Deep Reinforcement Learning for Multi-Sequence Combinatorial Optimization Problems

Dai Zhang[1,*], Xue Zhang[2]

{zhangd@hitachi.cn[1], zhangx@hitachi.cn[2]}

Hitachi (China) Ltd., Corp, Shanghai, China[1]
Hitachi (China) Ltd., Corp, Shanghai, China[2]
*corresponding author

Abstract. We propose a formulation for solving sequential combinatorial optimization problems (COP) by deep reinforcement learning (DRL) method. Some types of decision problems can be reduced to sequence combination optimization problems(SCOP) , such as knapsack problems, flow-shop problems, etc. Besides some single sequence combinatorial optimization problems, multi-sequence combinatorial optimization problems have more complex constraints and are difficult to solve quickly in finite time. In this paper we extend a DRL model based on multiple pointer networks and policy gradient approach to solve multiple-sequence COP. We take flow-shop problem as example to verify the efficiency of improved model. As the result shows, the DRL method can calculate better results within 1 minute than some heuristic algorithms that should optimize with 0.5 hour or 1 hour time consumption. Due to its randomness heuristic optimization method can also produce better solutions, but the difference between DRL and the best heuristic results is within 5%. Therefore, compared to heuristic optimization methods, DRL has competitive optimization capabilities and significantly higher computational efficiency.

Keywords: combinatorial optimization problem, multiple-sequence COP, deep reinforcement learning.

1 Introduction

In some application scenarios, decision problems are combinatorial optimization problems (COP), and many combinatorial optimization problems are NP-hard, making it difficult to provide effective solutions within a limited time. According to the different solving objectives, combinatorial optimization problems can be categorized into several types, such as those aiming to find optimal sets, sequences, or graphs from all feasible solutions [1]. In this paper, we will focus on the COP of output sequence solutions, which we refer to as sequential combinatorial optimization problem (SCOP). SCOP is a common type of decision-making COP appearing in production, supplying and service.

In the past, traditional operational research (OR) methods and heuristic optimization methods were the primary solutions for COP. Mixed-integer linear programming (MILP) based on branch & bound and cutting plane is a common exact optimization method for small and middle scale COP. Heuristic optimization methods, characterized by the ability to find sufficiently good feasible solutions at an acceptable computational cost, can generally be divided into single-solution based algorithms and population-based algorithms [2]. However, well-designed

CONF-MLA 2024, November 21, Adana, Turkey

DOI 10.4108/eai.21-11-2024.2354636

heuristic algorithms require a significant amount of expertise and trial work, and manually designed heuristic algorithms cannot guarantee consistent performance across different problem sets. Actually, traditional methods for solving COP face significant challenges in terms of speed, accuracy, and generalization ability [3]. With the advancement of deep reinforcement learning (DRL) technology, it has demonstrated remarkable learning and decision-making capabilities in fields such as Go and robotics. Consequently, DRL is also being studied to solve COP due to the advantage that decision strategy is trained by data in a general framework rather than made by hand and it is also easy for parallel computing.

2 Related work

As a traditional operations research planning method MILP has long been used to solve certain COP. To solve large-scale COP, which are NP-hard, advanced MILP algorithms have also been studied and used. In modern MILP solvers [4][5], a branch-and-bound tree search algorithm is employed, in which a linear programming (LP) relaxation of a MILP is solved at each node. Wang and Li [6] propose a hierarchical sequence model to learn cut selection policies via reinforcement learning, thereby enhancing the efficiency of solving MILP. Huang [7] use large neighborhood search (LNS) which is a heuristic method in MILP instead of branch-and-bound to obtain a better performance on several benchmarks.

In recent past heuristic approaches have also achieved some new results in hybrid applications and in combination with machine learning. Muthuraman and Venkatesan [8] studied on hybrid heuristic methods for different COP, and conclude the advantages and disadvantages of these approaches by analyzing the current state-of-the-art and its effectiveness in solving COP. To improve the performance of classic heuristic guided by hand-crafted rules, Wu [9] use a deep reinforcement learning framework to learn the improvement heuristics and guide the iterative selection for routing problems. In Zhao's study [10], a Cooperative Multi-Stage Hyper-Heuristic (CMS-HH) algorithm is proposed to address certain COP. In the CMS-HH, a genetic algorithm is introduced to perturb the initial solution to increase the diversity of the solution. In the search phase, an online learning mechanism based on the multi-armed bandits and relay hybridization technology are proposed to improve the quality of the solution. Compared with other hyper-heuristic methods, it achieves superior results on some COP instances.

