分类号 O221.7 学号 19069005

UDC 519.8 密级 公开

工学博士学位论文

基于深度强化学习的组合优化方法研究

博士生姓名 李凯文

学科专业 管理科学与工程

研究方向 计算智能与优化决策技术

指导教师 张涛 教授

协助指导教师 王锐 副研究员

国防科技大学研究生院

二〇二二年六月

Researches on Combinatorial Optimization Methods in Basis of Deep Reinforcement Learning

Candidate: Li Kaiwen

Supervisor: Prof. Zhang Tao

Associate Supervisor: Associate Prof. Wang Rui

A dissertation

Submitted in partial fulfillment of the requirements

for the degree of Ph.D of Engineering

in Management Science and Engineering

Graduate School of National University of Defense Technology

Changsha, Hunan, P. R. China

June, 2022

目 录

表 目 录

图 目 录

摘 要

组合优化问题广泛存在于国防、交通、工业、通信等各个领域，几十年来，传统运筹优化方法，例如分支定界法、启发式搜索算法、邻域搜索算法等，是解决组合优化问题的主要手段，但在实际应用中，例如在军事作战调度、网络资源分配、网约车订单派送等复杂优化场景中，优化问题的规模不断扩大，对在线优化的需求越来越高，传统组合优化算法面临着很大的计算压力，很难实现组合优化问题的在线求解和快速决策，急需提出新的突破性的方法克服传统组合优化方法的局限。

近年来，随着深度学习技术的迅猛发展，深度强化学习在围棋、机器人等领域的瞩目成果显示了其强大的学习能力与序贯决策能力，通过深度神经网络模型离线训练、在线优化的特性，可以有效学习问题的特征，实现问题的快速决策，从而为组合优化问题的快速求解提供了新的可能性。基于此，论文对基于深度强化学习的组合优化方法进行研究，提出基于深度强化学习的组合优化框架，从组合优化的单目标方法、多目标方法、分支定界法几个方面开展研究，主要贡献如下：

（1）提出了基于深度强化学习的组合优化框架。论文将深度强化学习方法求解组合优化问题的流程分为三步：根据问题类型进行模型选择；设计深度神经网络对组合优化问题进行建模；采用强化学习方法对深度神经网络参数进行学习。并针对各个步骤，对涉及到的深度神经网络模型、深度强化学习方法的相关理论进行了阐述，详细研究了基于深度神经网络的组合优化问题建模方法，理清了基于深度强化学习求解组合优化问题的科学原理，提出了基于深度强化学习的组合优化框架和求解范式。

（2）针对复杂单目标组合优化问题，提出了一种基于深度强化学习的覆盖旅行商组合优化方法。当前基于人工智能的组合优化方法大多应用于较为简单的组合优化问题，对于复杂大规模组合优化问题的研究较少，本文对具有复杂动态特性的覆盖旅行商问题进行了研究，提出了一种基于深度强化学习的覆盖旅行商问题求解方法，设计了一种基于动态嵌入的注意力模型，克服了当前人工智能模型在处理大规模、动态复杂特性问题上的局限，所提方法相对于传统启发式方法，在达到相同优化水平的前提下，能够获得 10 倍以上求解速度的提升，并且模型能够泛化到不同类型的覆盖旅行商问题上。

（3）针对多目标组合优化问题，提出了一种基于深度强化学习的多目标旅行商组合优化方法。当前基于人工智能的组合优化方法均针对单目标问题进行研究，

在多目标问题上仍然存在空白，论文首次采用深度强化学习方法对多目标组合优化问题进行了研究，设计了一种分解策略和基于邻域的参数迁移策略，采用深度神经网络对问题进行建模，采用端到端方式直接输出问题的帕累托前沿，从而实现对多目标组合优化问题的高效求解，所提方法在多达 500 个节点、多达 5 个优化目标的多目标旅行商问题上取得了超越传统方法的性能，具有快速求解能力和泛化能力。

（4）针对组合优化的精确求解算法，提出了一种基于深度强化学习的分支定界组合优化方法。分支定界法能够得到组合优化问题的理论最优解，但是传统分支定界法的核心缺陷是其极慢的求解速度，论文采用人工智能技术对传统分支定界法进行改进，设计了一种图指针神经网络模型对分支策略进行建模，对问题的全局特征和历史特征进行提取，并设计了一种模仿学习方法对模型进行训练，求解速度相对于传统分支定界法提升 40%，在多种不同组合优化问题上的表现均超越了当前基于机器学习改进的分支定界法。

（5）最后，采用深度强化学习优化方法对实际应用问题进行了研究，考虑了一种基于无线能量传输的无人机路径规划任务场景，通过无线能量传输技术对无人机进行无线充电，分别建立了无人机功耗模型、无线能量传输模型、以及深度神经网络模型，通过离线训练、在线优化的模式实现了无人机路径的在线优化，相对于粒子群、Google OR-tools 等传统优化方法，在不同规模问题上实现了 5 到 500 倍求解速度的提升。

关键词: **组合优化；强化学习；深度神经网络；人工智能；多目标优化；分支定界法；指针网络；图神经网络**

Abstract

Combinatorial optimization problems widely exist in various fields such as national defense, transportation, industry and communication. For decades, traditional operations research optimization methods, such as branch and bound method, heuristic search algorithm and neighborhood search algorithm, are the main means to solve combinatorial optimization problems. However, in practical applications, such as military operation scheduling, network resource allocation and online car hailing, the scale of optimization problems is expanding, and the demand for online optimization is higher and higher. The traditional combinatorial optimization algorithm is facing great computational pressure, which is difficult to realize online solution and rapid decision-making of combinatorial optimization problems. It is urgent to propose new breakthrough methods to overcome the limitations of traditional combinatorial optimization methods.

In recent years, with the rapid development of deep learning technology, the remarkable achievements of deep reinforcement learning in the fields of go and robot show its strong learning ability and sequential decision-making ability. Through the characteristics of off-line training and on-line optimization of deep neural network model, it can effectively learn the characteristics of the problem and realize the rapid decision-making of the problem, Thus, it provides a new possibility for the rapid solution of combinatorial optimization problems. Based on this, this paper studies the combinatorial optimization method based on deep reinforcement learning, puts forward the combinatorial optimization framework based on deep reinforcement learning, and carries out research from the aspects of single objective method, multi-objective method and branch and bound method of combinatorial optimization. The main contributions are as follows:

(1) A combinatorial optimization framework based on deep reinforcement learning is proposed. In this paper, the process of solving combinatorial optimization problems by deep reinforcement learning method is divided into three steps: model selection according to the type of problem; The deep neural network is designed to model the combinatorial optimization problem; Reinforcement learning method is used to learn the parameters of deep neural network. For each step, the related theories of deep neural network model and deep reinforcement learning method are described, the modeling method of combinatorial optimization problem based on deep neural network is studied in detail, the scientific prin-

ciple of solving combinatorial optimization problem based on deep reinforcement learning is clarified, and the combinatorial optimization framework and solution paradigm based on deep reinforcement learning are proposed.

(2) Aiming at the complex single objective combinatorial optimization problem, a covering traveling salesman combinatorial optimization method based on deep reinforcement learning is proposed. At present, the combinatorial optimization methods based on artificial intelligence are mostly applied to relatively simple combinatorial optimization problems, and there is little research on complex large-scale combinatorial optimization problems. In this paper, the covering traveling salesman problem with complex dynamic characteristics is studied, and a solution method of covering traveling salesman problem based on deep reinforcement learning is proposed, An attention model based on dynamic embedding is designed, which overcomes the limitations of the current artificial intelligence model in dealing with large-scale and dynamic complex problems. Compared with the traditional heuristic methods, the proposed method can improve the solution speed by more than 10 times on the premise of reaching the same optimization level, And the model can be generalized to different types of covering traveling salesman problems.

(3) Aiming at the multi-objective combinatorial optimization problem, a multi-objective traveling salesman combinatorial optimization method based on deep reinforcement learning is proposed. At present, the combinatorial optimization methods based on artificial intelligence are studied for single objective problems, and there is still a gap in multi-objective problems. In this paper, the deep reinforcement learning method is used to study the multi-objective combinatorial optimization problem for the first time, a decomposition strategy and a parameter transfer strategy based on neighborhood are designed, and the deep neural network is used to model the problem, The Pareto frontier of the problem is directly output in an end-to-end manner, so as to realize the efficient solution of the multi-objective combinatorial optimization problem. The proposed method outperforms the traditional methods on the multi-objective traveling salesman problem with up to 500 nodes and up to 5 optimization objectives, and has the ability of fast solution and generalization.

(4) Aiming at the exact algorithms for combinatorial optimization, a branch and bound combinatorial optimization method based on deep reinforcement learning is proposed. The branch and bound method can obtain the theoretical optimal solution of the combinatorial optimization problem, but the core defect of the traditional branch and

bound method is its extremely slow solution speed. This paper uses artificial intelligence technology to improve the traditional branch and bound method, designs a graph pointer neural network model to model the branch strategy, and extracts the global and historical features of the problem, A training method based on the imitation learning is designed, which realizes the optimization performance 40 % faster than the professional optimization solver. Its performance on three different combinatorial optimization problems exceeds the current branch and bound method based on machine learning, and realizes the effective improvement of the solution speed.

(5) Finally, the practical application problems are studied based on the deep reinforcement learning optimization method. A UAV path planning task scenario based on wireless energy transmission is considered. The UAV is charged wirelessly through wireless energy transmission technology. The UAV power consumption model, wireless energy transmission model and deep neural network model are established respectively through off-line training The online optimization mode realizes the online optimization of UAV path. Compared with traditional optimization methods such as particle swarm optimization and Google or tools, it improves the solution speed by 5 to 500 times on different scale problems.

Key Words: Combinatorial Optimization, Reinforcement Learning, Deep Neural Network, Artificial Intelligence, Multi-objective Optimization, Branch and Bound, Pointer Network, Graph Neural Network

第一章　绪论

1.1　研究背景

组合优化问题广泛存在于国防、交通、工业、生活等各个领域，几十年来，传统运筹优化方法是解决组合优化问题的主要手段，但随着实际应用中问题规模的不断扩大、求解实时性的要求越来越高，传统运筹优化算法面临着很大的计算压力，很难实现组合优化问题的在线求解。近年来随着深度学习技术的迅猛发展，深度强化学习在围棋、机器人等领域的瞩目成果显示了其强大的学习能力与序贯决策能力。鉴于此，近年来涌现出了多个利用深度强化学习方法解决组合优化问题的新方法，具有求解速度快、模型泛化能力强的优势，为组合优化问题的求解提供了一种全新的思路。

组合优化问题（Combinatorial optimization problem，COP）是一类在离散状态下求极值的最优化问题，其数学模型如下所示：

$$\begin{aligned} &\min F(x) \\ &\text{s.t. } G(x) \geq 0 \\ &x \in D \end{aligned} \tag{1.1}$$

其中 x 为决策变量、$F(x)$ 为目标函数、$G(x)$ 为约束条件，D 表示离散的决策空间，为有限个点组成的集合。组合优化问题在国防、交通、产品制造、管理决策、电力、通信等领域都有广泛的应用[1]，常见的组合优化问题包括旅行商问题（Traveling Salesman Problem，TSP）、车辆路径问题（Vehicle Routing Problem，VRP）、车间作业调度问题（Job-shop Scheduling）、背包问题（Knapsack）、最小顶点覆盖问题（Minimum Vertex Cover，MVC）、最小支配集问题（Minimum Dominating Problem，MDP）等。

组合优化问题的特点是其决策空间为有限点集，直观上可以通过穷举法得到问题的最优解，但是由于可行解数量随问题规模呈指数型增长，无法在多项式时间内穷举得到问题的最优解，为此，数十年来学者对组合优化问题的求解算法进行了大量的研究，目前求解组合优化问题的方法主要包括精确方法（Exact approaches）和近似方法（Approximate approaches）两大类:

（1）精确方法是可以求解得到问题全局最优解的一类算法，主要包括分支定界法（Branch and Bound）[1, 2] 和动态规划法（Dynamic Programming）[3, 4]，其均采用分而治之的思想通过将原问题分解为子问题的方式进行求解[5]，通过不断迭代求解得到问题的全局最优解。

（2）近似方法是可以求解得到问题局部最优解的方法，主要包括近似算法（Approximate Algorithms）和启发式算法（Heuristic Algorithms）两类[5]。近似算法是可以得到有质量保证的解的方法，包括贪心算法、局部搜索算法、线性规划和松弛算法、序列算法等[6, 7]；启发式算法是利用设定的启发式规则对解空间进行搜索的一类方法，能够在可行时间内找到一个较好的解，但是对解的质量没有保证，文献中用来求解组合优化问题的启发式算法主要包括模拟退火算法[8, 9]、禁忌搜索[10, 11]、进化算法[12]（如遗传算法[13, 14]，差分进化算法[15, 16] 等）、蚁群优化算法[17, 18]、粒子群算法[19, 20]、迭代局部搜索[21, 22]，变邻域搜索[23, 24] 等。

精确方法可以求解得到组合优化问题的全局最优解，但是当问题规模扩大时，该类算法将消耗巨大的计算量，很难拓展到大规模问题。

相对于精确方法，近似算法可以在多项式时间复杂度的条件下求解得到具有最优性理论保证的解，但实际组合优化问题不一定存在该类求解算法，并且在问题规模大的情况下仍然需要大量的求解搜索时间。

启发式算法可以在可接受的计算时间内搜索得到一个较好的解，更加适合实际工程应用，但无法保证解的质量，基于群体智能的进化方法和局部搜索等方法都是近年来的研究热点，但是该类方法仍然是迭代型优化算法，当问题规模很大时，大量的迭代搜索仍然会导致较大的计算耗时，依旧很难拓展到在线、实时优化问题上。

此外，上述方法一般针对特定问题进行算法设计，需要不断地进行试验迭代过程，当问题规模大时设计成本很高；并且，一旦问题发生细微变化，上述方法仍然需要重新运行，再次进行搜索求解，或者通过不断试错对启发式规则进行调整以获得更好的效果，计算成本高。因此，面对通信、电力、军事等场景中存在的复杂大规模组合优化问题，传统组合优化算法面临着在线求解计算压力大、算法设计困难等瓶颈问题。在这种背景下，随着近年来人工智能技术的发展，采用人工智能技术解决传统组合优化问题的方法开始涌现。

