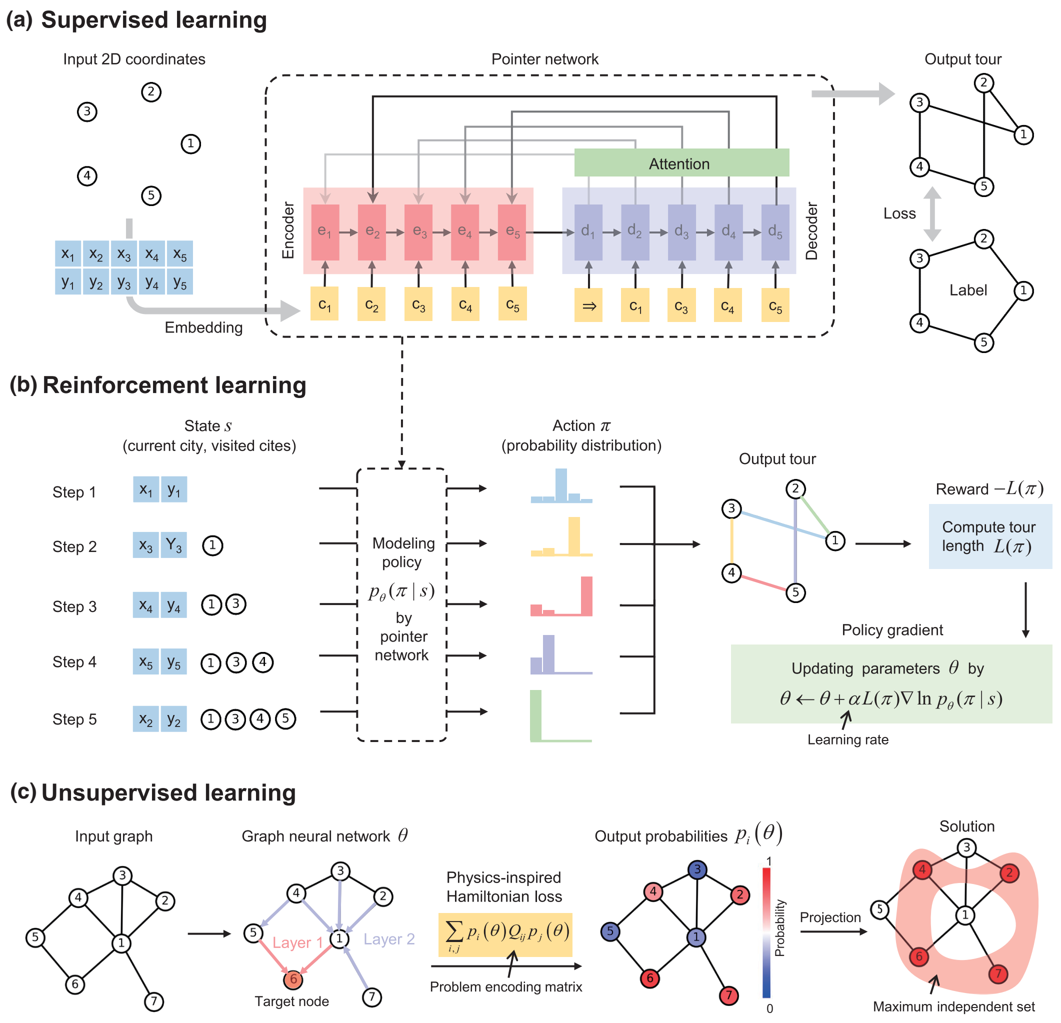
(a) Supervised learning
Input 2D coordinates
Pointer network
Output tour
Attention
Encoder
Decoder
Loss
Label
Embedding
(b) Reinforcement learning
State s (current city, visited cites)
Action π (probability distribution)
Output tour
Step 1
Step 2
Step 3
Step 4
Step 5
Modeling policy $p_\theta(\pi \mid s)$ by pointer network
Reward $-L(\pi)$
Compute tour length $L(\pi)$
Policy gradient
Updating parameters θ by $\theta \leftarrow \theta + \alpha L(\pi) \nabla \ln p_\theta(\pi \mid s)$
Learning rate
(c) Unsupervised learning
Input graph
Graph neural network θ
Layer 1
Layer 2
Target node
Physics-inspired Hamiltonian loss
$\sum_{i,j} p_i(\theta) Q_{ij} p_j(\theta)$
Problem encoding matrix
Output probabilities $p_i(\theta)$
Probability
Projection
Solution
Maximum independent set

Figure 1 (Color online) Three representative deep learning-based combinatorial optimization methods to train Ptr-Net (Figure 1(b) provides a simplified diagram). For each instance, the authors used the expected tour length $L(\pi)$ as a reward signal to train a policy $p_\theta(\pi|s)$ that outputs city selections (action π) based on the current state (s, including current city and visited cities). The policy gradient method can be used to update the network parameters θ, e.g., for the vanilla REINFORCE [126] algorithm, its update formula is

$$\theta \leftarrow \theta + \alpha L(\pi)\nabla \ln p_\theta(\pi|s), \tag{2.2}$$

where α is the learning rate. This work used a more advanced actor-critic [111] algorithm to further improve the stability of training. Testing on TSP instances with up to 100 cities indicated that this reinforcement learning-based method outperforms the results of Vinyals et al. [121]. It is worth noting that to achieve optimal performance during testing, Bello et al. [5] fine-tuned the model's weights for each instance.

2.1.3 *The graph neural network*

GNNs are also efficient tools for solving TSPs. For example, Nowak et al. [87] input the two-dimensional coordinates of TSP instances into a simple GNN to obtain the probability of each edge in the TSP path, but this GNN can only solve small TSP instances approximately. Joshi et al. [48] used a graph convolutional network (GCN) [53] to replace the GNN of Nowak et al. [87], and then utilized beam search to convert these probabilities into an effective path with a simulation-based strategy further enhancing the optimization performance [16]. Prates et al. [93] trained a GNN to encode nodes and edges in the TSP, determining whether the model's output, under a given loss, can form an effective path. Xin et al. [130] introduced the NeuroLKH method, which trained a sparse graph network (SGN) through supervised learning to enhance the edge candidate set artificially designed in the Lin-Kernighan-Helsgaun (LKH) algorithm. The quality of its solutions surpassed the LKH algorithm in TSP instances with up to 10,000 cities. It is important to note that due to the differing data distributions between the training and test sets, methods based on supervised learning often struggle to generalize well on the test set [47]. Some recent work has begun to address this problem. Fu et al. [23] employed various techniques such as graph sampling, graph transformation, and heat map fusion to train a small model, and then utilized the Monte Carlo tree search (MCTS) to guide the search for high-quality solutions, achieving generalization even on TSP instances with up to 10,000 nodes. Graikos et al. [29] pioneered the use of a diffusion model [37] to generate solutions for COPs. This method projects each TSP instance onto a 64×64 grayscale image space and processes it using a convolutional neural network (CNN). Building on the model of Graikos et al. [29], Sun and Yang [107] proposed DIFUSCO, utilizing a GNN instead of a CNN. This explicit modeling of variables further improves its optimization performance. Li et al. [68] introduced the T2TCO (training to testing) framework. During training, it leverages the diffusion model to estimate the distribution of high-quality solutions for each example. During testing, it performs a gradient-based search in the solution space. Experimental results on datasets with 500 and 1,000 cities demonstrate that T2TCO significantly outperforms DIFUSCO in terms of generalization performance.