As the AI technology development, some DRL model has been studied to solve the COP with simple form. Bengio's paper [11] surveys the recent attempts from the machine learning and operations research communities at leveraging machine learning to solve COP. They think machine learning is a natural candidate to make well decisions in a more principled and optimized way. Vinyals [12] proposes a neural network model with attention mechanism named pointer networks (Ptr-Net) to solve combinatorial optimization problems. Bello [13] and Nazari [14] further follow the idea to solve standard traveling salesman (TSP) and VRP with Ptr-Net as augmentation. Zhao [15] combines a DRL model, composing of an actor, an adaptive critic and a routing simulator, with a local search method to further improve slow running time and poor solution quality for VRP.

In practical scenarios, some COP which decision consists of multiple sequences, such as job-shop problems (JSP) and staff rostering problems (SRP) are not solved well by previous DRL model due to constraint relationship between sequences. In this paper, we study DRL to solve

some multiple sequential combinatorial optimization problems (MSCOP). By this research, we expect to provide a general intelligent approach for COP with same form and different scales in most industry scenarios, which is no need to manually filter out a suitable strategy and to consume much computation resources.

3 DRL for MSCOP

Firstly, we summarize the form of MSCOP and then give the basic model to be applied.

3.1 Problem definition

SCOP exists in variety of practical scenarios of manufacturing, supply chain management and enterprise management. SCOP is defined as that the final decision solution consists of multiple sub-decision variables, which are generated sequentially in chronological order or according to other sequences. Some practical problems can be categorized as the SCOP listed in Table-1.

Table1. Sequential combinatorial optimization problems

Problems	By	Step1	Step2	...	Step N	Final Solutions
KP Knapsack	Package	Item 1	Item 3	...	Item N	1-3-2...-n
FSP / JSP Flow-shop/Job-shop Scheduling	Machine	Job 2	Job 1	...	Job N	2-1-3...-n
SRP Staff rostering	Staff	Shift 3	Shift 1	...	Shift N	3-1-2...-n
TAP Task assignment	Employee	Task 1	Task 2	...	Task N	1-2-3...-n
VRP Vehicle routing	Traveler	Node 3	Node 2	...	Node N	3-2-1...-n

In order to solve the problem gradually from easy to difficult, we divide SCOP into two types, single sequence combinatorial optimization problem (SSCOP) and multi-sequence combinatorial optimization problems (MSCOP). MSCOP is more difficult than SSCOP due to the constraint relationship between sequences and computation complexity caused by problem scale.

Then we take FSP as a complex MSCOP target, which is a common problem in manufacturing. FSP starts with a set of n jobs J_i (i= 1,…, n) and a set of m machines M_j (j=1,…, m). Each job $J_i$consists of an ordered set of m operations O_{ij} (j=1,…, m), and each operation requires to process on a different machine. In FSP, every job must be processed on all machines in the same order; however, the processing time on the same machine of different jobs may not be the same. The processing time of the operation O_{ij} is given by T_{ij}. In this problem, the processing of jobs must satisfy the sequence requirement without conflict, which means the constraint conditions existing between sequences. The objective is to find a feasible time allocation of all operations

on the given machines to minimize the total processing time span. Specifically, in the following single machine case, the objective is to minimize the total waiting time of unprocessed jobs.

3.2 DRL road-map for MSCOP

Instead of solving by making heuristic strategy for each scenario, we focus on studying DRL approaches for common SCOP in a general framework. There are typically two DRL approaches for solving the decision process. The first one is evaluation-by-step learning using DQN, PPO and other DRL approaches. These models evaluate the reward of every action that arranges item into output sequence by steps, while the reward is hard to be evaluated before the sequence is constructed. The other one is two-stage learning based on seq2seq and DRL model. In stage-1 a complete feasible solution is constructed by seq2seq model (e.g. Ptr-Net), and in stage-2 the reward of solution is evaluated and used to improve the weights of policy network (the seq2seq network). So, the two-stage learning method is similar to the local searching approach. The second approach is obviously convenient to calculate the reward when a complete feasible solution is constructed. So, we prefer to employ the second approach to solve MSCOP.