1.2 研究意义

1.2.1 理论意义

近年来随着人工智能技术的发展，深度学习技术已经在很多领域打破了传统方法的壁垒，取得了令人瞩目的突破性进展。在计算机视觉领域，十多年前学者们主要利用人工设计的算法进行特征提取以及图像处理，但如今深度学习已经成为了当前的核心方法，深度神经网络（Deep Neural Networks，DNN）可以自动地对图像的特征进行学习，代替了人类的手工算法设计。

作为深度学习另外一个重要的分支，深度强化学习（Deep Reinforcement

Learning，DRL）主要用来做序贯决策，即根据当前的环境状态做出动作选择，并根据动作的反馈不断调整自身的策略，从而达到设定的目标。近年来深度强化学习在 AlphaGo Zero[25]、Atari[26] 等离散序列决策问题上的表现显示了其强大的学习能力和优化能力。

组合优化即在离散决策空间内进行决策变量的最优选择，与强化学习的“动作选择”具有天然相似的特征，且深度强化学习“离线训练、在线决策”的特性使得组合优化问题的在线实时求解成为了可能，因此利用深度强化学习方法解决传统的组合优化问题成为了近年来新兴的研究热点。相对于传统的迭代型优化算法，基于深度强化学习的组合优化方法无需搜索直接输出问题解，具有求解速度快的优势; 采用深度神经网络模型可以对相似问题的特征进行学习，模型一旦训练完成，可以对具有相同分布特性的所有问题实例进行求解，而不需要重新进行训练，具有很强的泛化能力，而传统算法一旦遇到一个新的问题实例，则需要从头开始重新进行搜索求解，不具备学习能力，耗费大量的计算成本。

人工智能技术的发展为求解组合优化问题提供了一种全新的思路，基于人工智能技术求解组合优化问题是最近几年刚兴起的研究方向，具有巨大的研究前景，但仍然处在起步阶段，因此论文对基于深度强化学习的组合优化新方法开展深入研究，对组合优化的单目标方法、多目标方法、分支定界法等不同方面进行了全方位的研究，突破了传统组合优化方法的局限性。

1.2.2 军事应用意义

组合优化问题在军事等领域存在大量应用，尤其在现代智能化、信息化作战的背景下，军事作战系统中的优化决策问题愈发复杂，例如作战力量组合优化调度、多作战主体作战优化协同、能源保障资源优化分配等优化场景，问题类型和规模愈发复杂庞大，存在多种复杂巨系统，存在大量装备、人员、信息网络的交互，系统内多类组合优化问题面临着问题规模大、高动态性、高约束等特点，并且战场情况瞬息万变，尤其需要优化引擎的快速响应，对优化算法的在线优化、实时响应能力提出了更高的要求。

而传统组合优化算法由于其原理性的局限性，很难在有限计算资源下实现复杂大规模组合优化问题的实时求解和在线优化，亟待研究新的优化方法对军事复杂巨系统的优化决策进行支持，论文所研究的基于深度强化学习的组合优化方法为解决组合优化问题提供了新思路，具有巨大的应用潜力，能够极大缩短组合优化问题的求解速度，以离线训练、在线优化、端到端的方式生成问题的近似解，在人工智能技术的辅助下有效缩短组合优化的求解时间，论文的研究内容覆盖了单目标、多目标、精确求解方法等组合优化的不同应用场景，可以为军事复杂系统

优化决策进行有效的理论算法支持，具有广泛的应用前景。

1.3 国内外研究现状

1.3.1 概述

利用神经网络解决组合优化问题的方法最早可追溯至 Hopfield 在 1985 年提出的 Hopfield 网络[27]，作者设计了一种 Hopfield 网络用于求解 TSP 问题以及其他组合优化问题[28]，但是该神经网络网络每次只能学习并解决单个小规模 TSP 问题实例，对于新给定的一个 TSP 问题需要从头开始再次训练，相对于传统算法并没有优势。

神经网络真正能够有效解决组合优化问题是在 2015 年，Vinyals 等人[29] 将组合优化问题类比为机器翻译过程（即序列到序列的映射），神经网络的输入是问题的特征序列（如城市的坐标序列），神经网络的输出是解序列（如城市的访问顺序），Vinyals 等人根据该思想，对机器翻译领域的经典序列映射模型（Sequence-to-Sequence，Seq2Seq）进行了改进，提出了可以求解组合优化问题的指针网络模型（Pointer Network）[29]，其原理详见第二章，作者采用监督式学习的方式训练该网络并在 TSP 问题上取得了较好的优化效果。多年来传统的组合优化算法都是以”迭代搜索”的方式进行求解，但是 Vinyals 等人的模型可以利用神经网络直接输出问题解，开启了组合优化一个新的研究领域。

自指针网络方法被提出后，近三年来多个基于指针网络架构的新方法在 NeurIPS，ICLR 等人工智能顶会上被相继提出[29–35]，在 TSP、VRP、Knapsack、多目标 TSP 等组合优化问题上显示了强大的优化效果，由于监督式学习需要构造大量带标签的样本，很难实际应用，目前大多数研究均利用深度强化学习方法对模型进行训练。

除指针网络模型之外，近年来随着图神经网络技术的兴起，部分学者采用图神经网络对组合优化问题进行求解，与指针网络模型不同的是，该类方法采用图神经网络对每个节点的特征进行学习，从而根据学习到的节点特征进行后续的链路预测、节点预测等任务。Dai 等人[36] 首次结合图神经网络和深度强化学习方法对 MVC、TSP 等组合优化问题进行了研究，作者利用图神经网络对各个“待选节点”的 Q 值进行估计，每次根据 Q 值利用贪婪策略向当前解插入一个新节点，直到构造一个完整的解。文献 [37–40] 使用了不同的图神经网络以及不同的解构造方法对组合优化问题进行求解，在二次分配问题、最大覆盖问题、MVC 等组合优化问题上取得了较好的效果。由于图神经网络主要进行节点特征的提取，部分研究[33, 34] 结合图神经网络和指针网络进行组合优化算法的设计，即首先使用图神经网络进行节点特征计算，再使用指针网络的注意力机制进行解的构造，在 TSP 等

问题上取得了较好的优化性能。

以上方法均为“构造（Construction）法”，即给定问题实例作为输入，利用深度神经网络对解进行构造，其中神经网络的参数一般利用深度强化学习方法训练得到。相对于传统的迭代型优化算法，构造方法无需搜索，直接输出问题解，具有求解速度快的优势; 且模型一旦训练完成，可以对具有相同分布特性的所有问题实例进行求解，而不需要重新进行训练，模型具有很强的泛化能力，而传统算法一旦遇到一个新的问题实例，则需要从头开始重新进行搜索求解。因此该类方法为求解组合优化问题提供了一种全新的思路，具有求解速度快、泛化能力强的优势。

但是由于构造方法通过神经网络直接输出最终解，解的最优性很难保证，上述方法在小规模问题上可以接近最优解，但是在中大规模问题上与 LKH3[41]、Google OR tools[42]、Gurobi[43]、Concorde 等专业组合优化求解器的优化能力还存在一定差距。

另一类是“改进（Improvement）法”，即采用深度学习技术改进传统的精确 / 近似方法，如利用机器学习模型对经典的精确求解算法: 分支定界法（Branch and bound）的 node selection 和 variable selection 策略进行选择，Lodi 等人[44] 已经对该类方法进行了详细的综述研究，本文不再赘述。除了对精确算法进行改进，近年来兴起的另一类方法是基于深度强化学习对迭代搜索类算法进行改进，局部搜索 / 邻域搜索是求解组合优化问题的常用近似方法，在局部搜索过程中，学者们通常手工设计各种启发式规则对解进行构造和搜索，但是随着人工智能技术的发展，通过神经网络模型代替手工规则设计是未来的发展趋势。鉴于此，近几年部分学者研究采用深度强化学习对解搜索的启发式规则进行学习和选择[45–48]，通过学习到的规则进行解的迭代搜索, 根据该思路，文献 [47,50] 所提方法在优化效果上达到甚至超过了 LKH3、Google OR tools 等专业组合优化求解器，文献 [48] 在求解速度上也超越了 LKH3、Google OR tools 等方法。

基于深度强化学习改进的局部搜索方法具有较好的优化效果，但其本质上仍然是迭代型搜索算法，求解速度仍然远不及构造方法; 构造方法具有求解速度快、泛化能力强的优势，但是该类方法的缺陷是解的优化效果无法保证，与传统组合优化方法的优化效果仍然存在一定差距。综上所述，目前基于 DRL 的组合优化方法主要分为基于 DRL 的构造方法和改进法两大类，其中构造方法主要包括基于指针网络的构造方法和基于图神经网络的构造方法两类，改进法主要对深度强化学习改进的局部搜索方法进行综述，本章对以上方法在近年来的研究进展进行综述介绍，对各类方法代表性算法的原理、优化性能、优缺点进行对比和介绍，各个算法的总结如表格 1.1 所示。

本章从理论方法研究和应用研究出发，对当前基于深度强化学习的组合优化方法的国内外研究现状进行总结和综述。

表 1.1　现有算法模型、训练方法、求解问题、以及优化效果比较

类别	研究	模型方法	求解问题及优化效果
基于指针网络的构造方法	2015 年 Vinyals[29]	指针网络 + 监督式训练	30 TSP 问题：接近最优解, 优于启发式算法; 40, 50-TSP：与最优解存在一定差距; 凸包问题、三角剖分问题
	2017 年 Bello[30]	指针网络 + REINFORCE	50-TSP：优于[29]; 100-TSP：接近 Concorde 最优解; 200-Knapsack：达到最优解。
	2018 年 Nazari[31]	指针网络 + REINFORCE	100-TSP：与[30] 优化效果相近, 训练时间降低约 60%; 100-CVRP：优于多种启发式算法。
	2018 年 Deudon[32]	Transformer Attention + REINFORCE	20, 50-TSP：优于[30]; 100-TSP：与[30] 优化效果相近。
	2019 年 Kool[33]	Transformer Attention + REINFORCE	100-TSP：优于[29–32, 36, 39]; 100-CVRP、100-SDVRP、100-OP、 100-PCTSP、SPCTSP：接近 Gurobi 最优解, 优于多种启发式方法。
	2020 年 Ma[34]	Graph Pointer Network + HRL	20, 50-TSP：优于[30, 39], 劣于[33]; 250, 500, 1000-TSP：优于[30, 33]; 20-TSPTW：优于 OR-Tools、蚁群算法。
基于图神经网络的构造方法	2017 年 Dai[36]	structure2vec + DQN	1200-TSP：接近[30]; 1200-MVC(最小顶点覆盖)：接近最优解; 1200-MAXCUT(最大割集)：接近最优解。
	2019 年 Mittal[37]	GCN + DQN	2k 至 20k-MCP(最大覆盖问题)：优于[36]; 10k, 20k, 50k-MVC：优于[36]。
	2018 年 Li[38]	GCN + 监督式训练 + 引导树搜索	实际数据集 MVC、MIS(最大独立点集)、 MC(极大团)、Satisfiability 问题：优于[36]。
	2017 年 Nowak[39]	GNN + 监督式训练 + 波束搜索	20-TSP：劣于[29]。
	2019 年 Joshi[40]	GCN + 监督式训练 + 波束搜索	20, 50, 100-TSP：略微优于[33], 优于[36]。

深度强化学习改进的局部搜索方法	2019 年 Chen[45]	指针网络 + Actor-Critic	20-CVRP：达到最优解； 50, 100-CVRP：优于[31, 33]、OR-Tools； 作业车间调度：优于 OR-Tools 、DeepRM。
	2019 年 Yolcu[46]	GNN + REINFORCE	实际数据集 Satisfiability、MIS、MVC、MC、图着色问题：更少搜索步数得到最优解、但单步运行时间长于传统算法。
	2020 年 Gao[47]	Graph Attention + PPO	100-CVPR：优于[33]； 100-CVPRTW：优于多个启发式方法； 400-CVRPTW：劣于单个启发式方法, 优于其他。
	2020 年 Lu[48]	Transformer Attention + REINFORCE	20, 50, 100-CVRP：优于[31, 33, 45], 以及优于 OR Tools、LKH3。且运行时间远低于 LKH3。

1.3.2 基于深度强化学习的组合优化理论研究综述

1.3.2.1 基于指针网络的构造方法

（1）方法综述

Vinyals 等人[29] 最早在 2015 年提出了 Pointer Network 模型进行组合优化问题求解，该文章也开启了利用深度神经网络进行组合优化问题求解的一系列研究，该模型借鉴机器翻译领域中的 Seq2Seq 模型求解组合优化问题，即利用基于深度神经网络的编码器将组合优化问题的输入序列（如城市坐标）进行编码，然后通过解码器和注意力计算机制（Attention）计算得到各节点的选择概率，并以自回归的方式逐步选择节点，直到得到完整解（如城市访问的顺序），该方法的详细原理见第二章。Vinyals 等人采用监督式学习的方法对该模型进行训练，即利用大量“TSP 城市坐标 - 最优路径”的样本对该 Pointer Network 的参数进行离线训练，利用该训练好的模型，可以求解与训练集具有相同分布特性的任意 TSP 问题。相对于传统的局部搜索或者启发式搜索方法，该模型不需要进行迭代搜索，具有泛化能力强、求解速度快的优势，论文实现了对至多 50 个城市的小规模 TSP 问题的快速求解。

由于 Vinyals 等人[29] 提出的方法采用监督式方式进行训练，导致其得到的解的质量永远不会超过样本的解的质量，并且事先构造训练样本需要耗费大量时间，因此限制了其应用于更大规模的组合优化问题。鉴于此，Bello 等人[30] 采用强化学习方法训练 Pointer Network 模型，他们将每个问题实例视为一个训练样本，以

问题的目标函数作为反馈信号，采用 REINFORCE 强化学习算法进行训练，并引入 Critic 网络作为 baseline 以降低训练方差。论文对至多 100 个城市的 TSP 问题以及 200 物品的 0-1 背包问题对该模型进行了测试，结果发现该模型在 50 城市 TSP 问题上超越了 Vinyals 等人监督式训练得到的模型，并可以在 100 城市的 TSP 问题上接近最优解，在背包问题上达到了最优解。