Dai et al. [18] proposed the S2V-DQN method, using a GNN to encode the structure of solutions and calculate the Q-values of the remaining optional nodes. Then, based on a greedy policy, they progressively constructed a complete solution. Training the GNN through deep Q-learning [83] allowed the method to approach the results of the CPLEX solver on the MVC problem, the MC problem and the TSP with up to 1,200 nodes. Ma et al. [76] introduced a GNN to Ptr-Net, encoding all cities with the GNN to obtain the graph embedding of each city. Simultaneously, they combined the point embeddings of each city to select the next city. Additionally, the authors employed hierarchical reinforcement learning to train the network, achieving efficient solving of the TSP with up to 1,000 cities. To further enhance the solving speed, Hudson et al. [43] proposed a hybrid data-driven approach based on GNNs and guided local search (GLS). It first predicted the regrets of including each edge of the graph in the solution, and then merged these predictions with the original graph, inputting them into GLS to output the final solution. While reinforcement learning-based combinatorial optimization methods have achieved considerable success, their generalization performance remains limited for out-of-distribution data [46].

To address this challenge, Jiang et al. [45] introduced group distributionally robust optimization (GDRO), which alternately optimizes the weights of different group distributions and the parameters of deep models during the training process. Qiu et al. [94] proposed the DIMES method, introducing a compact continuous space to parameterize the underlying distribution of candidate solutions, making training based on REINFORCE more stable. They also utilized meta-learning [39] for the effective initialization of model parameters during fine-tuning. Ant colony optimization (ACO) [21] is a classic heuristic method, but for specific problems, expert knowledge is often required to ensure solution quality. To address this, Ye et al. [133] proposed DeepACO, which utilizes deep reinforcement learning for automatic heuristic design, achieving performance surpassing the original ACO on 8 different COPs.

2.1.4 *Transformer*

Recently, Transformer has also achieved some competitive results. Luo et al. [75] introduced a novel light encoder and heavy decoder (LEHD) model to dynamically capture relationships between all available nodes of different sizes, enhancing the model's generalization performance on large-scale problems. To enhance the extraction of instance features, Deudon et al. [19] replaced the seq2seq model in Ptr-Net with Transformer [118] and further optimized the results using a simple 2-opt [17] method. However, Kool et al. [55] noted that the model used by Deudon et al. [19] did not fundamentally outperform traditional Ptr-Net. To harness the potential of Transformer, the authors proposed the attention model (AM) method. They first refined the decoding process, focusing on the first step and the last two steps of decision-making during decoding. Additionally, they introduced a rollout baseline to replace the critic network. Moreover, the best policy model during training serves as the baseline, and the parameters update only if a policy surpasses this baseline. These improvements resulted in the optimization performance surpassing previous reinforcement learning-based approaches and professional solvers like Concorde [3], LKH3 [36] and Gurobi [88]. Kim and Park [50] utilized a model similar to Kool et al. [55], introducing a learning collaborative policy (LCP) consisting of seeder and reviser strategies to further enhance optimization performance. The seeder strategy generates as diverse possible solutions as possible, while the reviser strategy decomposes these solutions into multiple parts for individual optimization, combining them into a superior solution. Wu et al. [128] further employed deep reinforcement learning to directly learn a superior rule. Additionally, earlier work did not account for the symmetry of solutions in path problems. Therefore, Kwon et al. [58] proposed the policy optimization with the multiple optima (POMO) method, leveraging the symmetry of solutions to reduce the baseline in the REINFORCE algorithm, making training faster and more stable. It is noteworthy that the GDRO method proposed by Jiang et al. [45] can be integrated with the POMO method, and thereby enabling training in the context of reinforcement learning. Based on the POMO method, Hottung et al. [40] introduced effective active search (EAS), which, compared with actively searching and fine-tuning all weights, selectively adjusts a small subset of weights for better optimization performance. To adapt the model to larger-scale problems, Son et al. [105] proposed the meta-SAGE method, where the scale meta-learner (SML) transforms context embeddings, and the scheduled adaptation with guided exploration (SAGE) adjusts model parameters for specific instances based on scale information. Ma et al. [77] found that positional encoding (PE) in Transformer is not suitable for path problems such as the TSP. Hence, they developed the cyclic positional encoding (CPE) method to capture the symmetry of solutions and proposed the dual-aspect collaborative Transformer (DACT) to separately learn embeddings of node and position features. Zheng et al. [140] simultaneously used three reinforcement learning methods (Q-learning [83], Sarsa [109] and Monte Carlo) to improve the LKH method, replacing the α value used for candidate city selection and sorting in LKH with a Q-value. The effectiveness was confirmed on the TSPLIB dataset with up to 85,900 cities, although the overall performance was inferior to NeuroLKH [130]. Bi et al. [8] presented an adaptive multi-distribution knowledge distillation (AMDKD) approach, using various knowledge from multiple teachers trained on example distributions to generate a lightweight student model, and thereby learning a more generalized deep model. Kim et al. [51] introduced Sym-NCO, a regularization-based training approach that leverages symmetries such as rotation and reflection invariance in various COPs and solutions to improve the generalization ability. Pan et al. [89] adopted a hierarchical reinforcement learning strategy, where the upper-level policy selects a small subset of nodes, and the lower-level policy outputs a route connecting these nodes to the existing partial route. This end-to-end approach allows for rapid solving of COPs with 10,000 nodes. Hou et al. [41] proposed the two-stage divide method (TAM) to generate subproblems, and then designed a two-step reinforcement learning process for training, ultimately achieving real-time solving of large-scale VRPs. Cheng et al. [15] iteratively optimized a subproblem, extending small-scale selectors and optimizers to large-scale TSP instances, significantly reducing solving time due to efficient parallel computing. Zhou et al. [141] introduced a general meta-learning framework that utilizes second-order techniques for effective model initialization, efficiently adapting to new tasks with limited data during the inference process.

2.1.5 *The large language model* (*LLM*)

LLMs have made tremendous breakthroughs in the last two years, significantly impacting various industries, for example, large models in computational biology like Geneformer [115] and time series models such as one fits all [142]. Some researchers have begun attempting to solve COPs based on LLMs. Wang et al. [123] were the first to use deep reinforcement learning to train bidirectional encoder representations from transformers (BERT) for solving COPs (named BDRL). Compared with previous encoder-decoder frameworks, there is no need to redesign the decoder for different problems, and the model can be fine-tuned for specific tasks. By pretraining BERT on a large number of TSP/VRP-20/50/100 instances and fine-tuning it on constrained variant tasks (CVRP, PCTSP), BDRL achieved better results than several reinforcement learning-based methods, such as AM. However, training large models is time-consuming and labor-intensive, and there is a growing interest in solving problems solely through prompts. Recently, Yang et al. [131] proposed a simple and effective method called optimization by prompting (OPRO) that utilizes LLMs as optimizers, where the optimization tasks are described in natural language. In each optimization step, the LLM generates new solutions from prompts that include previously generated solutions and their values, evaluates these new solutions, and adds them to the prompts for the next optimization step (see Figure 2(a)). They conducted OPRO performance tests on the TSP with 10 to 50 cities and compared them with the nearest neighbor (NN) and farthest insertion (FI) methods. The LLMs used in the tests include text-bison1), GPT3.5-turbo, and GPT42). They found that when $n = 10$, all the large models can find the optimal solution, but when $n = 50$, their performance rapidly declines and is surpassed by the FI method. Additionally, they observed that GPT4 outperforms GPT3.5-turbo and text-bison on all problem sizes. Liu et al. [72] further introduced the concept of evolutionary algorithms, which instructs the LLM to select parent solutions from the current population, and performs crossover and mutation to generate offspring solutions. Then these new solutions are included in the population for the next generation. Although the introduction of evolutionary algorithms improves the performance of OPRO, there is still a significant gap compared with heuristic methods such as FI.