4 Method description

4.1 Pointer-Networks

Based on the previous research [8], pointer-networks (Ptr-Net) has been verified to solve the SSCOP. Given sequential input data, the Ptr-Net is capable of picking out the item from the input set step by step and generating a new output sequence. The improved Ptr-Net remains the efficiency and decreases computational complications by omitting RNN type encoder due to the no meaningful order of input data. An embedding encoder network reads through the input sequence and stores the knowledge in a fixed-size vector representation (or a sequence of vectors); then, a decoder converts the encoded information back to an output sequence after the attention layer processing. Attention mechanism in Ptr-Net is a differentiable structure for selecting different parts of the input according to the output sequence. The input sequence includes two parts, static data s^t and dynamic data d^t, which respectively describe the input task's immutable property and mutable status. In Ptr-Net, when dynamic data is updated, the embedding vectors should be updated. The self-status mask operation is applied to cut off the impossible optional items after the attention possibilities computation. Figuar 1 shows the Ptr-Net structure.

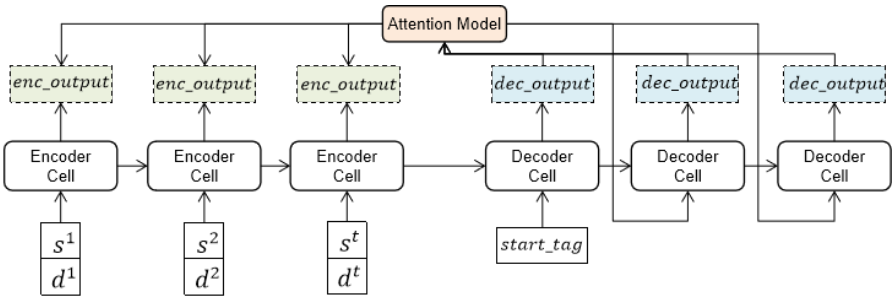


Fig. 1. Pointer-Networks structure.

4.2 Multiple Point Networks

Single Ptr-Net can only store a sequential generation policy for the problem with input and output in form of single sequence. If the problem has multiple input and output sequences, sequences with inherent differences should be generated by different calculation model with separated policy. To address the MSCOP, we extend single Ptr-Net as a multiple Ptr-Nets structure to generate sequences synchronously as Figure 2 showing. Several sequential data are input into different Ptr-Net, and the sequential indices are generated at the same pace. In this process, the change of dynamic data should be transmitted from top level to bottom level. It is necessary to adjust the optional range of nodes for output sequence based on hard constraints in the dynamic change transmitting process. The dynamic change transmitting mechanism can address the constraint conflicts between sequences. For example, in FSP, the previous process must be completed before proceeding with the next process. Otherwise, a waiting time is chosen instead of next job process.

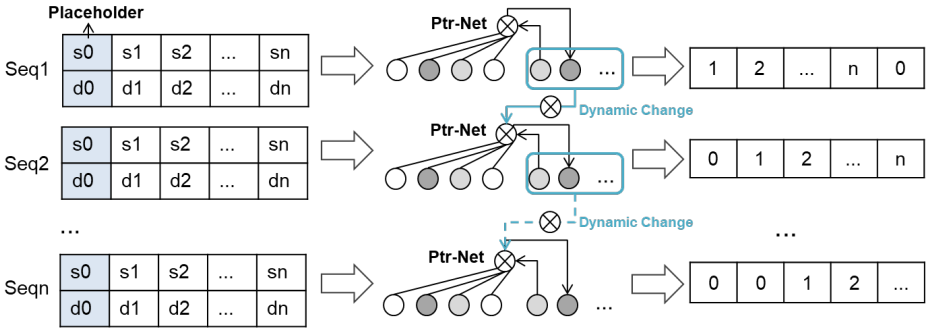


Fig. 2. Sequences generation by multiple Ptr-Nets.