进一步地，Nazari 等人[31] 将 Pointer Network 拓展至具有动态特性的 VRP 问题，作者将输入分为两部分，包括静态输入（顾客位置 / 坐标）和动态输入（顾客需求），由于考虑到在输入端改变顾客的顺序不会影响问题的求解，因此作者将 Encoder 输入层的 LSTM 替换成简单的一维卷积层，从而可以有效降低计算成本。在仍然采用 REINFORCE 强化学习算法进行训练的情况下，他们的模型将训练时间降低了 60%，在 TSP 问题上与 Bello 等人的模型取得了几乎相同的优化效果，并且在 VRP、随机 VRP 问题上取得了比 Clarke-Wright Savings 和 Sweep 经典启发式搜索算法更好的优化效果。

相对于传统的 Seq2Seq 模型，近年来 Transformer 模型[49] 在自然语言处理领域取得了巨大的成功，Transformer 的 Multi-head Attention 机制可以使模型更好的提取问题的深层特征，鉴于此，多个最新的研究借鉴了 Transformer 模型进行了组合优化问题求解的研究。

Deudon 等人[32] 借鉴 Transformer 模型改进了传统的指针网络模型，其编码层采用了与 Transformer 模型编码层相同的结构，即利用 Multi-head Attention 方法计算得到节点的特征向量; 其解码层没有采用 LSTM，而是将最近三步的决策进行线性映射得到参考向量，从而降低模型复杂度，其注意力计算方式与传统 Pointer Network 模型相同，仍然采用经典的 REINFORCE 方法对该模型进行训练，文章仅对 TSP 问题进行了求解，作者首先利用该训练好的神经网络模型输出初始解，随后在该初始解的基础上进行一个简单的 2OPT 局部搜索，结果发现这种方式可以有效提高解的质量。

Kool 等人[33] 在 2019 年指出，虽然 Deudon 等人[32] 的模型结合局部搜索可以提高性能，但是其神经网络模型本身与传统的 Pointer Network 模型相比并没有显著的优势，鉴于此，Kool 等人借鉴 Transformer 模型，提出了一个可以利用注意力机制求解多种组合优化问题的新方法，在 TSP、Capacitated VRP （CVRP）、OP（Orienteering Problem）、PCTSP（the Prize Collecting TSP）等问题上性能超越了前述介绍的所有 Pointer Network 模型 [30-33]，并且高度接近 Concorde、LKH3、Gurobi 等专业求解器得到的最优解。该方法的改进主要包括两方面: 1）与文献 [32] 相同，该模型的编码层采用了和 Transformer 模型相同的 Multi-head Attention 机制，但解码层和注意力机制存在很大不同，首先该模型每一步的解码过程中考

虑的是第一步所做的决策和最近两步的决策，Deudon 等人[32] 模型注意力的计算方式仍然和经典指针网络相同，而该模型采用了 Transformer 模型的 Self-Attention 计算方法，增加了更多计算层以提高模型的表现; 2）进一步地，文章对强化学习训练算法进行了改进，前述所有文章均采用 REINFORCE 算法结合 Critic-baseline 的方式进行训练，即增加一个 Critic 神经网络来估计。作者指出同时训练两个神经网络是低效的，而且 Critic 很难得到的准确估计，因此文章设计了一种 greedy rollout baseline 来代替 Critic 神经网络: 即在之前训练过程中得到的所有策略模型里，选择在测试集中表现最好的模型作为基线策略，并采用贪婪方式进行动作选择，将利用该基线策略对状态 s 求解得到的目标函数值作为，如果当前策略比历史最优策略的表现好，则进行正向激励，从而对当前策略进行评价和参数更新。实验证明该训练方法的收敛能力明显优于传统方法。经过以上改进，该方法的优化性能超越了之前所有的构造方法。

进一步地，Ma 等人[34] 结合指针网络和图神经网络设计了一种图指针网络（Graph Pointer Network，GPN）用来求解大规模 TSP 问题以及带时间窗约束的 TSP 问题。该模型的编码器包含两部分：Point Encoder 以及 Graph Encoder，Point Encoder 对城市坐标进行线性映射，并输入到 LSTM 中得到每个城市的点嵌入，Graph Encoder 通过 GNN 图神经网络对所有城市进行编码，得到每个城市的图嵌入。模型根据图嵌入和点嵌入，基于注意力机制计算每一步城市选择的概率，并引入 Vector context 提高模型的泛化能力。文章采用分层强化学习方法（Hierarchical RL，HRL）对模型进行训练。在 50-TSP 问题上训练得到的模型可以有效求解 250，500，750，1000-TSP 等大规模 TSP 问题，在大规模 TSP 问题上超越了 Kool 等人[33] 的方法，但是在 100 以内的 TSP 上仍然劣于[33]。文章并对带时间窗约束的 TSP 问题进行了实验，性能超越了 OR-tool 以及蚁群算法，证明了分层强化学习训练方法在处理约束问题上的有效性。

（2）方法总结

以上为按时间线对各个代表性方法的介绍，Vinyals 等人[29] 最早提出了求解组合优化问题的 Pointer Network 模型，Bello 等人[30] 最先提出采用强化学习方法对该模型进行训练。目前 Kool 等人[33] 的方法在 100 规模以下的 TSP 问题上取得了当前业界最优（State-of-the-art，SOA）的优化效果，超越了其它基于指针网络模型的方法[29–32]。Ma 等人[34] 的方法在小规模 TSP 问题上劣于[33]，但是在大规模 TSP 问题（250,500,1000）上超越了[33]，各方法详细的性能对比详见表格 1.1。值得注意的是，上述方法在 50 规模以上的 TSP 问题上均未达到 Concorde、LKH3 等求解器得到的最优解。并且，当前采用深度学习方法对组合优化问题的研究大多仅限于简单的单目标组合优化问题，例如旅行商问题，对于复杂组合优化问题的

研究较少。

1.3.2.2 基于图神经网络的构造方法

（1）方法综述

Dai 等人[36] 在 2017 年首先研究了如何采用图神经网络对组合优化问题进行求解，作者采用 structure2vec 图神经网络对当前解的图结构进行建模，并根据图神经网络计算剩余可选节点中各个节点的 Q 值，随后基于贪婪策略根据 Q 值选择一个新的节点添加到当前解中，直至得到完整解。作者采用了深度 Q 学习（Deep Q-Learning，DQN）算法对该图神经网络的参数进行训练，以使模型输出准确的 Q 值估计。文章首先在 50-100 节点的 MVC、MAXCUT、TSP 问题上对该模型进行了训练，将训练好的模型在多达 1200 个节点的上述问题上进行了测试，以 CPLEX 求得的解作为最优解对模型的优化能力 - 求解速度进行了研究，实验结果表明该方法在 TSP 问题上取得了接近 Bello 等人[30] 方法的效果，在 MVC、MAXCUT 问题上得到了接近最优解的优化效果，且超越了多个基准算法。

Mittal 等人[37] 采用了与 Dai 等人[36] 相同的模型架构对大型组合优化问题进行求解，即结合图神经网络、DQN 以及贪婪策略进行解的构造，作者采用了图卷积神经网络（Graph Convolutional Networks，GCN）对图结构进行建模，在 20k 规模的最大覆盖问题（MCP）、50k 规模的 MVC 问题上进行了模型测试，实验发现该模型在大规模问题上的表现比 Dai 等人[36] 的模型获得了 41% 的优化能力的提升。

Li 等人[38] 采用图神经网络对最小顶点覆盖问题、最大独立点集（Maximal Independent Set，MIS）、极大团（Maximal Clique，MC）、适定性问题（Satisfiability）进行了研究，由于所研究问题均为点选择问题，与 TSP 问题不同，对节点选择的顺序无要求，因此文章没有采用逐步添加节点的方式构造解，而是使用 GCN 图神经网络直接输出所有点选择概率的估计值，并基于该估计值以引导树搜索的方式构造可行解。为了解决问题可能存在多个最优解的情况，文章采用 hindsight loss 方式输出多个概率分布，在此基础上进行树搜索，并采用局部搜索的方式对解进行再处理。文章与 Dai 等人[36] 的模型以及测试问题的多个基准方法进行了对比，在优化效果上均超越了对比算法。

以上方法均为对选择各个节点的概率进行估计，文献 [39, 40] 利用图神经网络对选择各个“边”的概率进行估计，以 TSP 问题为例，利用图神经网络模型输出一个邻接矩阵，代表两点之间存在边的概率，值大则节点 i 和 j 大概率相连。随后根据各个边出现概率的估计值，使用波束搜索（beam search）的方式构造最终的可行解。文献 [39, 40] 均采用监督式方法进行训练，即利用 LKH3 或 Concorde 求解器构造大量“坐标 - 最优路径”的训练数据，根据最优解的真实邻接矩阵和图神经网络输出的邻接矩阵计算交叉熵，以交叉熵为损失函数训练模型。Nowak 等

人[39] 使用的是经典 GNN 图神经网络模型[50]，该模型的优化效果没有超越传统的启发式方法以及指针网络模型，Joshi 等人[40] 采用的是 GCN 图神经网络，该模型在 20，50，100 规模 TSP 问题上的优化效果略微超越了 Kool 等人[33] 的方法，接近 LKH3、Concorde 等求解器得到的最优解，但是该方法的求解时间长于 LKH3、Concorde 等方法，在泛化能力上该方法也不及 Kool 等人[33] 的方法。

（2）方法总结

指针网络模型主要用于求解 TSP、VRP 等具有序列特性的组合优化问题（即该类问题的解与节点的顺序有关），由于指针网络利用注意力机制以自回归的方式对解进行构造，因此适用于求解序列组合优化问题。而基于图神经网络的方法由于得到的是节点的特征向量，自然地可以计算得到节点选择的概率，因此在 MVC、MIS 等顺序无关的点选择问题上多有应用，针对 TSP 等序列优化问题，一类方法是仍然以自回归的方式逐步选择节点[36]，另一类方式是根据节点的特征向量计算边选择的概率，然后利用波束搜索等方法构造解[40]。

表 1.2　不同构造方法在 TSP 问题上优化性能比较

方法类别	模型	TSP-20	TSP-50	TSP-100
最优	Concorde	3.84	5.7	7.76
基于指针网络(Attention 机制)	Vinyals[29]	3.88	7.66	-
	Bello[30]	3.89	5.95	8.3
	Nazari[31]	3.97	6.08	8.44
	Deudon[32]	3.86	5.81	8.85
	Deudon[32]+2OPT	3.85	5.85	8.17
	Kool[33](greedy)	**3.85(0s)**	**5.8(2s)**	**8.12(6s)**
	Kool[33](Sampling)	**3.84(5min)**	**5.73(24min)**	**7.94(1hour)**
基于图神经网络	Dai[36]	3.89	5.99	8.31
	Nowak[39]	3.93	-	-
	Joshi[40]	3.86	5.87	8.41
	Joshi[40]	**3.84**	**5.7**	**7.87**

旅行商问题是文献中研究组合优化问题的经典问题，表 1.2 对上述构造方法在不同规模 TSP 问题上的优化性能进行比较研究，各个模型采用 TensorFlow 或者 Pytorch 深度学习工具平台实现，由于文献 [33, 40] 是各类方法的 SOA 模型且均在相同型号的 1080Ti-GPU 上进行的实验，因此对文献 [33, 40] 的求解时间也进行了对比。

其中 Greedy 是采用贪婪策略构建 TSP 问题的解，即每次选取具有最大选择

概率的城市；2OPT 是对模型得到的解进行进一步的 2OPT 局部搜索以提高解的质量；BS 是采用 Beam Search 波束搜索的方式根据边选择的概率构造解。通过实验结果可见 Kool 等人[33] 的方法是当前基于注意力机制的构造方法的 SOA 模型，采用简单的贪婪策略即可在短时间内实现对 TSP 问题的高效求解；Joshi 等人[40] 利用图神经网络结合波束搜索对 TSP 问题进行求解，其优化效果超越了 Kool 等人[33] 的模型，但是该方法耗时过长。

由于图神经网络能够有效处理很多组合优化问题的图结构，近年来利用图神经网络求解组合优化问题的研究呈上升趋势，但该类方法仍然有很多问题待解决，例如波束搜索通常需要大量搜索时间，并且大多研究仍然采用监督式方式进行训练，需要构造大量标签样本，实际应用困难。Ma 等人[34] 将图卷积神经网络（GCN）和指针网络相结合，但是该方法在 100 规模的 TSP 问题上仍然劣于 Kool 等人[33] 的方法，但是在大规模 TSP 上存在优势，如何将指针网络的注意力机制和图神经网络相结合是一个重要的问题。

1.3.2.3 深度强化学习改进的局部搜索方法

虽然构造方法可以通过深度神经网络模型直接输出问题的解，实现组合优化的快速求解，但是其优化效果与 LKH3、Google OR tools 等专业求解器相比仍有一定差距。局部搜索（local search）是求解组合优化问题的经典方法，当前的局部搜索算法主要是通过人工对搜索的启发式规则进行设计，以获得更好的优化效果，鉴于近年来深度强化学习在在各领域取得的瞩目的学习能力，学者们开始研究利用深度强化学习方法来自动学习局部搜索算法的启发式规则，从而比人工设计的搜索规则具有更好的搜索能力。

（1）方法综述

Chen 等人[45] 于 2019 年提出了一个基于深度强化学习的组合优化问题搜索模型 NeuRewriter，它和局部搜索具有相似的算法流程，即首先随机构造一个可行解，在该初始解的基础上通过局部搜索不断提高解的质量。相比于传统算法所采用的人工设计的启发式规则，作者利用深度强化学习方法对局部搜索的策略进行训练，利用学习到的策略对搜索过程进行引导。其策略由两部分构成：Region-Picker 和 Rule-Picker，以作业车间调度问题为例，首先利用 Region-Picker 选定一个工序，其次利用 Rule-Picker 对该工序的操作策略进行决策，如与另一个工序进行调换。文章利用 Actor-Critic 方法对 Region-Picker 和 Rule-Picker 策略进行了训练，其优化效果在作业车间调度问题上超越了 DeepRM 和 Google OR-tools 求解器，在 VRP 问题上超越了 Google OR-tools 求解器。