The above works use LLMs to generate new solutions at the operator level. However, their performance declines considerably when applied to large-scale problems, mainly due to the longer solution representation and large search space. To address this limitation, Liu et al. [71] proposed a novel approach called AEL. It utilized an LLM to automatically generate optimization algorithms via an evolutionary framework. The results show that heuristic methods derived from AEL can achieve better results than simple hand-crafted and LLM-generated heuristics in the TSP with up to 1,000 cities. Recently, an inspiring work was published in Nature, which introduced the FunSearch [96] method. It achieved the best results in the cap set problem and significantly outperformed the first fit and best fit methods in the online bin packing problem. Compared with previous methods, FunSearch uses a simple program template as a prompt, and the LLM only needs to generate the most essential functions. Additionally, FunSearch utilizes code-specific models (Codey3)) instead of general-purpose LLMs for generation and employs an island model to update the function database (see Figure 2(b)).

1) https://cloud.google.com/vertex-ai/docs/generative-ai/learn/models

2) http://openai.com/api/

3) https://cloud.google.com/vertex-ai/docs/generative-ai/code/code-models-overview

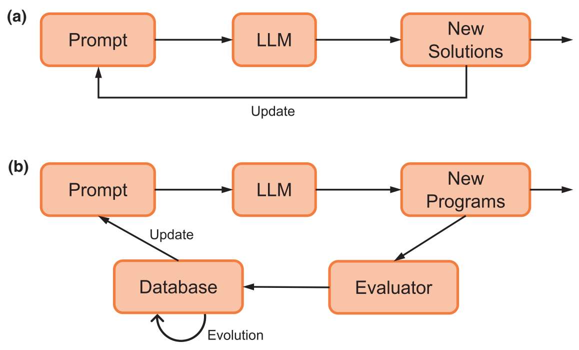

Figure 2 (Color online) Two representative prompt learning-based combinatorial optimization methods, which leverage LLMs to generate either (a) new solutions and (b) new programs, respectively

2.2 The vehicle routing problem (VRP)

The VRP is a more complex version of the TSP, with the objective of determining the best route of nodes (or cities) to minimize the total travel expense, while still adhering to other restrictions like capacity limits. A number of strategies outlined in the previous section for resolving the TSP are also applicable to the VRP. However, in this subsection, we focus solely on techniques that are uniquely tailored for the VRP. Nazari et al. [86] addressed the dynamic features of the VRP. Considering both customer coordinates (static) and demands (dynamic) simultaneously, the authors replaced the recurrent layer in the encoder with a one-dimensional convolutional layer, significantly improving computational speed. While its performance on the TSP did not surpass that of Bello et al. [5], it outperformed classical heuristic methods on the VRP.

Gao et al. [24] introduced a graph attention network (GAT) to encode problem features, and then used destroy and repair policies to improve solution quality. These policies were trained using the proximal policy optimization (PPO) [100] algorithm and the testing results on various scales of the capacitated vehicle routing problem (CVRP) indicated that this method outperformed that of Kool et al. [55].

Li et al. [66] proposed a learning-enhanced local search framework, learning-to-delegate, and it identifies appropriate subproblems and iteratively improves the quality of solutions using a black-box solver. Leveraging spatial locality, it only needs to consider a linear number of subproblems at each iteration. As a result, this approach accelerates SOTA VRP solvers by 10× to 100×, while still achieving competitive solution qualities for VRPs ranging in size from 500 to 3,000.

Notably, deep reinforcement learning methods can automatically learn search rules, often outperforming manually designed rules. An early example of such an approach is NeuRewriter proposed by Chen and Tian [14], which constructs a feasible solution and then guides the local search process using the region-picker and rule-picker policies (trained using an actor-critic algorithm) to continuously improve solution quality. This method outperforms OR-tools in the job shop scheduling problem (JSSP) and the VRP. Lu et al. [74] proposed the learn-to-improve (LSI) framework, which utilizes deep reinforcement learning to train a selection policy. In each iteration of local search, this policy selects one operator from a library of 9 improvement operators to enhance the current solution. When the solution reaches a local optimum, it is perturbed. Testing on different scales of the CVRP showed that LSI outperformed LKH in both solving speed and solution quality. GNNs efficiently represent graph structure information in COPs, making them a recent research hotspot.

2.3 The maximal independent set (MIS)

Li et al. [69] estimated the likelihood (probability) of a node belonging to the optimal solution on a graph by training a GCN. Based on these probabilities, they used tree search to construct feasible solutions, ultimately selecting the best solution from numerous possibilities. The method was applied to solve various COPs, including the MIS, with results indicating superior optimization performance compared with several benchmark methods. In addition, both the previously mentioned T2TCO and DIFUSCO are also based on supervised learning and the GNN to solve the MIS problem, while DIMES uses a reinforcement learning strategy. The emergence of unsupervised deep learning methods has only occurred in recent years. Toenshoff et al. [116] proposed a recurrent unsupervised neural network for the constraint satisfaction problem (CSP), RUN-CSP. It is based on a constraint language that can automatically or manually design a loss function. Optimizing this loss function yields solutions to the CSP, and the method has demonstrated results close to traditional greedy algorithms on CSPs with up to 5,000 nodes. The physics-inspired GNN (PIGNN), recently proposed by Schuetz et al. [98], represents an example of the unsupervised learning paradigm. Initially, it encodes the objective of COPs into the Hamiltonian corresponding to the quadratic unconstrained binary optimization (QUBO) problem

$$H = \boldsymbol{x}^{\mathrm{T}} \boldsymbol{Q} \boldsymbol{x} = \sum_{i,j} x_i Q_{ij} x_j, \tag{2.3}$$

where $\boldsymbol{Q}$ is a constant matrix encoding the COP and $x_i \in \{0,1\}$ is the state of variable i. For example, for the MIS problem in Figure 1(c), the Hamiltonian can be written as

$$H_{\mathrm{MIS}} = -\sum_{i \in V} x_i + P \sum_{(i,j) \in E} x_i x_j, \tag{2.4}$$

where V and E are the sets of nodes and edges of a given graph, respectively. P is a penalty parameter, usually set to 2. PIGNN transforms H_{MIS} into a differentiable loss function by replacing x_i with $p_i(\theta)$, where θ represents all parameters in a GNN and represents the final output of the GNN model on node i, i.e.,

$$\mathrm{loss}_{\mathrm{MIS}}(\theta) = \sum_{i,j} p_i(\theta) Q_{ij} p_j(\theta). \tag{2.5}$$

Just optimize this loss function to get $p_i(\theta)$, and a simple projection operation (the simplest one is to use a threshold of 0.5, where $p_i(\theta)$ above the threshold is set to be 1 and the values below the threshold are set to be 0, which will give the final solution). Two significant advantages of the PIGNN over previous work are that it can be applied to all COPs that can be transformed into QUBOs and that it can handle large-scale (up to a million variables) COPs. From the last section, it is evident that the PIGNN transforms the solution process of COPs into the optimization process of a loss function. Therefore, the learning algorithm of neural networks plays a decisive role in the results of COPs. Although deep learning has achieved remarkable results, the cornerstone of these achievements from the perspective of learning algorithms is still the famous error backpropagation (BP) [97] algorithm and its variants such as SGD [108] and Adam [52]. Despite the success of BP-based algorithms, they still suffer from two significant flaws. Firstly, due to the fact that the BP algorithm is primarily based on gradient dynamics, it is prone to get stuck in local minima (its variants may introduce stochastic dynamics to alleviate this). Secondly, from a biological perspective, existing experimental results [79, 104] suggest that the brain uses chaotic dynamics to process information rather than gradient dynamics. In order to address these two flaws of the BP algorithm, Tao et al. [113] proposed the chaotic backpropagation (CBP) algorithm, which introduces a loss function $\mathrm{loss}_{\mathrm{chaos}}$ from the internal neural interaction of a neural network, formulated as follows:

$$\mathrm{loss}_{\mathrm{chaos}} = -\sum_{i=1}^{l} \sum_{j=1}^{M_i} z_{ij} [I_0 \ln x_{ij} + (1 - I_0) \ln(1 - x_{ij})], \tag{2.6}$$

where x_{ij} is the output of the j-th neuron in the i-th layer of a multilayer perceptron (MLP) with l layers. There are M_i neurons in the i-th layer and w_{ijk} is the weight from $x_{i-1,k}$ to x_{ij}. I_0 is a constant between 0 and 1, and z_{ij} is a scalar parameter that controls the chaotic intensity (or annealing temperature) of w_{ijk}. It can be demonstrated in [113] that when z_{ij} is sufficiently large, the introduction of $\mathrm{loss}_{\mathrm{chaos}}$ leads to the emergence of chaotic dynamics for w_{ijk}. Due to the ergodicity of chaotic dynamics in the fractal space, w_{ijk} can sample a broader parameter space. Furthermore, as z_{ij} undergoes annealing ($z_{ij} \leftarrow \beta\, z_{ij}$, where $\beta > 0$ is annealing constant less than 1, e.g., 0.999), the corresponding $\mathrm{loss}_{\mathrm{chaos}}$ gradually approaches zero, at which point the CBP algorithm degenerates into the BP algorithm, ensuring the convergence of learning. Experimental results on benchmark datasets such as CIFAR10 demonstrate that the CBP algorithm not only outperforms BP in terms of optimization performance on the training set but also exhibits superior generalization performance on the test set. Recently, the authors further extended the CBP algorithm to GNNs and combined it with the PIGNN for solving large-scale COPs [114], and spiking neural networks [125]. The results show that it significantly outperforms the BP-based PIGNN on the MCut, MIS and GC, providing a promising approach for solving large-scale COPs.

2.4 Maximum cut (MCut)

To solve MCut problems, Abe et al. [1] improved the generalization ability by replacing the Q-learning in S2V-DQN with the more efficient AlphaGo Zero [103]. However, previous work constructed solutions by incrementally adding nodes, preventing the agent from correcting previous decisions and resulting in suboptimal optimization performance. Therefore, Barrett et al. [4] proposed training an agent through reinforcement learning to reassess the Q-values of adding or removing nodes during testing, achieving corrections for early decisions.

Yao et al. [132] directly employed the MCut number as a loss function. By introducing relaxation, they made the loss function differentiable. Ultimately, an unsupervised learning approach was utilized to solve the MCut problem with 500 nodes. In contrast, based on the QUBO framework, the PIGNN further increases the size of problems that can be handled to 1,000,000.

2.5 Minimal vertex cover (MVC)

The S2V-DQN model proposed by Dai et al. [18] has been demonstrated to tackle the MVC problem effectively, showing a capacity to generalize for graphs with up to 1,200 nodes. Using the S2V-DQN model as a reference point, the methods put forth by Li et al. [69] and Abe et al. [1] have shown superior results. Recently, Manchanda et al. [78] employed an innovative probabilistic greedy mechanism to estimate a node's quality and applied GCNs to MVC problems involving up to 5,000 nodes. This approach yielded a noteworthy enhancement in performance compared with the previously mentioned methods.

2.6 Graph coloring (GC)

The GC problem is a common extension of the MCut problem. The goal of the GC problem is to find the smallest number of colors needed to color the nodes of a graph such that no edge connects two nodes of the same color. Lemos et al. [61] trained a simple GNN on a large number of randomly generated graphs to solve the GC problem, showing that its performance can surpass tabu search and greedy methods under certain graph distributions. More recently, the unsupervised learning approach of the PIGNN was expanded to handle more complex GC problems. This method outperformed traditional tabu search techniques on citation graphs with up to 20,000 nodes [67, 99].

2.7 Satisfiability (SAT)

Selsam et al. [101] introduced NeuroSAT, a model that predicts the satisfiability of SAT problems using a trained message-passing neural network. The method maintains the permutation invariance and negation invariance of Boolean formulae via symmetric edge connections and message passing. While the optimization performance of this approach might not match up to state-of-the-art (SOTA) solvers, it does demonstrate significant benefits in terms of computational efficiency and generalization performance. In contrast to the end-to-end approach of NeuroSAT, Zhang et al. [138] put forth NLocalSAT. This method enhances the performance of the stochastic local search (SLS) solver by guiding the initial assignments with a GCN. The output of this network is a predicted solution. It is important to note that the aforementioned methods aim to predict a single satisfying assignment for a satisfiable formula. However, there can be multiple satisfying solutions, which raises the question of which particular solution should be generated. To address this, NSNet [70] performs marginal inference in the solution space of an SAT problem, estimating the assignment distribution of each variable among all satisfying assignments. This approach provides a more comprehensive understanding of the solution space, potentially leading to more effective and diverse solutions.

Contrasting the supervised-learning based methods mentioned earlier, Yolcu et al. [135] introduced a variable selection heuristic for SLS solvers, computed by a GNN through deep reinforcement learning. In this methodology, the policy GNN takes the current assignment as input and outputs a probability distribution over variables, reflecting their likelihood of being flipped in the next iteration. Despite these advancements, the selection heuristics typically employed in SAT solvers often make suboptimal decisions. To address this, Kurin et al. [57] proposed Graph-Q-SAT. This method formulates the Boolean formulae as variable-clause graphs and learns a value function for each variable node, with a simple policy to select the variables with the maximum value. For a more in-depth discussion of deep learning-based SAT solvers, please refer to the review by Guo et al. [32].

2.8 Mixed integer linear programming (MILP)

MILP is indeed a widely used modeling technique for COPs. Traditionally, MILP can be solved using a linear-programming based branch-and-bound (B&B) algorithm. This algorithm partitions the search space by branching on variables' values and smartly uses bounds from problem relaxations to prune unpromising regions from the tree. However, solving a complex MILP problem often requires a large number of branching variable selection (BVS) decisions, which are crucial for the performance of the solver. The selection of branching variables typically relies on a multitude of expert-designed rules. Recently, deep learning-based methods have shown competitive results. For example, Ding et al. [20] created a tripartite graph from the MILP formulation and trained a GCN for variable solution prediction based on the collected features, labels and tripartite graphs. Gupta et al. [33] proposed a hybrid architecture that uses a GNN model only at the root node of the B&B tree and a weaker but faster predictor at the remaining nodes. This approach results in an effective time-accuracy trade-off in branching.

In contrast to SL-based methods, RL can learn a policy for BVS from scratch. This approach provides a more flexible and potentially more effective means of solving complex MILP problems. Gasee et al. [25] proposed a new GCN for learning branch-and-bound variable selection policies. This method takes advantage of the natural variable-constraint bipartite graph representation of MILPs. Nair et al. [84] encoded a MILP into a GCN as a bipartite graph and computed an initial feasible solution. The GCN is then trained to imitate the policy branching that un-assigns one variable at a time, interleaved with solving a sub-MILP problem to compute a new solution. This method shows a significant improvement over SCIP (a popular MILP solver) on both large-scale real-world application datasets and MIPLIB. The cutting plane technique is often integrated into the branch-and-bound method to improve efficiency. Some works have sought to use deep learning to select cutting planes. For example, Tang et al. [112] introduced a Markov decision process (MDP) formulation for the problem of sequentially selecting cutting planes for MILP, and trained an RL agent using evolutionary strategies. Inspired by the observation that a greedy selection rule looking ahead to select cuts that yield the best bound improvement delivers strong decisions for cut selection, Paulus et al. [90] proposed NeuralCut for imitation learning on the lookahead expert. Recently, Wang et al. [124] found that the order of selected cuts significantly impacts the efficiency of solving MILPs. Therefore, they proposed a novel hierarchical sequence model to learn policies by selecting an ordered subset via reinforcement learning. This method greatly improves the efficiency of solving MILPs compared with human-designed and learning-based baselines on both synthetic and large-scale real-world MILPs. For a more comprehensive discussion of the deep learning-based MILP solver, please refer to the reviews [42, 137].