4.3 Actor-Critic Model with Multiple Pointer Networks

To upgrade the sequential generation policy in multiple Ptr-Nets, policy gradient approach as a well-known reinforcement learning method is adopted. The basic two-stage DRL model framework is as follows Figure 3.

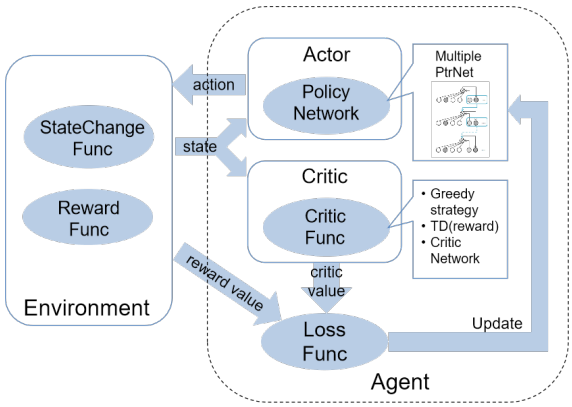


Fig. 3. DRL model framework

In the first stage, the actor policy network employs multiple Ptr-Nets to generate searching action according to the policy determined by weights parameters. This policy network is very suitable for sequential decision problem solution because of its advantage to generate node sequence by steps. How many Ptr-Nets are used as actor policy network is adaptive determined according to the problem type. The model suits for either single sequence problem or multiple sequences problem, but it does not mean the model needs no training for different problem.

In the second stage of policy learning, Actor-Critic(A2C) structure is used to simulate the neighbor searching process. The actor network that predicts a probability distribution over the next action at any given decision step, and a critic network that estimates the reward for any problem instance from a given state. By minimizing the error between reward of action and a value calculated by critic network, the policy network and critic network become more and more suitable to generate the optimal solution for the problem.

4.4 Training

When generating the feasible sequential solution using Ptr-Net with initial network parameters θ, we begin to train the network model using gradient descent. Policy gradient methods improve the policy iteratively using an estimation of the gradient which represents the expected reward change with respect to the policy parameters. We compute the corresponding rewards by default actor network. The value of critic is used as a baseline and it can be obtained in three ways: the moving average exponential of rewards, a solution reward from greedy strategy and a solution reward from full-connecting network. Then we update the policy gradient $d\theta$ to update the actor network. And we also update the critic network $d\phi$ in the direction of reducing the difference between the expected rewards with the observed ones.

Algorithm 1: Actor-Critic Training for Multiple sequences

Input: number of epochs E, steps per eporch S, batch size B.

Output: Agent model θ.

Initialize actor Ptr-Net with parameter θ and critic network with parameter ϕ;

for *epoch* = 1 → E **do**

  Reset gradients $d\theta \leftarrow 0$, $d\phi \leftarrow 0$; generating batches $\{B_1, B_2, ..., B_s\}$;

  for *step* = 1 → S **do**

    Obtain random instance batch $X_i = \{(s_i, d_i), i = 1,2 \dots B\}$;

    for *itemNum* = 1 until all items processed **do**

      for *SeqNum sn* = 1 until all sequences processed **do**

        Chose item_index y by $p_\theta(y|X)$ via $PtrNet_{sn}(\theta)$;

        Update dynamic data set of **this** sequence;

        Update dynamic data set of **next** sequence;

    Compute reward R as the last sequence performance;

Compute critic reward C via critic network with ϕ;

Compute advantage function A=R-C

$d\theta = \frac{1}{B}\sum_{i=1}^{B} A_i * \nabla_\theta \log p_\theta(Y \mid X)$

$d\phi = \frac{1}{B}\sum_{i=1}^{B} {A_i}^2$

Update θ, ϕ using $Adam(\theta, d\theta)$, $Adam(\phi, d\phi)$;

Verification on test data and persist model θ.