Yolcu 等人[46] 采用深度强化学习改进的局部搜索方法对适定性问题（Satisfiability）进行了研究，仍然采用局部搜索的求解框架，利用深度强化学习对局部搜

索中变量选择算子进行学习，作者采用图神经网络对变量选择的策略进行参数化，利用 REINFORCE 算法更新图神经网络的参数，实验显示相对于传统的启发式算法，该方法可以在更少的步数内找到最优解，但是运行时间却远长于传统算法。

Gao 等人[47] 基于大规模邻域搜索框架对组合优化问题进行求解，作者利用深度强化学习方法对大规模邻域搜索的 Destroy 和 Repair 算子进行学习，采用图注意力神经网络（Graph Attention Network）对问题特征进行编码，并采用基于循环神经网络的解码器输出 Destroy 和 Repair 算子。具体地，Destroy 算子是从当前解中选择多个节点，并将其从当前解中移除，Repair 算子是将移除的节点以一定的顺序重新插入到当前解中，因此该模型对 Destroy 算子的点集选择策略和 Repair 算子的排序策略进行学习。文章采用 Proximal Policy Optimization （PPO）算法对模型进行训练，并用来解决 CVRP、带时间窗的 CVRP 等问题，实验表明该方法在 100 节点的 CVRP 问题上优化效果超越了 Kool 等人[33] 的方法，并在 400 节点的大规模 CVRP 问题上具有比传统启发式算法更快的收敛性能，在优化能力上接近但未达到最优解，但本文并没有提供求解时间对比。

Lu 等人[48] 于 2020 年提出了 Learn to Improve（LSI）组合优化问题求解方法，该方法不只在优化效果上超越了 LKH3、Google OR-tools 等组合优化求解器，其求解速度也超越了上述专业求解器。作者首先对 LSI 框架进行设计，算法总体流程仍然采用局部搜索的方式，在每一步搜索过程中，算法决定是继续提升当前解还是对当前解进行扰动，因此算法包括两个算子：提升算子和扰动算子，作者采用了九种不同的提升算子作为算子库，采用深度强化学习训练提升算子的选择策略，每次迭代，算法根据问题特征和当前的解，利用学习到的策略从算子库中选择提升算子，从而不断提升当前解的质量，如果达到局部最优，算法对当前解进行扰动。论文通过实验证明该方法在 20-，50，100-CVRP 问题上超越了当前 state-of-the-art 的 LKH3 求解器，并且其求解速度也远超 LKH3 算法。

（2）方法总结

深度强化学习改进的局部搜索方法是自 2019 年以来最新提出的一类组合优化方法，主要用于求解 VRP 等路径优化问题，表 1.3 对该类方法以及构造方法在求解不同规模 VRP 问题上的优化能力进行了比较，结果取自各个文献的实验数据。各算法均在 GPU 上运行（各算法使用不同型号 GPU，但运算时间不存在较大差距），均采用 Pytorch 深度学习工具实现。

从实验对比可以看出，深度强化学习改进的局部搜索方法在优化能力上优于当前性能最好的构造方法，Lu 等人[48] 模型的优化性能甚至超越了 LKH3 专业组合优化求解器，且算法运行时间数倍少于 LKH3；但是该方法的运算时间仍然远长于构造方法，Kool 等人[33] 的模型采用简单的贪婪策略可见在数秒内运算得到

表 1.3　不同模型在 VRP 问题上的优化性能比较

模型	VRP-20	VRP-50	VRP-100
LKH3	**6.14 (2h)**	**10.38 (7h)**	**15.65 (13h)**
Nazari[31]	6.4	11.15	16.96
Kool[33](greedy)	6.40 (1s)	10.98 (3s)	16.80 (8s)
Kool[33](sampling)	6.25 (6m)	10.62 (28m)	16.23 (2h)
Chen[45]	6.12	10.51	16.1
Lu[48]	**6.12 (12m)**	**10.35 (17m)**	**15.57 (24m)**

接近最优解的方案，具有快速在线求解的优势。可见两类不同的方法具有不同的优势，需要根据不同应用场景和问题规模进行选择.

1.3.2.4　基于深度强化学习的组合优化理论方法总结

从上述方法可见，构造方法具有求解速度远超传统优化算法的优势，也是近年来研究较多的一类方法，模型一旦训练完成，可以对任意同类型的问题进行求解，具有很强的泛化能力，但是很难保证解的优化效果，尤其随着问题规模的扩大，其优化能力与传统优化算法之间的差距会不断扩大。深度强化学习改进的局部搜索方法是近年来兴起的另外一类方法，其本质上仍然是启发式搜索算法，但是没有采用人工设计的搜索规则，而是利用深度强化学习算法对搜索规则进行学习，因此该方法具有较强的优化能力，其优化效果可以超越传统的优化算法，但是其求解时间仍然远慢于构造方法，因此需要根据优化效果和求解速度之间的权衡来选择不同的方法。

由于神经网络模型可以采用监督式和强化学习方法进行训练，文献 [51, 52] 对监督式和强化学习训练方法进行了详细的实验对比和分析，论文发现强化学习训练方法收敛比监督式训练方法慢，但强化学习得到的模型具有更强的泛化能力。

由于存在多种组合优化问题，不同文献的研究重点不同，导致存在多种不同的模型方法。例如 [31, 33, 48] 等文献偏重于解决 TSP、VRP 等路径优化问题，其中节点选择的顺序对结果有很大影响，因此基于注意力机制的方法在此类问题上有较好的效果。并且对于复杂的路径选择问题，如 CVRP 问题，目前的研究均采用注意力机制，而没有单纯采用图神经网络的方法，可见注意力机制在处理具有序列特性的组合优化问题上具有较好的性能；而 [36, 38, 46] 等文献偏重于解决 MVC、MAXCUT 等问题，即点选择问题，该类问题对节点的顺序没有要求，此种情况下图神经网络在该类问题上应用较多; 同时，结合图神经网络和注意力机制的方法在 TSP 等路径优化问题上也取得了较好的效果[34, 47]。

鉴于此，为了更清晰地对求解不同类型组合优化问题的不同模型方法进行分

析和总结，本节对解决不同组合优化问题的不同方法进行分类分析，并对各个研究所采用的模型进行归纳，结果如表 1.4 所示。可见，近年来图神经网络结合各种搜索方法（波束搜索、树搜索）在各种组合优化问题上得到了广泛的应用，其主要应用于没有序列特性的组合优化问题，如 MVC、MAXCUT 等。而基于注意力机制的指针网络方法是解决具有序列决策特性组合优化问题的主要方法，如 TSP、VRP 等问题。针对基于指针网络的构造方法，由于其核心是借鉴机器翻译领域的注意力机制，因此追踪当前自然语言处理领域的前沿成果是提升指针网络模型性能的重要思路，如 Kool 等人[33] 借鉴了 Transformer 模型中的 Multi-head Attention 机制，使得其模型在组合优化问题上取得了当前业界最优的效果。针对基于图神经网络的构造方法，由于图神经网络是当前人工智能领域的研究热点，如何从众多模型中选择改进适合求解不同组合优化问题的图神经网络模型是一个重要的问题，同时波束搜索、树搜索耗时长也是制约该类方法的一个瓶颈，如何高效地将图神经网络和注意力机制相结合是需要解决的难题。针对深度强化学习改进的局部搜索方法，目前的研究仍然处于起步阶段，但已经取得了超越传统组合优化求解器的成果，如何提高解搜索的效率以及扩大启发式算子的搜索空间是未来提升算法性能的重要内容。

1.3.3 基于深度强化学习的组合优化应用研究综述

组合优化问题广泛存在实际生产生活中的各个领域，随着在各个领域中实际问题规模的不断扩大以及对算法求解时间的严格要求，传统运筹优化方法很难在可接受时间内实现问题的在线求解，基于深度强化学习的组合优化方法作为近年来提出的一类前沿方法，具有求解速度快、泛化能力强的优势，本章对近年来该类方法的应用研究进行综述。首先对应用较多的网络与通信领域的研究进行综述，其次对交通、电网等其他领域的应用研究进行介绍。

1.3.3.1 网络与通信领域

由于网络与通信领域存在多种典型的组合优化问题，如资源分配、路由拓扑优化等，因此基于深度强化学习的组合优化在网络与通信领域存在较多的应用。

（1）资源分配

在网络与通信领域，资源分配问题是指将有限的 CPU、内存、带宽等资源分配给不同的用户或任务需求，如虚拟网络功能部署问题、网络资源切片问题等。

网络功能虚拟化技术（network function virtualization，NFV）通过标准的 IT 虚拟化技术将网络功能虚拟化，是当前网络通信的前沿技术，虚拟网络功能（Virtual Network Function，VNF）是 NFV 架构中的虚拟网络功能单元，VNF 的放置与部署问题是当前网络通信领域研究较多的一类问题，鉴于传统的 VNF 部署方法通

表 1.4　不同模型针对不同组合优化问题的方法比较

组合优化问题	文献	模型细节
TSP 问题	[29–35]	基于指针网络架构
	[36]	GNN+DQN
	[39, 40]	GNN+ 监督式训练 + 波束搜索
VRP 问题	[31, 33]	基于指针网络架构
	[45, 47, 48]	均采用强化学习训练局部搜索算子。[45]: 指针网络架构，[47]: Graph Attention 模型，[48]: Transformer Attention 模型
最小顶点覆盖问题 (MVC)	[36, 37, 46]	GNN + RL
	[38]	GNN + 监督式训练 + 树搜索
最大割集问题 (MaxCut)	[36]	GNN + DQN
	[53]	Message Passing Neural Network (MPNN) + DQN
	[54]	CNN&RNN + PPO
适定性问题 (Satisfiability)	[38, 46]	GNN + 监督式训练 /RL
最小支配集问题 (MDS)	[46]	GNN + RL
	[55]	Decision Diagram + RL
极大团问题 (MC)	[38, 46]	GNN + 监督式训练 /RL
最大独立集问题 (MIS)	[38]	GNN + 监督式训练 + 树搜索
	[56]	GNN + RL + 蒙特卡洛树搜索
背包问题 (Knapsack)	[30]	指针网络 + RL
车间作业调度问题	[45]	LSTM + RL 训练局部搜索算子
装箱问题 (BPP)	[57]	LSTM + RL
	[58]	NN + RL + 蒙特卡洛树搜索
图着色问题	[46]	GNN + RL
	[59]	LSTM + RL + 蒙特卡洛树搜索

常需要数十分钟才可以完成优化过程，近年来涌现出利用深度强化学习实现 VNF 智能在线部署的多个研究。文献 [60] 将 VNF 部署问题建模为混合整数规划问题，并将其转化为马尔可夫过程，在满足不同的端到端时延需求的前提下，以最小化总任务时间为目标，文章以 DQN 强化学习方法对模型进行训练，从而实现 VNF 在线部署，该方法在收敛性能以及优化能力上优于多个基准方法。文献 [61, 62] 基于 GNN 图神经网络对 VNF 网元资源需求进行预测，以提高 VNF 部署的准确

性。文献 [63] 考虑 VNF 网元资源分配的特性，指出传统强化学习方法很难处理 VNF 部署问题中的大规模离散决策空间探索问题，因此文章对 Deep Deterministic Policy Gradient algorithm （DDPG）强化学习算法进行了改进，提出了增强 DDPG 算法，该方法的优化能力超过了传统 DDPG 方法以及整数规划方法。文献 [64] 采用指针网络的 Encoder-Decoder 架构对 VNF 部署问题进行了求解，其 Encoder 和 Decoder 均采用 LSTM 模型，文章利用拉格朗日松弛将该约束问题转化为无约束问题，并采用基于蒙特卡洛的策略梯度方法对模型进行训练。通过与约束问题求解器 Gecode 和传统启发式方法对比，实验显示该方法的优化性能在大多数小规模和大规模问题上均优于对比算法。

文献 [65] 基于深度强化学习对无线边缘计算网络的切片策略进行了研究，文章设计了一种 D-DRL 分布式强化学习框架，采用一个中心协调器和多个分布式智能体对切片策略进行协同优化，并采用 DDPG 强化学习算法对模型进行训练。传统网络切片方法通常会受到不确定的资源需求、不确定的服务时间等不确定性因素影响，文献 [66] 将网络切片过程建模为半马尔可夫过程，采用 Deep Double Q-Learning 方法对切片策略进行优化，以克服 DQN 收敛慢的缺点，模型在训练时可以充分地对大规模决策空间进行探索，从而能够在在线优化时有效处理不确定的状态信息，进行实时在线响应，将不同的网络资源分配给不同种类的用户，实验证明该方法的长期优化能力相比于当前最优的方法提高 40%，且在线优化耗时可以忽略不计。

文献 [67] 对雾计算中的复杂资源分配问题进行了研究，将雾计算建模为马尔可夫过程，以在既定时延内满足用户最大需求为目标，利用 DQN 强化学习方法对雾计算中的资源分配进行在线求解，可以得到接近最优解的优化效果，并优于传统的启发式资源分配方法。

（2）拓扑与路由优化

在通信网络或者无线传感网络中，通常需要对路由策略、传感器的连接拓扑进行优化，以降低通信时延和成本。文献 [68] 基于深度强化学习方法对无线通信网络的路由策略进行研究，文章采用图神经网络对通信网络的图结构进行建模，对当前网络信息进行编码，并输出选择不同节点的 Q 值，采用 DQN 算法对图神经网络进行训练。实验显示该方法具有很强的泛化能力，一旦模型训练完成，能够对任意结构的网络进行路由策略的在线优化。文献 [69] 基于 DRL 和蒙特卡洛树搜索提出了一种 DRL-TC 方法，利用该方法对无线自组织传感网络的通信拓扑连接进行优化，文章采用深度神经网络对问题进行建模，利用强化学习方法对其进行训练，利用该神经网络的输出指导蒙特卡洛树搜索的过程，从而得到最优的通信拓扑连接。