3 Discussions and future directions

In the previous section, we have discussed how three different learning paradigms can address COPs. Specifically, end-to-end methods based on supervised learning exhibit significantly faster solving speeds than traditional operations research methods. Once the model is trained, it can rapidly solve problems of the same type through inference. However, these methods require a large number of samples that have already been solved with high quality to serve as a training set. Moreover, the quality of solving depends significantly on the training set, making it challenging for them to outperform traditional methods, especially in large-scale scenarios [134]. End-to-end methods based on reinforcement learning can avoid dependence on high-quality training sets and exhibit stronger generalization capabilities. However, the training speed is noticeably slower than that of methods based on supervised learning. End-to-end methods based on unsupervised learning can solve large-scale COPs. However, with the exception of a few cases of self-imitation learning [106], they can only solve one instance at a time, requiring a new training process for each given instance. Consequently, it is currently challenging to extend these methods to online optimization scenarios [44]. In addition to end-to-end methods, recently, there has been a focus on improving local search methods using deep reinforcement learning. These methods replace manually designed rules with learned search rules, resulting in strong optimization performance that in some scenarios, even surpasses that of professional solvers. However, it is crucial to note that the vast majority of them are heuristics, and their computational efficiencies are still usually much lower than supervised learning-based methods. Finally, directly using LLMs to generate solutions for COPs is not a suitable choice. On the contrary, utilizing LLMs to generate programs or functions has not only achieved significantly better performance but also demonstrated good interpretability. In conclusion, although there has been some progress in using LLMs and prompt learning to solve COPs, the overall field is still in its early stages [30]. For some classic COPs, these methods still have a noticeable performance gap compared with deep learning approaches based on traditional operations research. Deep integration with existing SOTA methods and automatic prompt optimization may bring brighter prospects to this field. In summary, current methods based entirely on deep learning (i.e., not relying on existing solvers at all) cannot achieve both computational efficiency and quality simultaneously higher than traditional operations research methods. In practical applications, a trade-off between speed and quality needs to be considered [73].

Through analyzing existing methods, it is evident that the choice of network models significantly influences optimization performance. For COPs where the order of nodes holds significance, such as the TSP and VRP, models based on attention mechanisms, like Transformer, often exhibit promising results [55, 86]. In contrast, for COPs where the order of nodes is irrelevant, such as the MC and MIS, graph neural networks are more commonly used [18, 69]. However, these two network models are not mutually exclusive, and some studies combining them have achieved favorable results [24, 76].

It is worth noting that deep learning-based combinatorial optimization methods have been developed for less than 10 years. In the early stages, these methods could only solve very small instances, such as the TSP with fewer than 100 cities [5, 121]. Moreover, both in terms of solving speed and solution quality, they fell far behind professional solvers. Subsequently, numerous efforts have been made to extend these methods to larger and more complex datasets and enhance solving speed using various approaches [23, 55]. Recent works have further focused on the generalization performance of these methods [68, 75, 105]. Although, currently, deep learning-based combinatorial optimization methods cannot yet replace professional solvers in many practical scenarios, considering that these professional solvers have undergone lengthy development and optimization, the progress achieved by deep learning-based combinatorial optimization methods is significant. We can anticipate the emergence of more universal and practical methods in this active research field in the foreseeable future.

Finally, while deep learning-based combinatorial optimization methods have achieved results surpassing professional solvers in some scenarios, overall, they are still in the early stages of development. This is especially true in the following aspects, where there is significant room for improvement.

(1) For end-to-end methods, the choice of the network model significantly influences optimization performance. For example, the multi-head attention layer in Transformer compared with the regular attention layer in Prt-Net, or GraphSAGE [35] compared with GCN. Therefore, designing more efficient network models tailored to different COPs is a crucial research direction.

(2) For deep learning-based local search methods, although they have achieved good optimization results, they are fundamentally search methods, and their solving speed is still much slower than end-to-

end methods. Therefore, improving the efficiency of their search is a crucial issue for such methods.

(3) Currently, there is relatively less research on unsupervised learning. The main issue is that each problem requires resolving, and it remains unknown whether there exists a more efficient algorithm to overcome this limitation.

(4) Most learning algorithms for neural networks are based on gradient dynamics, such as REINFORCE in reinforcement learning and Adam in supervised learning. However, learning algorithms based on gradient dynamics are prone to getting stuck in local optima. A recent research has proposed a learning algorithm inspired by the chaotic dynamics in the brain [113]. Introducing this chaotic learning algorithm into the training process to solve COPs is a promising avenue worth exploring.

(5) Most deep learning-based methods are designed for static, single-objective and unconstrained COPs. Exploring how to leverage deep learning to solve practical applications involving dynamic, multi-objective and constrained COPs is an important direction for future research [63, 64, 139].

(6) Quantum approximate optimization algorithms (QAQAs) have shown tremendous potential in solving combinatorial optimization problems, such as coherent Ising machines [28, 34, 81]. Combining deep learning methods with QAQA is also a promising direction worthy of in-depth research.

(7) Through prompt engineering, LLMs have the potential to offer superior solutions or programs for COPs, and preliminary progress has already been made. Consequently, the application of automatic prompt optimization [131] and the integration of SOTA methods hold great promise for tackling larger-scale and more intricate COPs.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant Nos. T2350003, 31930022, 12131020 and T2341007), the National Basic Research Program of China (Grant No. 2022YFA1004800) and Strategic Priority Research Program of the Chinese Academy of Sciences (Grant No. XDB38040400).

References

1 Abe K, Xu Z, Sato I, et al. Solving NP-hard problems on graphs with extended AlphaGo zero. arXiv:190511623, 2019

2 Aihara K, Takabe T, Toyoda M. Chaotic neural networks. Phys Lett A, 1990, 144: 333–340

3 Applegate D, Bixby R, Chvatal V, et al. Certification of an optimal TSP tour through 85,900 cities. Oper Res Lett, 2009, 37: 11–15

4 Barrett T, Clements W, Foerster J, et al. Exploratory combinatorial optimization with reinforcement learning. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence, vol. 34. Palo Alto: AAAI Press, 2020, 3251–3258

5 Bello I, Pham H, Le Q V, et al. Neural combinatorial optimization with reinforcement learning. arXiv:1611.09940, 2016

6 Bengio Y, Lodi A, Prouvost A. Machine learning for combinatorial optimization: A methodological tour d'horizon. European J Oper Res, 2021, 290: 405–421

7 Bertsekas D. Dynamic Programming and Optimal Control: Volume I. Belmont: Athena Sci, 2012

8 Bi J, Ma Y, Wang J, et al. Learning generalizable models for vehicle routing problems via knowledge distillation. In: Proceedings of the 36th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 35. La Jolla: NIPS, 2022, 31226–31238

9 Cai Q, Hang W, Mirhoseini A, et al. Reinforcement learning driven heuristic optimization. arXiv:190606639, 2019

10 Chen L, Aihara K. Chaotic simulated annealing by a neural network model with transient chaos. Neural Netw, 1995, 8: 915–930

11 Chen L, Aihara K. Chaos and asymptotical stability in discrete-time neural networks. Phys D, 1997, 104: 286–325

12 Chen L, Aihara K. Global searching ability of chaotic neural networks. IEEE Trans Circuits Syst I Regul Pap, 1999, 46: 974–993