End

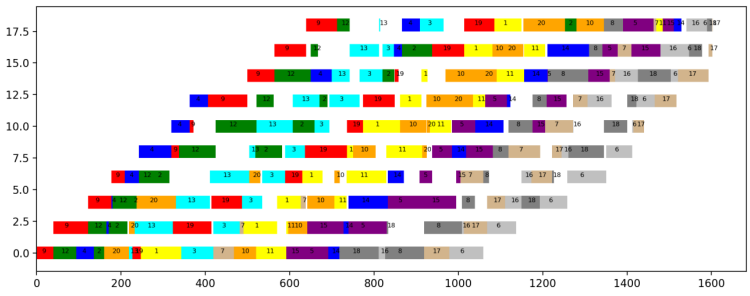
Fig. 4. Actor-Critic training process.

4.5 Verification

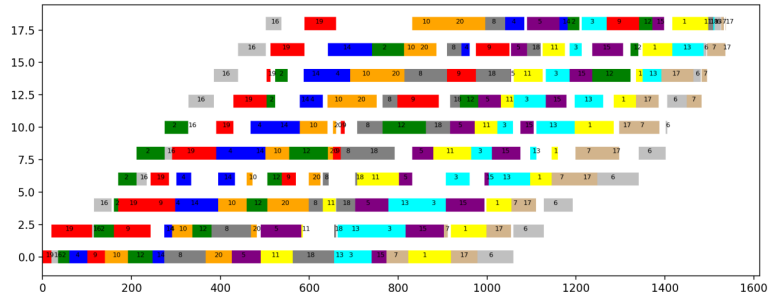
As a typical multiple sequence decision problem, FSP is solved to verify the performance of DRL approach. We trained and compared with the heuristic algorithm. We used the standard flow-shop dataset in OR-Library [16] as a test example, and compared it with four algorithms: simulated annealing (SA)[17], taboo search(TS)[18], genetic algorithm(GA)[19], and genetic algorithm with taboo search(GATS)[20]. The computer we used to run these algorithms is with Intel(R) Core (TM) i7-9700 CPU, 16.0 GB RAM and Nvida 2080Ti GPU (12GB). The multiple Ptr-Nets DRL approach is implemented with Pytorch framework.

We train DRL models for different scale FSP separately. In the training dataset, we generate dataset consisting corresponding number of jobs and machines which needs a processing time random in [0.1, 0.9], and a placeholder of waiting period with a value of 0.1. The target is to minimize the completion time of all jobs. In this research we process 100 batches of 128 instances in each epoch. And we train for 10 epochs to find the optimal model. In the training period, it will take about 3.5 hours for 20×5 data set and over 10 hours for 20×10 data set.

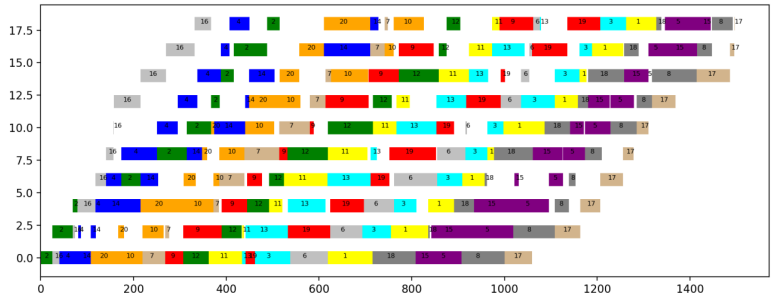
The effect of scheduling results of test data fs_reC11 by different approaches is compared as Figure 4 shows. The job index and processing time are marked on the job period. Different jobs are represented by different colors, but jobs are represented by the same color in different method result graphs. The figure clearly shows the order in which different jobs are executed on different machines, as well as the total time it takes for all jobs to be completed. The shorter the total time required to complete all jobs, the better.