文献 [70] 对无线传感网络中移动代理的路由策略进行了研究，采用指针网络模型的 Encoder-Decoder-Attention 架构输出移动代理的最优路径，文章利用 Actor-Critic 算法对其进行训练，实验显示该方法能够有效地对无线传感网络的流量进行控制，降低能量消耗。文献 [71] 基于深度强化学习方法对 D2D 无线通信网络的链路选择策略进行优化，文章采用指针网络模型结构对链路进行选择，其 Encoder 和 Decoder 均使用 LSTM 模型，并利用策略梯度方法对模型进行训练，实验证明该方法能够在更短计算时间内达到传统方法相似的优化性能。

（3）计算迁移

计算迁移（Computation Offloading）是通过将部分计算任务从本地迁移到远程设备以解决移动终端资源受限问题的一个有效途径，由于无线信道状况变化较快，需要快速进行策略相应，而传统的数值优化方法很难实现在线快速优化，鉴于此，多个文献 [72–74] 采用深度强化学习实现计算迁移策略的在线优化。文献 [72] 提出了一种基于深度强化学习的计算迁移框架，基于 DQN 算法对移动边缘计算网络对在线计算迁移策略进行优化，在优化时间和优化效果上优于传统整数规划算法。文献 [73] 基于深度强化学习方法对多址边缘计算的计算迁移策略进行了研究，文章采用 Seq2Seq 模型对策略进行建模，利用 PPO 算法对模型进行训练，在不同任务数量的情况下均接近最优解，且超越了多个基准方法。文献 [74] 将移动边缘计算网络中的计算迁移问题转换成了多维多背包问题，利用多指针网络对策略进行建模，并采用 REINFORCE 强化学习方法对该指针网络进行训练，最后采用波束搜索得到最终的方案。实验表明该方法的优化性能较基准启发式算法提高了 25%，且运行时间优于 OR-tool。

1.3.3.2 其他领域

（1）交通领域

在货物配送领域，随着电商规模的不断扩大，当前的配送路径优化方法很难做到城际交通规划系统的在线实时响应，鉴于此，文献 [75] 利用深度强化学习方实现了配送路径的在线快速生成，文章设计了一种基于 Struct2Vec 图神经网络和指针网络注意力网络的模型，采用图注意力机制对路径生成的过程进行建模，并采用策略梯度方法对该模型进行训练，文章基于德国科隆市的城市交通路网对该方法进行了测试，实验显示该方法可以在可接受时间内实现配送路径的快速生成，且在相同求解时间内优于传统的整数规划方法和启发式方法。

在网约车领域，订单的分配和司机载客区域的分配是一个复杂的组合优化问题，传统运筹优化方法很难处理大规模订单的在线调度和响应，文献 [76] 利用深度强化学习方法对该问题进行了研究，文章参考注意力机制对深度神经网络的结构进行了设计，并分别研究了 DQN 和 PPO 训练算法在不同场景下的表现，实验

表明 DQN 和 PPO 训练方法在不同的客流需求场景各具优势，且能够实现在线实时优化。文献 [77] 采用深度强化学习方法对交通信号灯控制策略进行优化，以信号灯持续时间作为优化变量，结合决策网络、目标网络、Double-Q-Learning 等深度强化学习方法对模型进行训练，取得了比传统交通信号灯控制方法更好的效果。

（2）生产制造领域

在生产制造领域存在大量组合优化问题，近年来基于深度强化学习的组合优化模型在生产制造领域也多有应用。文献 [45, 78] 对经典的车间工作流调度问题进行了研究，采用深度强化学习方法对局部搜索的解搜索规则进行学习，利用 Actor-Critic 深度强化学习算法对搜索规则进行优化学习，实验表明该两个模型在优化性能上均超越了传统启发式局部搜索方法。

置换车间工作流调度问题是对是对流水车间调度问题的进一步约束，文献 [79, 80] 均采用指针网络模型对该问题进行了研究，文献 [79] 采用了经典的指针网络模型，并利用 CPLEX 求解器构造大量样本对该模型进行监督式训练，实验结果表明注意力机制能够有效提高解的质量，文献 [80] 采用了 Multi-head 注意力机制进行建模，并利用 REINFORCE 强化学习算法对模型进行训练，实验表明该模型超越了多个启发式搜索算法和传统的指针网络模型。

（3）高性能计算领域人工智能模型的训练是一个耗时极长的任务，合理地对计算资源进行规划和调度能够有效提高计算效率，随着神经网络规模的不断扩大，多 CPU 和多 GPU 混合训练是当前通用的一个方法，将神经网络模型的不同计算功能部署在不同的计算设备上对训练速度有很大的影响，该问题是一个典型的组合优化问题，文献 [81, 82] 利用深度强化学习方法对该部署问题进行了研究，文献 [81] 采用了经典的指针网络模型架构对问题进行建模，并利用策略梯度方法进行分布式训练，实验证明该方法可以将 Tensorflow 计算图的训练速度提高 20% 以上。文献 [82] 在此基础上提出了一个分层模型，首先 Grouper 层将计算图中的不同计算部分进行分组，然后 Placer 层根据 Grouper 层得到的分组结果输出部署方案，Placer 层仍采用 Encoder-Decoder 模型，实验证明该方法可以将 Tensorflow 计算图的训练速度提高 60%。

（4）微电网能量管理领域

在微电网能量管理问题中，用电、储能等设备的启停控制是典型的离散优化问题，部分学者采用不同的深度强化学习方法对该问题进行了研究。Lavet 等人[83] 使用深度强化学习方法对微电网能量管理问题进行了研究，文章考虑了一个包含短期储能、长期储能以及光伏发电的微电网系统，将微电网能量管理问题建模为马尔可夫过程，以最小化用电费用为目标，以充电、放电、无操作作为动作空间，以卷积神经网络对该问题进行建模，采用 DQN 算法进行训练，实验发现该训练好

的模型能够有效地提高能源利用率，降低用电费用，但是文章没有和基准方法进行对比。

文献 [84] 构建了一个包含光伏发电、储氢装置、蓄电池的孤岛型复合能源系统，并将复合储能系统的协调控制转化为序列决策问题，文章仍然采用卷积神经网络对问题进行建模，采用 Double DQN 深度强化学习方法对该系统的协调控制策略进行优化，实验表明该方法与 DQN 算法相比具有更快的收敛速度。

文献 [85] 利用 Double Q-learning 深度强化学习方法对空调和通风系统的温度调节、启停等策略进行优化，从而在保证良好的温度舒适度和空气质量的前提下达到最低的能量消耗，以实现最优能量管理优化，文章在实际环境中对该模型进行测试，结果发现该方法与传统的温度调节方法相比更具有优越性。

文献 [86] 利用深度强化学习算法对对楼宇的智能能量管理问题进行了研究，对楼宇中空调、电视、电动汽车等用电设备的启停进行控制，文章利用深度神经网络对该问题进行建模，分别研究了 DQN 和 Deep Policy Gradient（DPG）训练算法在该问题上的表现，实验表明 DPG 策略梯度方法能够更加有效地实现削峰填谷和降低能源消耗。

由于基于深度强化学习的组合优化方法是近年来刚提出的一类方法，其应用研究较少。近年来的应用研究对指针网络、图神经网络模型以及其他深度神经网络模型均有研究，对 DQN、Double DQN、Actor-Critic、PPO、DDPG 等先进的深度强化学习方法也均有涉及。但各个应用研究都是针对各自不同的实际问题进行建模，其模型的结构、状态空间、动作空间都有较大区别，方法之间缺乏横向比较，实验对比较为匮乏，虽能体现出算法的优化效果，但对算法相对于传统方法的优化性能分析较少，虽然当前基于深度强化学习的组合优化应用研究较为初步，但由于工业、制造、交通等领域存在大量组合优化问题，基于深度强化学习的优化算法具有求解速度快、泛化能力强的优势，因此该类方法在国防、工业、交通等实际场景中具有广泛的应用前景。

从上述综述可见，当前基于深度强化学习的组合优化方法研究仍然处于起步阶段，采用人工智能方法对传统组合优化问题的求解大多只针对简单的组合优化问题，研究的问题规模较小，并且没有相关文献针对多目标问题进行研究，而实际生产生活当中的组合优化问题具有更加复杂的特性，亟需针对复杂大规模、多目标、动态非线性的组合优化问题进行研究，论文在此背景下，对基于深度强化学习的组合优化方法进行更加深入的研究。

1.4 论文主要内容与创新点

1.4.1 研究内容

传统组合优化方法很难实现问题的在线求解和快速决策，论文对基于深度强化学习的组合优化方法进行研究，从基于深度强化学习的组合优化框架、单目标、多目标、分支定界法几个方面开展研究，克服了传统组合优化方法的局限性，研究内容包括：

（1）借鉴人工智能技术解决传统的组合优化问题是近几年来新兴的研究方向，为组合优化问题的求解提供了一种全新的思路，因此论文首先总结回顾了近些年利用深度强化学习方法解决组合优化问题的相关理论方法与应用研究，对其理论方法研究和应用研究进行系统地总结和综述，指出当前各类方法的优势和缺陷，指出亟待解决的若干问题，从而引出本文的研究内容。

（2）不同于求解组合优化问题的传统运筹优化方法，采用深度强化学习方法求解组合优化方法涉及到大量新技术、新概念、新思路，需要对基于深度强化学习求解组合优化问题的基本原理和求解框架进行研究，基于此，论文对涉及到的深度神经网络模型、深度强化学习方法的相关理论进行阐述，对基于深度强化学习求解组合优化问题的科学原理和建模流程进行研究，提出基于深度强化学习求解组合优化问题的基本流程和框架。

（3）近年来涌现出应用人工智能技术求解组合优化问题的一系列方法，该类方法在一定程度上克服了传统组合优化算法的局限性，但是当前方法大多针对较为简单的单目标组合优化问题，如旅行商问题，对于复杂组合优化问题的研究较少，而且模型能够处理的问题规模通常较小，如当前方法大多只对 100 节点的旅行商问题进行研究，在这种背景下，论文采用深度强化学习方法对具有复杂动态特性的覆盖旅行商问题进行研究，对模型结构、训练方法进行研究和设计。

（4）当前采用人工智能技术求解组合优化问题的方法均针对单目标组合优化问题，目前尚无文献采用该类方法对“多目标组合优化问题”进行研究，但多目标问题广泛存在与实际生产、交通、通信、军事等实际领域中，采用人工智能方法求解传统多目标组合优化问题仍然存在很大挑战，论文对基于深度强化学习的多目标组合优化方法进行探索，对基于深度强化学习的多目标旅行商组合优化方法进行研究。

（5）上述方法均为组合优化的近似方法，即对组合优化问题的近似解进行求解，而分支定界法作为组合优化的另一类重要方法，能够求解得到问题的理论最优解，是当前 CPLEX 等主流优化求解器的核心算法，在大量场景下存在应用需求，但是传统分支定界法求解速度极慢，论文进一步对基于深度强化学习的分支

定界组合优化方法进行研究，以提高传统分支定界法的求解速度，从而实现对组合优化单目标、多目标、近似算法、精确算法的全方位研究，提出在不同应用场景下求解组合优化问题的系统的解决方案。

（6）论文最后将基于深度强化学习的组合优化方法应用于无人机路径优化实际问题上，考虑了一种基于无线能量传输的无人机路径优化场景，无人机无需降落回收，通过无线能量传输技术实现无人机执行任务的全流程无人化，首先对无人机功耗模型和无线能量传输模型进行研究，其次对该复杂问题的深度强化学习方法进行探索，从而实现无人机路径的实时在线优化。

本文六个研究内容的关系如图1.1所示。

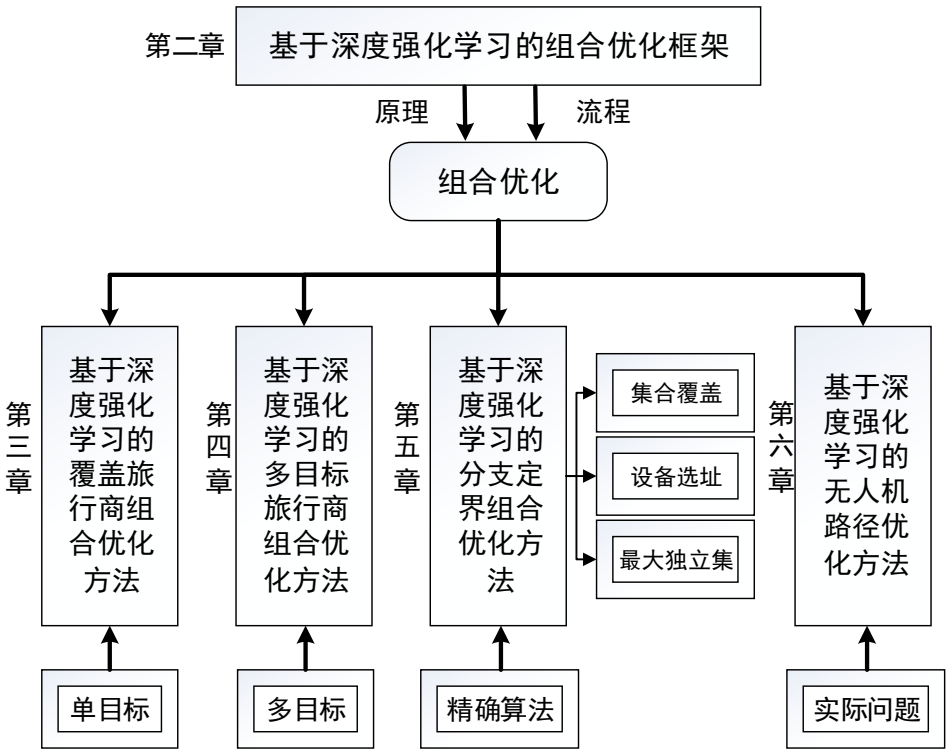


图 1.1　研究内容

1.4.2　主要创新点

针对基于深度强化学习的组合优化方法的关键问题和研究内容，本文的创新点如下。

（1）针对传统组合优化方法的缺陷，本文提出了基于深度强化学习的组合优化框架。将深度强化学习方法求解组合优化问题的流程分为三步：根据问题类型进行模型选择；设计深度神经网络对组合优化问题进行建模；采用强化学习方法对深度神经网络参数进行学习。并针对各个步骤，详细研究了组合优化问题建模