13 Chen P, Liu R, Aihara K, et al. Autoreservoir computing for multistep ahead prediction based on the spatiotemporal information transformation. Nat Commun, 2020, 11: 4568

14 Chen X, Tian Y. Learning to perform local rewriting for combinatorial optimization. In: Proceedings of the 33rd International Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 32. La Jolla: NIPS, 2019, 6281–6292

15 Cheng H, Zheng H, Cong Y, et al. Select and optimize: Learning to solve large-scale TSP instances. In: Proceedings

of Machine Learning Research. International Conference on Artificial Intelligence and Statistics, vol. 206. San Diego: JMLR, 2023, 1219–1231

16 Choo J, Kwon Y-D, Kim J, et al. Simulation-guided beam search for neural combinatorial optimization. In: Proceedings of the 36th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 35. La Jolla: NIPS, 2022, 8760–8772

17 da Costa P R, Rhuggenaath J, Zhang Y, et al. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning. In: Proceedings of Machine Learning Research. Asian Conference on Machine Learning, vol. 129. San Diego: JMLR, 2020, 465–480

18 Dai H, Khalil E, Zhang Y, et al. Learning combinatorial optimization algorithms over graphs. In: Proceedings of the 31st Annual Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 30. La Jolla: NIPS, 2017, 1–11

19 Deudon M, Cournut P, Lacoste A, et al. Learning heuristics for the TSP by policy gradient. In: Proceedings of the 15th International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Lecture Notes in Computer Science, vol. 10848. Berlin: Springer-Verlag, 2018, 170–181

20 Ding J-Y, Zhang C, Shen L, et al. Accelerating primal solution findings for mixed integer programs based on solution prediction. In: Proceedings of the 34th AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence, vol. 34. Palo Alto: AAAI Press, 2020, 1452–1459

21 Dorigo M, Birattari M, Stutzle T. Ant colony optimization. IEEE Comput Intell Mag, 2006, 1: 28–39

22 Feng L, Huang Y, Zhou L, et al. Explicit evolutionary multitasking for combinatorial optimization: A case study on capacitated vehicle routing problem. IEEE Trans Cybern, 2021, 51: 3143–3156

23 Fu Z-H, Qiu K-B, Zha H. Generalize a small pre-trained model to arbitrarily large TSP instances. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence, vol. 35. Palo Alto: AAAI Press, 2021, 7474–7482

24 Gao L, Chen M, Chen Q, et al. Learn to design the heuristics for vehicle routing problem. arXiv:200208539, 2020

25 Gasse M, Chételat D, Ferroni N, et al. Exact combinatorial optimization with graph convolutional neural networks. In: Proceedings of the 33rd Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 32. La Jolla: NIPS, 2019, 1–12

26 Glover F, Laguna M. Tabu Search. New York: Springer, 1998

27 Goodfellow I, Bengio Y, Courville A. Deep Learning. Cambridge: MIT press, 2016

28 Goto H, Endo K, Suzuki M, et al. High-performance combinatorial optimization based on classical mechanics. Sci Adv, 2021, 7: eabe7953

29 Graikos A, Malkin N, Jojic N, et al. Diffusion models as plug-and-play priors. In: Proceedings of the 36th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 35. La Jolla: NIPS, 2022, 14715–14728

30 Guo P-F, Chen Y-H, Tsai Y-D, et al. Towards optimizing with large language models. arXiv:231005204, 2023

31 Guo T, Han C, Tang S, et al. Solving combinatorial problems with machine learning methods. In: Nonlinear Combinatorial Optimization. Springer Optimization and Its Applications, vol. 147. Cham: Springer, 2019, 207–229

32 Guo W, Zhen H-L, Li X, et al. Machine learning methods in solving the Boolean satisfiability problem. Mach Intell Res, 2023, 20: 640–655

33 Gupta P, Gasse M, Khalil E, et al. Hybrid models for learning to branch. In: Proceedings of the 34th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 33. La Jolla: NIPS, 2020, 18087–18097

34 Hamerly R, Inagaki T, McMahon P L, et al. Experimental investigation of performance differences between coherent Ising machines and a quantum annealer. Sci Adv, 2019, 5: eaau0823

35 Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. In: Proceedings of the 31st Annual Conference on Neural Information Processing Systems. Advances in neural information processing systems, vol. 30. La Jolla: NIPS, 2017, 1025–1035

36 Helsgaun K. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems. Technical Report. Roskilde: Roskilde University, 2017

37 Ho J, Jain A, Abbeel P. Denoising diffusion probabilistic models. In: Advances in Neural Information Processing Systems, vol. 33. La Jolla: NIPS, 2020, 6840–6851

38 Hopfield J J, Tank D W. "Neural" computation of decisions in optimization problems. Biol Cybernet, 1985, 52: 141–152

39 Hospedales T, Antoniou A, Micaelli P, et al. Meta-learning in neural networks: A survey. IEEE Trans Pattern Anal Mach Intell, 2021, 44: 5149–5169

40 Hottung A, Kwon Y-D, Tierney K. Efficient active search for combinatorial optimization problems. arXiv:2106.05126, 2021

41 Hou Q, Yang J, Su Y, et al. Generalize learned heuristics to solve large-scale vehicle routing problems in real-time.

In: The Eleventh International Conference on Learning Representations, https://openreview.net/pdf/1360725553e9aebf2b149f234ac6d83c46a077d4.pdf, 2023

42 Huang L, Chen X, Huo W, et al. Branch and bound in mixed integer linear programming problems: A survey of techniques and trends. arXiv:211106257, 2021

43 Hudson B, Li Q, Malencia M, et al. Graph neural network guided local search for the traveling salesperson problem. arXiv:2110.05291, 2021

44 James J, Yu W, Gu J. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning. IEEE Trans Intell Transp Syst, 2019, 20: 3806–3817

45 Jiang Y, Wu Y, Cao Z, et al. Learning to solve routing problems via distributionally robust optimization. In: Proceedings of the 36th AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence, vol. 36. Palo Alto: AAAI Press, 2022, 9786–9794

46 Joshi C K, Cappart Q, Rousseau L-M, et al. Learning the travelling salesperson problem requires rethinking generalization. Constraints, 2022, 27: 70–98

47 Joshi C K, Laurent T, Bresson X. On learning paradigms for the travelling salesman problem. arXiv:191007210, 2019

48 Joshi C K, Laurent T, Bresson X. An efficient graph convolutional network technique for the travelling salesman problem. arXiv:190601227, 2019

49 Kennedy J, Eberhart R. Particle swarm optimization. In: Proceedings of the IEEE International Conferences on Neural Networks, vol. 4. New York: IEEE, 1995, 1942–1948

50 Kim M, Park J. Learning collaborative policies to solve NP-hard routing problems. In: Proceedings of the 35th Annual Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 34. La Jolla: NIPS, 2021, 10418–10430

51 Kim M, Park J, Park J. Sym-NCO: Leveraging symmetricity for neural combinatorial optimization. In: Proceedings of the 36th Conference on Neural Information Processing Systems Advances in Neural Information Processing Systems, vol. 35. La Jolla: NIPS, 2022, 1936–1949

52 Kingma D P, Ba J. Adam: A method for stochastic optimization. arXiv:1412.6980, 2014

53 Kipf T N, Welling M. Semi-supervised classification with graph convolutional networks. arXiv:1609.02907, 2016

54 Kirkpatrick S, Gelatt C D Jr, Vecchi M P. Optimization by simulated annealing. Science, 1983, 220: 671–680

55 Kool W, Van Hoof H, Welling M. Attention, learn to solve routing problems! arXiv:1803.08475, 2018

56 Korte B H, Vygen J, Korte B, et al. Combinatorial Optimization. New York: Springer, 2011

57 Kurin V, Godil S, Whiteson S, et al. Can q-learning with graph networks learn a generalizable branching heuristic for a SAT solver? In: Proceedings of the 34th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 33. La Jolla: NIPS, 2020, 9608–9621