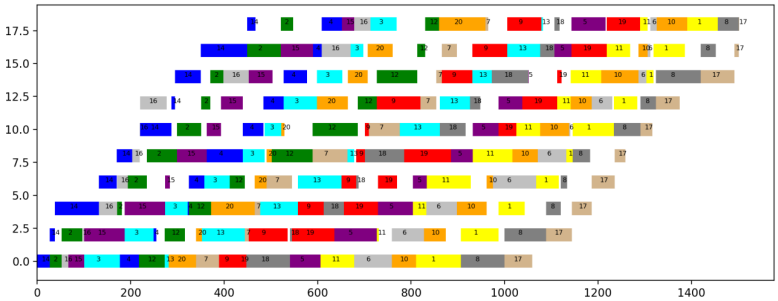
(a) SA



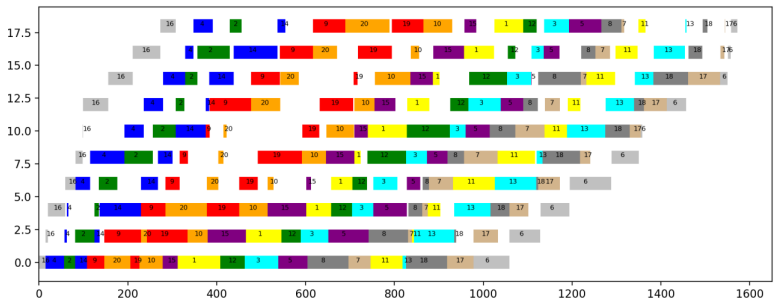
(b) TS



(c) GA



(d) GATS



(e) DRL

Fig. 5. Test results comparison by different approaches for FSP data (fs_reC11)

In this FSP case, the jobs are scheduled in the order that satisfying the intended goals. In the former machines, jobs with short processing time are listed at first to save other jobs' waiting time or to make the machine behind work as soon as possible. We can see the scheduled result of DRL has earlier starting in all of 10 machines than other methods. If the job whose processing time in former machine longer than that in latter machine is listed at first, the worse situation occurs and waiting periods appears in the job queue. In Figure 5, DRL result still exists more waiting periods due to the training is not enough.

Table 2 shows the comparison result between DRL and several heuristic algorithms, and DRL has stronger potential in solving SCOP. We have separately listed the results of four heuristic methods under two different iteration running periods (0.5hour and 1hour). We also calculated the running time and results of DRL inferring calculation (resA) on the same datasets after training. We can see the DRL result resA is better than some of results of heurisitic method calculated in both short period and long period, but it is also a little worse compared with GA random optimization results. To evaluate the difference in results, we also calculated a boundary value of 5% improvement in DRL result as resB. From Table 2, we can see that the boundary value resB is better than the majority of heuristic optimization results. Meanwhile, the time consumption of DRL has greatly improved compared to heuristic algorithms, reaching within 1 minute. Therefore, we can conclude that the DRL method can calculate better results within 1 minute than some heuristic algorithms that should optimize with 0.5 hour or 1 hour time consumption. Due to its randomness heuristic optimization method can also produce better solutions, but the difference between DRL and the best heuristic results is only within 5%.

Table 2. Comparison with heuristic methods (less is better).

Dataset	SA		TS		GA		GATS		DRL		DRL Use Time
	0.5h	1h	0.5h	1h	0.5h	1h	0.5h	1h	resA	resB	
fs_reC3 (20x5)	1187	1167	1144	1135	1105	1126	1138	1111	1135	1078	6.15s
fs_reC5 (20x5)	1278	1273	1262	1251	1257	1264	1243	1253	1310	1245	10.51s
fs_reC7 (20x10)	1768	1780	1640	1646	1609	1647	1697	1646	1717	1631	52.89s
fs_reC9 (20x10)	1695	1659	1652	1651	1553	1638	1614	1666	1707	1622	58.66s
fs_reC11 (20x10)	1635	1603	1555	1537	1517	1496	1506	1502	1572	1493	50.02s

5 Conclusions

In this paper, we presented a DRL model consisting of multiple Ptr-Nets and A2C learning framework to solve MSCOP. The model can solve both SSCOP (e.g. KP) and MSCOP (e.g. FSP). Through our test, the DRL model has success to optimize multiple sequences FSP. Compared to the traditional heuristic methods, DRL method has better generalization and scalability to solve different MSCOP by adjusting the constraint relationship between sequences.