需要采用的深度神经网络模型，研究基于深度强化学习求解组合优化问题的科学原理，提出了基于深度强化学习的组合优化方法的流程框架和求解范式。

（2）针对复杂单目标组合优化问题，本文提出了一种基于深度强化学习的覆盖旅行商组合优化方法。设计了一种基于动态嵌入的注意力模型，该方法可以对具有动态特性、解序列长度不定、大规模的单目标覆盖旅行商问题进行求解，克服了当前深度强化学习优化方法在处理大规模、具有动态复杂特性的组合优化问题上的局限，相对于传统启发式方法，在达到相同优化水平的前提下，所提基于深度强化学习的覆盖旅行商问题求解方法能够获得 10 倍以上求解速度的提升，同时模型表现优于当前业界最优的其他人工智能模型。

（3）针对多目标组合优化问题，本文提出了一种基于深度强化学习的多目标旅行商组合优化方法。当前基于深度强化学习的组合优化方法均针对单目标问题进行研究，本文率先采用深度强化学习方法对多目标组合优化问题进行了研究，填补了该领域的空白。设计了一种分解策略和基于领域的参数迁移策略，所提方法在大规模、超多目标的多目标旅行商问题上均取得了超越传统启发式方法的性能，能够以数倍的求解速度获得更好的优化精度。

（4）针对能够得到组合优化问题理论最优解的分支定界法。本文提出了一种基于深度强化学习的分支定界组合优化方法，设计了一种图指针神经网络模型对分支定界法的分支策略进行建模，设计了全局特征和历史特征对问题状态进行更精确的表征，设计了一种基于 top-k KL 散度的模仿学习方法对模型进行训练，该方法实现了比专业优化求解器优化速度提升 40% 的性能，在三种不同组合优化问题上的表现均超越了当前基于机器学习改进的分支定界法，在求解速度和模型准确率上均获得了提升。

（5）针对实际应用问题。本文利用深度强化学习方法解决了考虑无线能量传输的无人机路径优化问题，首先对无人机功耗模型和无线能量传输模型进行了构建，考虑了悬浮充电以及移动充电模式，基于实际问题的约束，构建了基于多头注意力机制的深度神经网络模型，通过强化学习对模型进行离线训练，实现了无人机路径的实时在线优化，该方法相对于 Google OR-tools、粒子群、自适应大邻域搜索等传统方法，在不同规模问题上实现了 4 倍到 500 倍求解速度的提升。

1.4.3 论文结构

本文由七章组成，组织结构如下所示。

第一章绪论，首先介绍本文的研究背景和亟待解决的关键问题，综述了近些年利用深度强化学习方法解决组合优化问题的相关理论方法与应用研究，对其基本原理、相关方法、应用研究进行系统地总结和综述，指出当前各类方法的优势

和缺陷，指出亟待解决的若干问题，从而引出本文的研究内容。

第二章提出了基于深度强化学习的组合优化框架，首先对组合优化问题、深度神经网络模型、深度强化学习方法的相关理论进行了阐述，从而提出了基于深度强化学习求解组合优化问题的科学原理和求解框架，设计了基于深度强化学习求解组合优化问题的基本方法和流程。

第三章针对组合优化的复杂单目标问题进行了研究，提出了一种基于深度强化学习的覆盖旅行商组合优化方法。首先介绍该问题的研究背景，其次提出了问题的建模方法、神经网络模型架构、模型训练方法，最后，本章通过大量算法对比和模型测试验证了所提出算法的有效性。

第四章针对组合优化的多目标问题进行了研究，提出了一种基于深度强化学习的多目标旅行商组合优化方法。首先介绍该问题的研究背景，其次针对多目标优化问题的特点，提出了一个基于深度强化学习求解多目标优化问题的构造方法，尔后介绍了问题的建模方法和训练方法，最后，本章通过大量算法对比和模型测试验证了所提出算法的有效性。

第五章针对分支定界法进行了研究，提出了一种基于深度强化学习的分支定界组合优化方法。首先将分支定界过程建模为马尔科夫决策过程，并提出了针对该问题的特征提取和建模方法，随后设计了一种高效的模型训练算法，最后通过与传统分支定界法和机器学习方法的实验对比验证了所提出算法的有效性。

第六章针对实际问题进行了研究，将基于深度强化学习的组合优化方法应用到了一种考虑无线能量传输的无人机路径优化问题上，并提出了该问题的模型和深度强化学习方法，通过离线训练在线应用的方式实现了无人机路径的实时在线优化，与传统方法的实验对比验证了所提算法的有效性。

第七章对全文内容进行总结，并展望下一步的研究方向。

第二章　基于深度强化学习的组合优化框架

不同于传统组合优化方法，采用人工智能方法求解组合优化方法涉及到大量新技术、新概念、新思路，本章对基于深度强化学习求解组合优化问题的科学原理进行研究，提出基于深度强化学习的组合优化方法的框架和求解范式。深度强化学习方法求解组合优化问题主要分为三步：根据问题类型进行模型选择；设计深度神经网络对组合优化问题进行建模；采用强化学习方法对深度神经网络参数进行学习。本章针对各个步骤，详细研究了组合优化问题建模需要采用的深度神经网络模型，对建模过程和训练方法进行介绍，最终提出基于深度强化学习的组合优化框架。

2.1　基于深度强化学习的组合优化基本概念和原理

本节主要对基于深度强化学习的组合优化方法的基本原理进行研究，并对相关概念进行介绍。

2.1.1　深度神经网络模型

2.1.1.1　循环神经网络模型

在组合优化问题中，很多问题都具有序列特征，例如旅行商问题需要对城市访问顺序进行优化，因此，可考虑采用能够处理序列数据的神经网络模型对组合优化问题进行建模，循环神经网络（Recurrent Neural Network，RNN）是专门用来处理序列数据的神经网络架构。

传统神经网络一般为前馈神经网络，即输入信息通过神经网络节点向输出层传播，输出只与当前输入有关。而循环神经网络不仅以当前的输入信息作为模型输入，同时将上一步模型的输出作为当前步模型的输入，因此能够记忆序列信息。具体地，循环神经网络结构如图 2.1 所示，其计算方法为：

$$\boldsymbol{h}^{(t)} = f\left(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}\right) \tag{2.1}$$

当前的隐向量 $\boldsymbol{h}^{(t)}$ 由参数为 $\boldsymbol{\theta}$ 的 RNN 计算得到，其输入是上一步 RNN 输出的隐向量 $\boldsymbol{h}^{(t-1)}$ 和当前的输入信息 $\boldsymbol{x}^{(t)}$。

但是循环神经网络在实际应用时常出现梯度消失、梯度爆炸等问题，即随着步数增多，梯度呈指数级下降或者上升的问题，难以处理长序列信息。因此近年来多种循环神经网络的变体被提出，如带存储单元的长短时记忆网络（Long Short Term Memory，LSTM）和门控循环单元（Gated Recurrent Unit，GRU）等，LSTM 设计了一种门机制来解决梯度消失问题，并采用 cell state 来保存长期记忆，配合

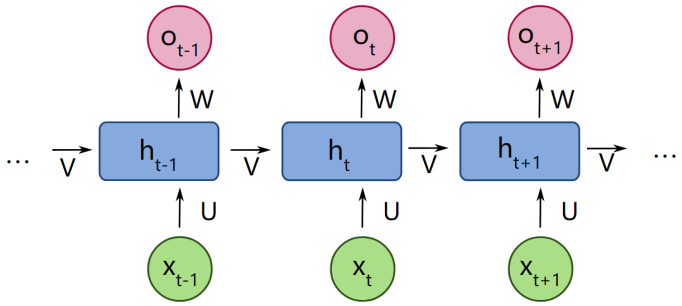


图 2.1　循环神经网络结构图

门机制对信息进行过滤，从而可以有效存储长期记忆，GRU 也采用门机制来解决梯度问题以及无法处理长期记忆的问题，但是 GRU 相对于 LSTM 具有更少的参数，因此具有更快的收敛速度，其结构如图 2.2 所示。

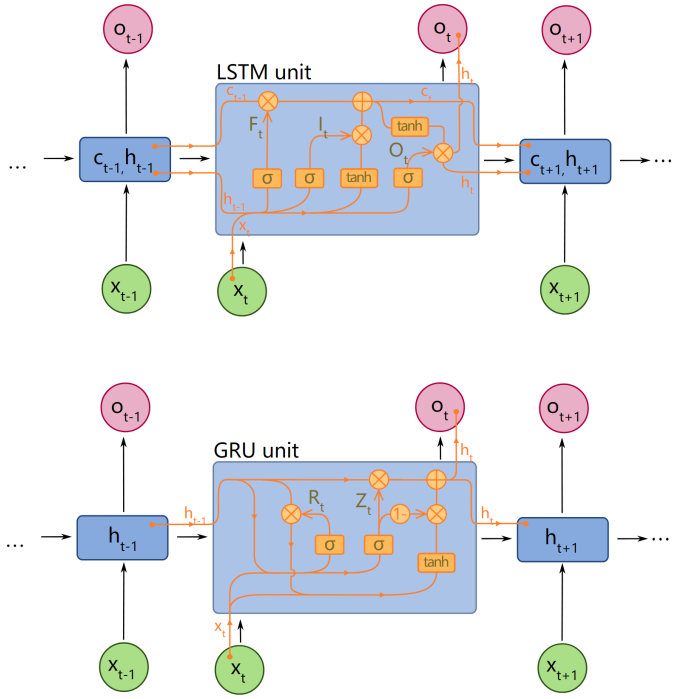


图 2.2　LSTM 和 GRU 等循环神经网络变体

2.1.1.2　序列到序列模型

从组合优化问题的特征可以发现，很多组合优化问题可以看作“序列到序列”的映射，例如旅行商问题以城市坐标序列作为输入，输出城市的访问序列，因此，可以借鉴机器翻译领域著名的“序列到序列模型”对组合优化问题进行建模。

序列到序列模型于 2014 年被文献 [87] 和 [88] 相继提出，分别被命名为编码器 - 解码器模型（Encoder-Decoder Model）以及序列到序列模型（Sequence-to-

Sequence，Seq2Seq）。其应用特点是能够实现序列到序列的映射，常用于机器翻译等自然语言处理领域，该神经网络模型的基本结构如图 2.3 所示。

模型由编码器和解码器构成，编码器和解码器均为循环神经网络以处理序列信息，首先通过编码器将输入序列转换为一个定长的向量，解码器以该向量为首个隐层状态，以上一步预测得到的值作为输入，不断解码得到每一步的预测值，最终构成输出序列。从图 2.3 可以看出，该神经网络架构能够实现输入序列到输出序列的映射，因此可以借鉴该神经网络架构实现组合优化问题的建模。

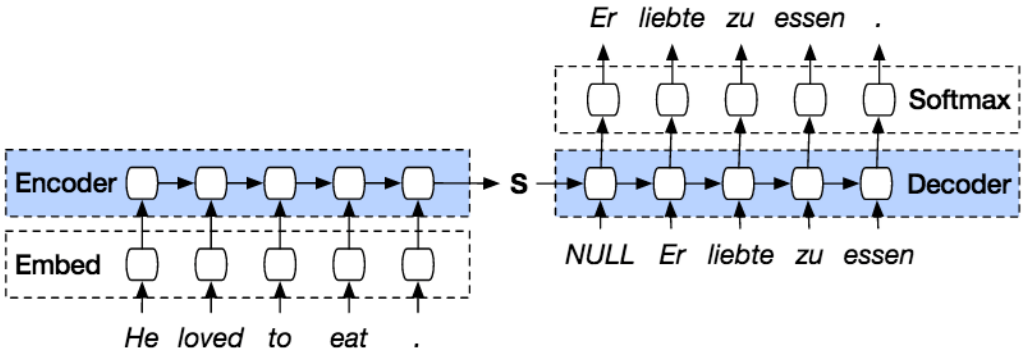


图 2.3　序列到序列模型示意图

2.1.1.3　注意力机制

传统的序列到序列模型将输入序列编码为一个向量，再对该向量进行解码得到输出序列，但是该方式存在一系列问题。首先编码器将整个输入序列的信息压缩到了一个向量当中，一旦问题规模变大，会导致输入序列的长度增加，这种压缩的方式难以避免信息的损失，无法表示整个输入序列的信息；其次，解码器根据该定长向量不断解码，随着序列长度的增加，存在信息稀释的问题，并且无法得到序列内部长期依赖的关系。

同时，针对组合优化问题，相对于机器翻译的应用，组合优化的规模常常远大于翻译语句的长度，因此传统序列到序列模型很难实际应用于组合优化问题的建模。

为了解决传统序列到序列模型在机器翻译任务中存在的局限性，文献 [89] 在 2014 年提出了著名的注意力机制（Attention），该神经网络模型的架构如图 2.4 所示，以英语 - 德语翻译为例，编码器不只将输入序列编码为一个定长的向量，并且生成每个输入节点的特征向量，在解码过程中，每翻译一个单词时，将该步的上下文向量（Context）和编码器生成的每个输入节点的特征向量进行注意力计算，从而得到针对每个输入节点的注意力值，根据该注意力值进行解码，从而使得翻译过程更加关注与当前翻译单词关系更大的输入单词。

例如，在解码器的第一步翻译德语 Er 时，通过注意力机制可以计算得到与英语 He 之间的注意力值更大，因此翻译该单词时主要关注于将 He 翻译为德语，后续每一步的翻译过程都会通过注意力机制来计算与输入单词之间的关系，翻译每个单词时都将更多“注意力”投入于与当前关联最大的单词，因此能够克服序列长度增加而导致的信息丢失和长期依赖计算困难的问题，提高解码的准确度。