58 Kwon Y-D, Choo J, Kim B, et al. POMO: Policy optimization with multiple optima for reinforcement learning. In: Proceedings of the 34th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 33. La Jolla: NIPS, 2020, 21188–21198

59 Lawler E L, Wood D E. Branch-and-bound methods: A survey. Oper Res, 1966, 14: 699–719

60 Lechner M, Hasani R, Amini A, et al. Neural circuit policies enabling auditable autonomy. Nat Mach Intell, 2020, 2: 642–652

61 Lemos H, Prates M, Avelar P, et al. Graph colouring meets deep learning: Effective graph neural network models for combinatorial problems. In: Proceedings of the IEEE 31st International Conference on Tools with Artificial Intelligence. Los Alamitos: IEEE, 2019: 879–885

62 Leppek K, Byeon G W, Kladwang W, et al. Combinatorial optimization of mRNA structure, stability, and translation for RNA-based therapeutics. Nat Commun, 2022, 13: 1536

63 Li J, Ma Y, Gao R, et al. Deep reinforcement learning for solving the heterogeneous capacitated vehicle routing problem. IEEE Trans Cybern, 2021, 52: 13572–13585

64 Li K, Zhang T, Wang R. Deep reinforcement learning for multiobjective optimization. IEEE Trans Cybern, 2020, 51: 3103–3114

65 Li K, Zhang T, Wang R, et al. Research reviews of combinatorial optimization methods based on deep reinforcement learning. Acta Autom Sin, 2021, 47: 2521–2537

66 Li S, Yan Z, Wu C. Learning to delegate for large-scale vehicle routing. In: Proceedings of the 35th Annual Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 34. La Jolla: NIPS, 2021, 26198–26211

67 Li W, Li R, Ma Y, et al. Rethinking graph neural networks for the graph coloring problem. arXiv:220806975, 2022

68 Li Y, Guo J, Wang R, et al. T2T: From distribution learning in training to gradient search in testing for combinatorial optimization. In: Proceedings of the 37th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems. La Jolla: NIPS, https://github.com/Thinklab-SJTU/T2TCO, 2023

69 Li Z, Chen Q, Koltun V. Combinatorial optimization with graph convolutional networks and guided tree search. In: Proceedings of the 32nd Conference on Neural Information Processing Systems. Advances in Neural Information

Processing Systems, vol. 31. La Jolla: NIPS, 2018, 537–546

70 Li Z, Si X. NSNet: A general neural probabilistic framework for satisfiability problems. In: Proceedings of the 36th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 35. La Jolla: NIPS, 2022, 25573–25585

71 Liu F, Tong X, Yuan M, et al. Algorithm evolution using large language model. arXiv:231115249, 2023

72 Liu S, Chen C, Qu X, et al. Large language models as evolutionary optimizers. arXiv:231019046, 2023

73 Liu S, Zhang Y, Tang K, et al. How good is neural combinatorial optimization? A systematic evaluation on the traveling salesman problem. IEEE Comput Intell Mag, 2023, 18: 14–28

74 Lu H, Zhang X, Yang S. A learning-based iterative method for solving vehicle routing problems. In: International Conference on Learning Representations, https://openreview.net/attachment?id=BJe1334YDH&name=original_pdf, 2019

75 Luo F, Lin X, Liu F, et al. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In: Proceedings of the 37th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 36. La Jolla: NIPS, 2023, 8845–8864

76 Ma Q, Ge S, He D, et al. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning. arXiv:191104936, 2019

77 Ma Y, Li J, Cao Z, et al. Learning to iteratively solve routing problems with dual-aspect collaborative transformer. In: Proceedings of the 35th Annual Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 34. La Jolla: NIPS, 2021, 11096–11107

78 Manchanda S, Mittal A, Dhawan A, et al. Learning heuristics over large graphs via deep reinforcement learning. In: Advances in Neural Information Processing Systems. La Jolla: NIPS, arXiv:1903.03332, 2020

79 Matsumoto G, Aihara K, Hanyu Y, et al. Chaos and phase locking in normal squid axons. Phys Lett A, 1987, 123: 162–166

80 Mazyavkina N, Sviridov S, Ivanov S, et al. Reinforcement learning for combinatorial optimization: A survey. Comput Oper Res, 2021, 134: 105400

81 McMahon P L, Marandi A, Haribara Y, et al. A fully programmable 100-spin coherent Ising machine with all-to-all connections. Science, 2016, 354: 614–617

82 Mitchell M. An Introduction to Genetic Algorithms. Cambridge: MIT press, 1998

83 Mnih V, Kavukcuoglu K, Silver D, et al. Human-level control through deep reinforcement learning. Nature, 2015, 518: 529–533

84 Nair V, Bartunov S, Gimeno F, et al. Solving mixed integer programs using neural networks. arXiv:201213349, 2020

85 Naseri G, Koffas M A. Application of combinatorial optimization strategies in synthetic biology. Nat Commun, 2020, 11: 2446

86 Nazari M, Oroojlooy A, Snyder L, et al. Reinforcement learning for solving the vehicle routing problem. In: Proceedings of the 32nd Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 31. La Jolla: NIPS, arXiv:1802.04240, 2018

87 Nowak A, Villar S, Bandeira A S, et al. A note on learning algorithms for quadratic assignment with graph neural networks. Stat, 2017, 1050: 22

88 Optimization G. Gurobi optimizer reference manual. https://docs.gurobi.com/current/#refman/index.html,aufgerufen%20am%2027.10.20202020, 2020

89 Pan X, Jin Y, Ding Y, et al. H-TSP: Hierarchically solving the large-scale traveling salesman problem. In: Proceedings of the 37th AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence, vol. 37. Palo Alto: AAAI Press, 2023, 9345–9353

90 Paulus M B, Zarpellon G, Krause A, et al. Learning to cut by looking ahead: Cutting plane selection via imitation learning. In: Proceedings of the 39th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 162. San Diego: JMLR, 2022, 17584–17600

91 Peng H, Chen P, Yang N, et al. One-core neuron deep learning for time series prediction. Nat Sci Rev, 2025, 12: nwae441

92 Popel M, Tomkova M, Tomek J, et al. Transforming machine translation: A deep learning system reaches news translation quality comparable to human professionals. Nat Commun, 2020, 11: 4381

93 Prates M, Avelar P H C, Lemos H, et al. Learning to solve NP-complete problems: A graph neural network for decision TSP. In: Proceedings of the 33rd AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence, vol. 33. Palo Alto: AAAI Press, 2019, 4731–4738

94 Qiu R, Sun Z, Yang Y. Dimes: A differentiable meta solver for combinatorial optimization problems. In: Proceedings of the 36th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 35. La Jolla: NIPS, 2022, 25531–25546

95 Ribeiro A H, Ribeiro M H, Paixão G M M, et al. Automatic diagnosis of the 12-lead ECG using a deep neural network. Nat Commun, 2020, 11: 1760

96 Romera-Paredes B, Barekatain M, Novikov A, et al. Mathematical discoveries from program search with large language models. Nature, 2024, 625: 468–475

97 Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors. Nature, 1986, 323: 533–536

98 Schuetz M J A, Brubaker J K, Katzgraber H G. Combinatorial optimization with physics-inspired graph neural networks. Nat Mach Intell, 2022, 4: 367–377

99 Schuetz M J A, Brubaker J K, Zhu Z, et al. Graph coloring with physics-inspired graph neural networks. Phy Rev Res, 2022, 4: 043131