At present, the small basic DRL optimization model of multiple sequences decision problem has been developed and verified in single GPU platform. Some works to improve the efficiency

and feasibility of model should be further researched and updated in future. As the scale of the problem increases, the computation resource consumption should be optimized. And distribution training is another way to solve the large-scale multiple sequences problem. In some sequential problems in practical, the optimization target is scored several criteria, and the constraints between sequences may be more complex. We should verify the model performance and difficulties when addressing these complicated problems.

References

[1] Korte, B, Vygen, J. (2011). Combinatorial Optimization Theory and Algorthms. Springer.

[2] Talbi, E. G. (2009). Metaheuristics: From Design to Implementation. John Wiley & Sons.

[3] Wang, Y., Chen, Z. B., Wu, Z. R. (2022). Review of Reinforcement Learning for Combinatorial Optimization Problem. Journal of Frontiers of Computer Science and Technology, 16(2), 261-279.

[4] Gurobi. Gurobi solver, https://www.gurobi.com/, 2021.

[5] Bestuzheva, K., Besançon, M., Chen, W. K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., ... & Witzig, J. (2021). The SCIP optimization suite 8.0. arXiv preprint arXiv:2112.08872.

[6] Wang, Z., Li, X., Wang, J., Kuang, Y., Yuan, M., Zeng, J., ... & Wu, F. (2023). Learning cut selection for mixed-integer linear programming via hierarchical sequence model. arXiv preprint arXiv:2302.00244.

[7] Huang, T., Ferber, A. M., Tian, Y., Dilkina, B., & Steiner, B. (2023). Searching large neighborhoods for integer linear programs with contrastive learning. In International Conference on Machine Learning (pp. 13869-13890). PMLR.

[8] Muthuraman, S., & Venkatesan, V. P. (2017). A comprehensive study on hybrid meta-heuristic approaches used for solving combinatorial optimization problems. In 2017 world congress on computing and communication technologies (WCCCT) (pp. 185-190). IEEE.

[9] Wu, Y., Song, W., Cao, Z., Zhang, J., & Lim, A. (2021). Learning improvement heuristics for solving routing problems. IEEE transactions on neural networks and learning systems, 33(9), 5057-5069.

[10] Zhao, F., Di, S., Cao, J., & Tang, J. (2021). A novel cooperative multi-stage hyper-heuristic for combination optimization problems. Complex System Modeling and Simulation, 1(2), 91-108.

[11] Bengio, Y., Lodi, A., & Prouvost, A. (2021). Machine learning for combinatorial optimization: a methodological tour d'horizon. European Journal of Operational Research, 290(2), 405-421.

[12] Vinyals, O. , Fortunato, M. , Jaitly, N. . (2015). Pointer networks. Computer Science, 28.

[13] Bello, I. , Pham, H. , Le, Q. V. , Norouzi, M. , Bengio, S. . (2016). Neural combinatorial optimization with reinforcement learning.

[14] Nazari, M. , Oroojlooy, A. , Snyder, L. V. , Takáč, Martin. (2018). Reinforcement learning for solving the vehicle routing problem.

[15] Zhao, J., Mao, M., Zhao, X., & Zou, J. (2020). A hybrid of deep reinforcement learning and local search for the vehicle routing problems. IEEE Transactions on Intelligent Transportation Systems, 22(11), 7208-7218.

[16] Beasley, J. E. OR-Library: distributing test problems by electronic mail. Journal of the Operational Research Society, 1990. 41(11), 1069-1072. http://people.brunel.ac.uk/~mastjjb/jeb/info.html.

[17] Jeffcoat, D. E. , Bulfin, R. L. . (1993). Simulated annealing for resource-constrained scheduling. European Journal of Operational Research, 70(1), 43-51.

[18] Zhang, K. Xu, J. Geng, X.T. (2008). Improved tabu search algorithm for designing DNA sequences. Progress in Natural Science, vol.18, 623-627.
[19] Park, B. J. , Choi, H. R. , Kim, H. S. . (2003). A hybrid genetic algorithm for the job shop scheduling problems. Computers & Industrial Engineering, 45(4), 597-613.
[20] Zhang, X. , Zhang, D. , Hu, X. . (2021). A novel optimization algorithm for job-shop production scheduling. IEEE.