因此，可以借鉴注意力机制，采用注意力增强的序列到序列模型对组合优化问题进行建模，从而提高优化求解的准确性。

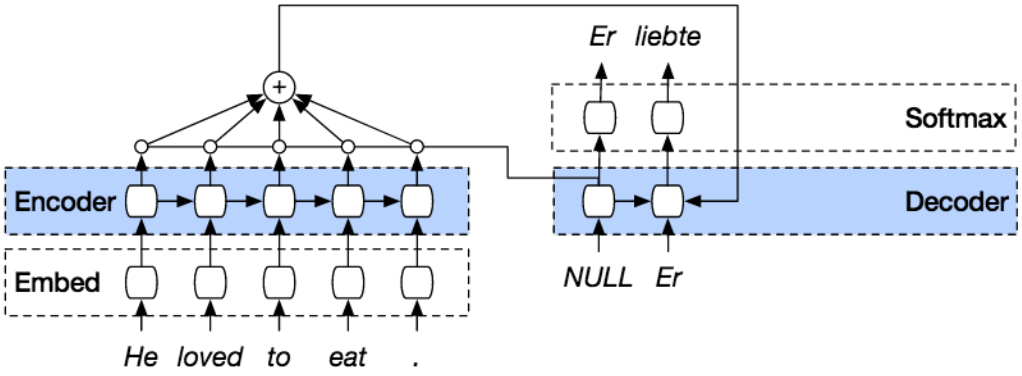


图 2.4　注意力增强的序列到序列模型示意图

2.1.1.4　图神经网络

图在现实生活中广泛存在，如社交网络图，引文网络图等，很多组合优化问题可以建模为图结构，如旅行商问题，每个节点代表城市，边即两个城市之间的路径，而边的权重代表城市之间的距离，因此可以将组合优化问题建模为图结构。

图是一类由节点和边构成的数据结构，一个图通常由其邻接矩阵 A 表示，如果一个图含有 N 个节点，那么邻接矩阵 A 的大小为 $N \times N$，其中 $A_{ij} = 1$ 表示第 i 个节点和第 j 个节点相邻。图可以分为多种类型，如有向图和无向图，对于有向图 A_{ij} 和 A_{ji} 不相等，无向图则相反，A_{ij} 的值也可以为其他数值，比如边的权重。

传统处理图数据的方法主要包括以下几个类别：

- 搜索算法，如深度遍历和广度遍历方法。
- 最短路算法，如 Dijkstra 算法和最近邻算法。
- 生成树算法，如 Prim 算法等。

但是传统算法需要得到图的先验知识，并且应用范围很小，无法处理诸如节点分类、图分类等任务。因此近年来图神经网络作为处理图数据的一种深度学习方法得到了广泛关注。

图神经网络，顾名思义，即用神经网络对图数据进行建模，但是图数据的复杂特性为神经网络方法带了了极大的挑战，例如图结构具有不规则的特点，图中

的节点可能具有不同数量的邻居，边的稀疏性等。图神经网络通过节点和边的关系，利用设计的神经网络结构和计算方法，计算得到能够代表节点特征的节点向量，当前图神经网络方法主要包含以下三类:

(1) 循环图神经网络

循环图神经网络（Recurrent Graph Neural Networks, ReCNNs）最具有代表性的一类图神经网络方法，它利用循环神经网络的思想进行节点特征的学习，即递归交换邻域信息来更新节点状态，直到达到稳定的均衡状态，具体地，ReCNNs通过集成节点上一步的特征向量、节点当前的特征、邻接的特征、边的特征对节点的特征向量进行不断更新，通过多步更新直至达到稳定平衡态，从而得到节点最终的向量表示。

循环图神经网络基于巴拿赫不动点定理：如果 $f : X \to X$ 为压缩映射，且 (X, d) 为完备度量空间，则 f 具有唯一不动点，且从任意起始点 x_0 开始，当 $n \to \infty$ 时，$f(x_n)$ 收敛于 $f(x_*)$，其中 x_* 为不动点。这意味着如果将映射函数 f 应用于初始点 x_0 k 次，$f(x_k)$ 应当与 $f(x_{k-1})$ 几乎相同，即达到收敛稳态。

因此，循环图神经网络基于上述不动点假设，对节点向量进行更新，节点向量在第 t 步的更新过程如下:

$$\mathbf{h}_v^{(t)} = \sigma_{u \in N(v)} f\left(\mathbf{x}_v, \mathbf{x}^{\mathbf{e}}_{(v,u)}, \mathbf{x}_u, \mathbf{h}_u^{(t-1)}\right) \tag{2.2}$$

其中，节点的初始特征 $\mathbf{h}_v^{(0)}$ 通过随机的方式进行初始化，或者通过节点已知的特征进行初始化，$f(\cdot)$ 代表一个参数化的映射函数，通常为一个神经网络，σ 代表一个集成操作，如求和、平均或者取最大值，$\mathbf{h}_v^{(t)}$ 代表节点 v 在当前更新步的节点特征向量，$N(v)$ 代表 v 的邻居节点的集合，$\mathbf{x}_v$ 是节点 v 的特征，$\mathbf{x}_u$ 是邻居节点 u 的特征，$\mathbf{x}^{\mathbf{e}}_{(v,u)}$ 是与 v 相连的边的特征，$\mathbf{h}_u^{(t-1)}$ 是邻居节点 u 在上一步更新的特征向量。通过上述计算方法得到节点最终的特征向量。

(2) 基于频谱的图卷积神经网络

卷积即为各节点聚合自身与邻居的状态得到新的状态，与循环图神经网络不同的是，图卷积神经网络（Convolutional graph neural networks, ConvGNNs）采用固定的层数，且各卷积层神经网络的参数不同。图卷积神经网络主要包括基于频谱的图神经网络（Spectral-based ConvGNNs）和基于空间域的图神经网络（Spatial-based ConvGNNs）两种。

基于频谱的图神经网络将图卷积操作定义为图信号处理领域中的滤波器操作，图卷积算子定义为从图信号中去除噪音，该类方法具有严格的数学基础，具体地，

每一层通过卷积滤波对节点向量的更新方式如下:

$$\mathbf{H}_{:,j}^{(k)}=\sigma\left(\sum_{i=1}^{f_{k-1}}\mathbf{U}\Theta_{i,j}^{(k)}\mathbf{U}^{T}\mathbf{H}_{:,i}^{(k-1)}\right)\quad(j=1,2,\cdots,f_{k}) \tag{2.3}$$

其中 k 代表第 k 层，$\mathbf{H}^{(k-1)}\in\mathbf{R}^{n\times f_{k-1}}$ 代表输入的图信号为上一层的滤波输出，$\Theta_{i,j}^{(k)}$ 为一个对角矩阵，每一个元素为可学习的神经网络参数，其中 $\mathbf{U}$ 通过计算拉普拉斯矩阵的特征向量得到，具体地，计算无向图邻接矩阵 A 的拉普拉斯矩阵:

$$L=I-D^{-\frac{1}{2}}AD^{-\frac{1}{2}}, \tag{2.4}$$

其中 D 为节点度的对角矩阵，即 $\mathbf{D}_{ii}=\sum_{j}(\mathbf{A}_{i,j})$。因为对称性，$L$ 有标准正交的特征向量矩阵 U 使得 $L=U\Lambda U^{T}$，其中 $\Lambda=diag(\lambda)$。

(3) 基于空间域的图卷积神经网络

由于 Spectral-based ConvGNNs 需要通过邻接矩阵计算节点的向量表示，一旦邻接矩阵发生微小的变化，特征值和特征向量都会发生改变，从而影响节点向量的计算，因此很难泛化到其他图结构上，并且特征值和特征向量的计算耗时明显，不适用于大规模图。

鉴于此，基于空间域的图神经网络在一定程度上克服了基于频谱图神经网络的缺陷，基于空间域的图神经网络模仿传统卷积神经网络中的卷积运算，根据节点的空间关系定义图的卷积，将图中节点的信息与其邻居节点的信息进行聚合，从而得到节点最终的特征向量表示。具体地，基于空间域的图神经网络可大体分为以下计算步骤:

在信息传递阶段，首先确定好节点的邻域范围，得到以节点为中心的子图，根据节点邻域定义的子图进行邻域信息聚合:

$$m_{v}^{k}=\sum_{u\in N(v)}M_{k}\left(h_{v}^{k-1},h_{u}^{k-1},e_{vw}\right) \tag{2.5}$$

即在第 k 步的消息传递过程中，对节点自身的隐层向量、节点邻域的隐层向量、邻域边的特征进行聚合，利用聚合函数 $M_{k}(*)$ 进行聚合，得到邻域节点的信息表示。

在更新阶段，将得到的邻域节点信息和节点当前的隐层向量进行结合:

$$h_{v}^{(k)}=U_{k}\left(h_{v}^{(k-1)},m_{v}^{k}\right) \tag{2.6}$$

从而得到节点该阶段的节点向量，因 Spatial-based ConvGNNs 计算节点向量的整体过程就是将节点邻域信息聚集，结合中心节点状态来更新中心节点，得到下一

阶段的节点向量，每层神经网络的参数可以不同，计算方式如下：

$$h_v^{(k)} = U_k\left(h_v^{(k-1)}, \sum_{u\in N(v)} M_k\left(h_v^{(k-1)}, h_u^{(k-1)}, x_{vu}^e\right)\right) \tag{2.7}$$

2.1.2 强化学习方法

组合优化问题属于离散决策问题，通常可以建模为马尔可夫决策过程，每个决策变量的选择问题可以看作为智能体的动作，从而可以采用强化学习方法进行求解。强化学习（Reinforcement Learning，RL）是智能体（agent）通过与环境的交互，以不断试错的方式学习得到可以最大化回报的最优策略的一类方法，深度强化学习方法近年来提出的一类新方法，它将深度神经网络和传统强化学习方法相结合，在 Alpha Zero 等领域的成功展示了其强大的序贯决策能力。

在强化学习方法中，智能体根据环境信息以及自身的策略进行动作的选择，动作进行之后环境得到了更新，同时返回一个奖励值，强化学习的目标就是通过对动作选择的策略进行学习以获得最高的累计奖励。

在强化学习中，通常采用 (S, A, T, R) 元组代表马尔可夫过程，其中 S 代表环境的状态集合，A 代表智能体所能执行的动作集合，$T : S \times S \times A \to [0, 1]$ 是状态转移函数，$T(s, s', a)$ 代表状态 s 在执行动作 a 后转移到状态 s' 的概率，在此满足 $\sum_{s'}^{S} T\left(s, s', a\right) = 1 \forall s \in S, a \in A$，其中，$R : S \times A \to \mathbb{R}$ 是在该状态下采取动作的奖励值。

具体地，在每一步 t，智能体在状态集合 S 中感知当前的状态 s_t，并根据自身策略 $\pi(a_t|s_t)$ 从动作空间 A 中选取动作 a_t，执行动作 a_t 后根据状态转移概率 $p(s_{t+1}|s_t, a_t)$ 状态转移到状态 s_{t+1}，并且根据回报函数 $R(s, a)$ 返回一个奖励值 r_t。在回合制问题中，该过程不断重复直到回合结束，累计奖励为 $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$，其中 $\gamma \in (0, 1]$ 是折扣因子。智能体的目标是最大化期望奖励。

强化学习的关键概念是值函数（Value Function），包括状态值函数（V 函数）和动作值函数（Q 函数）。状态值函数定义如下：

$$V_\pi(s) = \mathbb{E}\left[G_t \mid S_t = s\right] \tag{2.8}$$

表示在该策略 π 下，对状态 s 的评价，定义为从状态 s 开始执行策略 π 的期望回报，其中 G_t 代表奖励回报值。因此强化学习的任务就是寻找一个最优的策略使得期望回报最大，即 $V_*(s) = \max_\pi V_\pi(s)$。

同样地，对于一个状态动作对 (s,a) 的期望回报称作动作值函数，即 Q 函数，表示在该状态 s 下执行动作 a 的期望回报：

$$q_\pi(s,a)=\mathbb{E}_\pi\left[G_t \mid S_t=s, A_t=a\right] \tag{2.9}$$

最优的动作值函数定义为 $q_*(s,a)=\max_\pi q^\pi(s,a)$。

根据贝尔曼方程（Bellman Equation），状态值函数 v_π 可以分解为：

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\left[G_t \mid S_t=s\right] \\ &= \mathbb{E}_\pi\left[\sum_{k=0}^{\infty}\gamma^k R_{t+k+1} \mid S_t=s\right] \\ &= \mathbb{E}_\pi\left[R_{t+1}+\sum_{k=0}^{\infty}\gamma^k R_{t+k+2} \mid S_t=s\right] \\ &= \sum_a \pi(a \mid s)\sum_{s'}\sum_r p\left(s', r \mid s, a\right) \\ &= \sum_a \pi(a \mid s)\sum_{g'}\sum_r p\left(s', r \mid s, a\right)\left[r+\gamma v_\pi\left(s'\right)\right] \end{aligned} \tag{2.10}$$

同样地动作值函数可以根据贝尔曼方程分解为：

$$q^\pi(s,a)=\sum_{s',r} p\left(s', r \mid s, a\right)\left[r+\gamma V_\pi\left(s'\right)\right] \tag{2.11}$$

根据 V 函数和 Q 函数的子问题结构，可以通过蒙特卡洛学习、时序差分学习（Temporal difference learning）等方法对值函数进行近似。

2.1.2.1 蒙特卡洛学习

由于在实际环境中状态转移函数 $p\left(s', r \mid s, a\right)$ 未知，因此蒙特卡洛学习方法通过蒙特卡洛采样积累大量 $(S_t, A_t, R_{t+1}, S_{t+1})$ 状态转移对，在学习过程，每次都需要执行完成整个马尔可夫过程，从而记录最终的回报 G_t，每个状态的值函数从而根据实际的回报 G_t 进行更新：

$$V\left(S_t\right) \leftarrow V\left(S_t\right)+\alpha\left[G_t-V\left(S_t\right)\right] \tag{2.12}$$

该过程不断迭代，从而使得学习得到的值函数逼近最优。

2.1.2.2 时序差分学习

蒙特卡洛学习方法需要等待整个问题回合结束之后，统计实际奖励值对值函数进行更新，而时序差分学习在每步动作执行之后就进行策略更新：

$$V\left(S_t\right) \leftarrow V\left(S_t\right)+\alpha\left[R_{t+1}+\gamma V\left(S_{t+1}\right)-V\left(S_t\right)\right] \tag{2.13}$$

其中 α 是学习率，$R_{t+1}+\gamma V\left(S_{t+1}\right)-V\left(S_t\right)$ 为 TD-error，式 (2.12) 采用回合结束之后的总回报值 G_t 进行策略的更新，而时序差分采用单步的回报值进行策略更新，即根据贝尔曼方程采用 TD-error 对值函数 $V\left(S_t\right)$ 进行更新。SARSA[90] 和 Q-learning[91] 是应用广泛的时序差分学习方法，Q-learning 对 Q 函数更新的方法如下：

$$Q(s,a) \leftarrow Q(s,a)+\alpha\left[r+\gamma Q\left(s',a'\right)-Q(s,a)\right] \tag{2.14}$$