100 Schulman J, Wolski F, Dhariwal P, et al. Proximal policy optimization algorithms. arXiv:170706347, 2017

101 Selsam D, Lamm M, Bünz B, et al. Learning a SAT solver from single-bit supervision. arXiv:1802.03685, 2018

102 Senior A W, Evans R, Jumper J, et al. Improved protein structure prediction using potentials from deep learning. Nature, 2020, 577: 706–710

103 Silver D, Schrittwieser J, Simonyan K, et al. Mastering the game of Go without human knowledge. Nature, 2017, 550: 354–359

104 Skarda C A, Freeman W J. How brains make chaos in order to make sense of the world. Behav Brain Sci, 1987, 10: 161–173

105 Son J, Kim M, Kim H, et al. Meta-sage: Scale meta-learning scheduled adaptation with guided exploration for mitigating scale shift on combinatorial optimization. In: Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202. San Diego: JMLR, 2023, 32194–32210

106 Song J, Lanka R, Zhao A, et al. Learning to search via self-imitation with application to risk-aware planning. In: Advances in Neural Information Processing Systems. La Jolla: NIPS, arXiv:1804.00846, 2017

107 Sun Z, Yang Y. Difusco: Graph-based diffusion solvers for combinatorial optimization. In: Proceedings of the 37th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 36. La Jolla: NIPS, 2023, 3706–3731

108 Sutskever I, Martens J, Dahl G, et al. On the importance of initialization and momentum in deep learning. In: Proceedings of the 30th International Conference on International Conference on Machine Learning. International Conference on Machine Learning, vol. 28. San Diego: JMLR, 2013, 1139–1147

109 Sutton R S. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In: Proceedings of the 9th Annual Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 8. Cambridge: MIT Press, 1996, 1038–1044

110 Sutton R S, Barto A G. Reinforcement Learning: An Introduction. Cambridge: MIT Press, 2018

111 Sutton R S, McAllester D, Singh S, et al. Policy gradient methods for reinforcement learning with function approximation. In: Proceedings of the 13th Annual Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 12. Cambridge: MIT Press, 1999, 1057–1063

112 Tang Y, Agrawal S, Faenza Y. Reinforcement learning for integer programming: Learning to cut. In: International Conference on Machine Learning, San Diego: JMLR, 2020, 9367–9376

113 Tao P, Cheng J, Chen L. Brain-inspired chaotic backpropagation for MLP. Neural Netw, 2022, 155: 1–13

114 Tao P, Aihara K, Chen L. Brain-inspired chaotic graph backpropagation for large-scale combinatorial optimization. arXiv:2412.09860, 2024

115 Theodoris C V, Xiao L, Chopra A, et al. Transfer learning enables predictions in network biology. Nature, 2023, 618: 616–624

116 Toenshoff J, Ritzert M, Wolf H, et al. Graph neural networks for maximum constraint satisfaction. Front Artif Intell, 2021, 3: 580607

117 Tokuda I, Aihara K, Nagashima T. Adaptive annealing for chaotic optimization. Phys Rev E, 1998, 58: 5157

118 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of the 31st Annual Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 30. La Jolla: NIPS, 2017, 5998–6008

119 Vazirani V V. Approximation Algorithms. New York: Springer, 2001

120 Vesselinova N, Steinert R, Perez-Ramirez D F, et al. Learning combinatorial optimization on graphs: A survey with applications to networking. IEEE Access, 2020, 8: 120388–120416

121 Vinyals O, Fortunato M, Jaitly N. Pointer networks. In: Proceedings of the 29th Annual Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 28. La Jolla: NIPS, 2015, 2692–2700

122 Wang L, Li S, Tian F, et al. A noisy chaotic neural network for solving combinatorial optimization problems: Stochastic chaotic simulated annealing. IEEE Trans Syst Man Cybern Part B-Cybern, 2004, 34: 2119–2125

123 Wang Q, Lai K H, Tang C. Solving combinatorial optimization problems over graphs with BERT-based deep reinforcement learning. Inform Sci, 2023, 619: 930–946

124 Wang Z, Li X, Wang J, et al. Learning cut selection for mixed-integer linear programming via hierarchical sequence

model. arXiv:230200244, 2023

125 Wang Z, Tao P, Chen L. Brain-inspired chaotic spiking backpropagation. Nat Sci Rev, 2024, 11: nwae037

126 Williams R J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. Mach Learn, 1992, 8: 229–256

127 Williamson D P, Shmoys D B. The Design of Approximation Algorithms. Cambridge: Cambridge Univ Press, 2011

128 Wu Y, Song W, Cao Z, et al. Learning improvement heuristics for solving routing problems. IEEE Trans Neural Netw Learn Syst, 2021, 33: 5057–5069

129 Wu Z, Pan S, Chen F, et al. A comprehensive survey on graph neural networks. IEEE Trans Neural Netw Learn Syst, 2020, 32: 4–24

130 Xin L, Song W, Cao Z, et al. Neurolkh: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem. In: Proceedings of the 35th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 34. La Jolla: NIPS, 2021, 7472–7483

131 Yang C, Wang X, Lu Y, et al. Large language models as optimizers. arXiv:230903409, 2023

132 Yao W, Bandeira A S, Villar S. Experimental performance of graph neural networks on random instances of max-cut. In: Wavelets and Sparsity XVIII, vol. 11138. Bellingham: SPIE-INT Soc Optical Engineering, 2019, 242–251

133 Ye H, Wang J, Cao Z, et al. DeepACO: Neural-enhanced ant systems for combinatorial optimization. In: Proceedings of the 37th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 36. La Jolla: NIPS, https://proceedings.neurips.cc/paper_files/paper/2023/hash/883105b282fe15275991b411e6b200c5-Abstract-Conference.html, 2023

134 Yehuda G, Gabel M, Schuster A. It's not what machines can learn, it's what we cannot teach. In: International Conference on Machine Learning. Ann Arbor: PMLR, 2020, 10831–10841

135 Yolcu E, Póczos B. Learning local search heuristics for Boolean satisfiability. In: Proceedings of the 33rd Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 32. La Jolla: NIPS, https://proceedings.neurips.cc/paper/2019/hash/12e59a33dea1bf0630f46edfe13d6ea2-Abstract.html, 2019

136 Zhang C, Wu Y, Ma Y, et al. A review on learning to solve combinatorial optimisation problems in manufacturing. IET CIM, 2023, 5: e12072

137 Zhang J, Liu C, Li X, et al. A survey for solving mixed integer programming via machine learning. Neurocomputing, 2023, 519: 205–217

138 Zhang W, Sun Z, Zhu Q, et al. NLocalSAT: Boosting local search with solution prediction. In: Proceedings of the 29th International Joint Conference on Artificial Intelligence. Freiburg: IJCAI-INT Joint Conf Artif Intell, 2020, 1177–1183

139 Zhang Z, Wu Z, Zhang H, et al. Meta-learning-based deep reinforcement learning for multiobjective optimization problems. IEEE Trans Neural Netw Learn Syst, 2023, 34: 7978–7991

140 Zheng J, He K, Zhou J, et al. Combining reinforcement learning with Lin-Kernighan-Helsgaun algorithm for the traveling salesman problem. In: Proceedings of the 35th AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence, vol. 35. Palo Alto: AAAI Press, 2021, 12445–12452

141 Zhou J, Wu Y, Song W, et al. Towards omni-generalizable neural methods for vehicle routing problems. In: Proceedings of the 40th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 202. San Diego: JMLR, 2023, 42769–42789

142 Zhou T, Niu P, Wang X, et al. One fits all: Power general time series analysis by pretrained LM. In: Proceedings of the 37th Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems, vol. 36. La Jolla: NIPS, arXiv:230211939, 2023