2.1.2.3 策略梯度方法

蒙特卡洛学习和时序差分学习都是对值函数进行学习，另一种方法是直接对策略 $\pi(a \mid s;\boldsymbol{\theta})$ 进行优化，策略由参数化的函数进行近似，通过梯度下降方法对策略的参数 $\boldsymbol{\theta}$ 进行更新，更新的方向和数值由 $\nabla_{\boldsymbol{\theta}} \log \pi\left(a_t \mid s_t;\boldsymbol{\theta}\right) R_t$ 决定，即以最大化回报的方向对策略进行更新。

2.1.2.4 深度强化学习

传统强化学习方法的值函数或者策略以表格形式进行存储，很难应对动作空间很大或者连续控制的场景，因此对值函数或者策略进行近似是一个有效的方法。

与传统方法不同，深度强化学习方法采用深度神经网络对 Q 值进行估计、或者直接对策略进行估计，因此称作深度强化学习，强化学习与深度神经网络的结合是近年来人工智能领域最具突破的研究之一，在多个领域取得了瞩目的成功。

具体地，在深度强化学习中，强化学习的以下元素通过深度神经网络进行参数化：值函数 $\hat{v}(s;\boldsymbol{\theta})$ 或者 $\hat{q}(s,a;\boldsymbol{\theta})$，策略 $\pi(a \mid s;\boldsymbol{\theta})$，以及马尔可夫过程的状态转移函数以及奖励函数。其中 $\boldsymbol{\theta}$ 为深度神经网络的权值参数，通常采用梯度下降方法对神经网络参数进行更新，从而使得智能体的动作选择能够最大化奖励值。

例如 Deep Q-learning 采用深度神经网络对 Q 函数进行参数化近似，并引入了经验重播（Experience Replay）和目标网络（Target Network）等技巧提高了智能体的探索和利用效率。Actor-Critic 方法采用深度神经网络对智能体策略进行参数化，Actor 神经网络对策略进行近似，Critic 神经网络对值函数进行近似，两个神经网络同时更新，从而得到最优的策略近似。

2.2 基于深度强化学习的组合优化框架

从上述内容可以发现，可以采用序列到序列模型、图神经网络模型等深度神经网络模型对组合优化问题进行建模，然后采用深度强化学习方法对模型进行训练，从而使得模型具有求解组合优化问题的能力。本节首先对基于深度强化学习的组合优化建模和求解流程进行介绍，其次对建模和求解流程的各个细节进行展开，最后提出基于深度强化学习的组合优化框架。

2.2.1 基于深度强化学习的组合优化建模和求解流程

采用深度强化学习方法求解组合优化问题主要分为三步：1）首先，根据组合优化问题特征进行模型选择；2）其次，设计深度神经网络对组合优化问题进行建模；3）最后，采用强化学习方法对深度神经网络参数进行学习。

（1）模型选择

首先，不同组合优化问题呈现出不同的特点，需要根据不同组合优化问题的特点进行模型选择。针对具有序列特性的组合优化问题，例如凸包问题、工作流调度问题、TSP 问题以及 VRP 问题等路径优化问题，该类问题具有序列的特性，即问题的解中节点的顺序对解的好坏有很大影响，例如 TSP 问题城市访问的顺序影响路径的长度。针对该类具有序列特性的组合优化问题，可以采用序列到序列模型对其进行建模，从而有效提高模型对序列信息的处理能力。

对于另一类不需要考虑节点的顺序的点选择问题，例如集合覆盖问题、最大独立集问题、最小顶点最大覆盖等问题，由于该类问题一般具有图结构，因此可以选择图神经网络模型对该类问题进行建模。对既具有序列特性又具有图结构的组合优化问题，可以将序列到序列模型和图神经网络模型相结合，从而提高模型的表现。

（2）模型设计

其次，设计深度神经网络对组合优化问题进行建模。传统的序列到序列模型和图神经网络模型无法直接对组合优化问题进行求解，需要根据组合优化问题的特性对深度神经网络模型进行特殊设计，从而实现对组合优化问题的求解。采用人工智能方法求解组合优化问题的核心思路是利用深度神经网络实现组合优化问题的输入到输出解的映射，因此核心问题是如何利用深度神经网络对组合优化问题的解进行构造。

针对序列到序列模型，可以类比机器翻译的过程，即每次输出一个翻译的单词，可以参考该过程采用序列到序列模型构造组合优化的解，即每次向当前解添加一个决策变量，从而构造得到组合优化的最终解，但是传统序列到序列模型是每一步生成一个新的词向量，而组合优化问题是生成输入序列的排列组合，因此需要借鉴序列到序列模型对神经网络的结构进行重新设计。针对图神经网络模型，可以将组合优化问题中每个节点的特征当作图中节点的初始特征，节点之间的关系作为图中的边，采用图神经网络计算每个节点的特征向量，从而映射为节点选择的概率，根据概率则可以对该类点选择的组合优化问题进行求解。

但是直接应用上述模型无法实现对组合优化问题的求解，需要根据组合优化问题的具体特征进行设计，例如对于 VRP 问题，每一个城市选择的决策都会导致节点状态的动态变化，对于最大独立集问题，需要满足任意两点之间都没有连边

的约束，因此需要考虑组合优化问题的特征，基于此，章节 2.2.2 和章节 2.2.3 分别对基于序列到序列模型和图神经网络模型的组合优化问题建模流程和原理进行了展开和详细介绍。

（3）模型训练

深度神经网络模型实质上是一个非线性函数 $f_\theta(x)$，该函数能够实现组合优化问题输入到输出的映射，但是该函数中存在大量参数 θ，只有将这些参数设置为最优，才能实现准确的映射，因此需要对深度神经网络模型的参数进行训练。传统神经网络的训练方法一般为监督式训练方法，即提供训练集中每个实例的标签，以最小化神经网络的输出和实际标签之间的差距进行模型训练，对于组合优化问题，标签就是组合优化问题的最优解，但是组合优化问题本来就面临求解速度慢的局限性，因此很难提供大量用于训练的标签，如果将近似最优解作为标签，那么训练得到的模型性能无法超越训练集近似最优解的性能，因此传统监督训练方法面临着局限性。

强化学习是无监督训练方法，可以直接以最小化组合优化目标函数的方法进行训练，不需要构建标签数据集，并且具有更强的探索能力，因此可以采用强化学习方法对组合优化的深度神经网络模型进行训练，强化学习状态、反馈的设计需要结合组合优化问题的特征，章节 2.2.4 对采用强化学习方法训练组合优化神经网络模型的流程进行了阐述。

2.2.2 基于指针网络的组合优化建模

本节对基于序列到序列模型的指针网络模型进行介绍，该方法可以实现对具有序列特性的组合优化问题进行建模。指针网络模型借鉴序列到序列模型的思想，即利用神经网络实现序列到序列的映射，其核心思路是：利用编码器对组合优化问题的输入序列进行编码，得到一个包含输入序列信息的向量，再利用解码器对该向量进行解码，结合一种新的注意力机制以自回归的方式逐步构造解，自回归即每一步向当前解中增加一个解，直到构造得到完整解。

具体地，定义一个组合优化问题 $P = P_1, P_2, \cdots, P_n$，其解为 $C^P = C_1, \cdots, C_{m(P)}$，例如，对于凸包问题，$P$ 为节点的坐标序列，解 C^P 为节点序号的排列组合。可以将该组合优化问题建模为问题序列 P 到解序列 C^P 的映射，因此需要构建一个参数为 θ 的深度神经网络模型，对该映射的概率分布 $p(C^P|P;\theta)$ 进行估计，具体地，可以通过条件概率的链式法则对 $p(C^P|P;\theta)$ 进行分解：

$$p\left(\mathcal{C}^{\mathcal{P}} \mid \mathcal{P};\theta\right) = \prod_{i=1}^{m(\mathcal{P})} p_\theta\left(C_i \mid C_1, \ldots, C_{i-1}, \mathcal{P};\theta\right) \tag{2.15}$$

神经网络模型的参数通过最大化条件概率的方式进行优化：

$$\theta^* = \arg\max_{\theta} \sum_{\mathcal{P},\mathcal{C}^{\mathcal{P}}} \log p\left(\mathcal{C}^{\mathcal{P}} \mid \mathcal{P}; \theta\right), \tag{2.16}$$

针对上述建模过程，可采用指针网络模型实现该序列到序列的映射。指针网络模型的结构如图 2.5 ，与序列到序列模型相同，指针网络由编码器、解码器和注意力机制组成，编码器和解码器均为循环神经网络 LSTM，即采用 LSTM 对 $p_\theta\left(C_i \mid C_1, \ldots, C_{i-1}, \mathcal{P}; \theta\right)$ 进行建模。

编码器 LSTM 将节点坐标进行编码，得到每个节点的编码器隐层状态 $e_1, \cdots, e_n$，并输出最后的定长向量，解码器 LSTM 对编码器输出的定长向量进行解码，每一步均输出解码器隐层状态 $d_1, \cdots, d_{m(P)}$，传统的序列到序列模型对定长向量进行解码，从而得到输出序列。注意力增强的序列到序列模型采用注意力机制提高解码过程信息表示的能力，在每一步解码过程 i 中，将当前步的解码器隐层状态 d_i 与所有编码层隐层状态 $e_1, \cdots, e_n$ 进行注意力计算，得到注意力值加权后的隐层状态，从而利用该注意力加权后的隐层状态进行序列输出，注意力计算公式如下：

$$\begin{aligned} u_j^i &= v^T \tanh\left(W_1 e_j + W_2 d_i\right) \quad j \in (1, \ldots, n) \\ a_j^i &= \operatorname{softmax}\left(u_j^i\right) \qquad\qquad\quad j \in (1, \ldots, n) \\ d_i' &= \textstyle\sum_{j=1}^{n} a_j^i e_j \end{aligned} \tag{2.17}$$

其中 softmax 函数对输出的注意力值进行正则化，v, W_1, W_2 为需要优化的神经网络参数，将注意力值加权的编码器隐层状态 e_i' 与解码器隐层状态进行拼接 d_i，从而进一步得到该步的输出值，该计算方法即为传统的注意力机制，在机器翻译等任务中具有广泛的应用。

但是针对组合优化问题，无法采用上述方法进行序列的映射，组合优化问题解的长度依赖于输入序列的长度，输出序列通常为输入序列中节点序号的排列组合，式 (2.17) 无法完成该过程。因此，指针网络模型对传统注意力计算方法进行了改进，使其可以求解组合优化问题。

具体地，指针网络模型采用式 (2.18) 对条件概率 $p\left(C_i \mid C_1, \ldots, C_{i-1}, \mathcal{P}\right)$ 进行计算：

$$\begin{aligned} u_j^i &= v^T \tanh\left(W_1 e_j + W_2 d_i\right) \quad j \in (1, \ldots, n) \\ p\left(C_i \mid C_1, \ldots, C_{i-1}, \mathcal{P}\right) &= \operatorname{softmax}\left(u^i\right) \end{aligned} \tag{2.18}$$

与传统注意力机制不同，指针网络模型的注意力机制去掉了式 (2.17) 中的第三个公式，没有进一步计算注意力加权的编码器隐层状态，而是将直接将第二个

公式计算得到的注意力值当作“指针”，指向输入序列的序号，从而实现“组合”优化，输出节点序号的排列组合。

模型参数通过最大化条件概率 $p\left(\mathcal{C}^{\mathcal{P}} \mid \mathcal{P}; \theta\right)$ 来确定，即给定输入序列 $P = P_1, P_2, \cdots, P_n$，选择能够提高解 $C^P = C_1, \cdots, C_{m(P)}$ 质量的参数 θ:

$$\theta^* = \arg\max_{\theta} \sum_{\mathcal{P}, \mathcal{C}^{\mathcal{P}}} \log p\left(\mathcal{C}^{\mathcal{P}} \mid \mathcal{P}; \theta\right), \tag{2.19}$$

以图 2.5 的凸包问题为例，利用指针网络模型构造解的过程如下[92]:

首先，将问题的每个输入（位置坐标）转换成高维的节点向量，编码器的 LSTM 依次读入各个节点的节点向量，最终编码得到一个存储输入序列信息的定长向量 Vector，同时 LSTM 在计算的过程中可以得到每个节点的隐层状态 $e_1, \cdots, e_n$，其过程如图 2.5 所示。

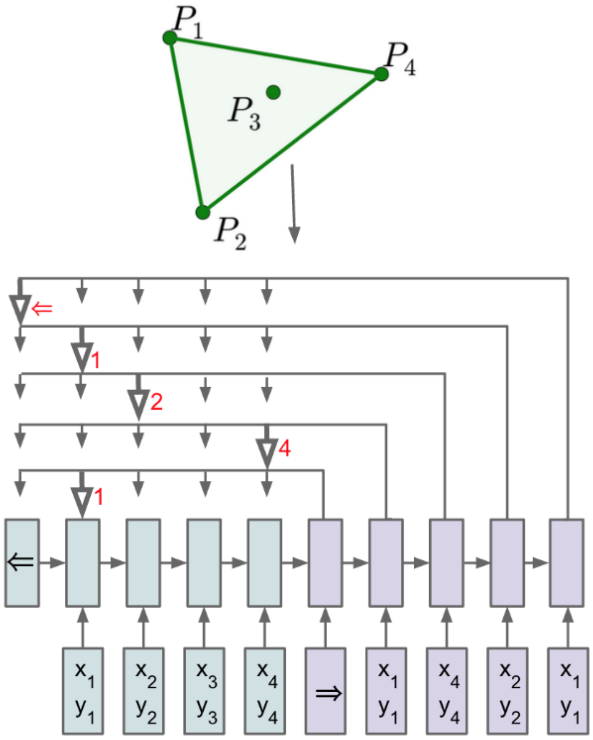


图 2.5　Pointer Network 模型示意图

然后，解码器对 Vector 进行解码，过程如图 2.5 所示。在第一步解码过程中，LSTM 读入 Vector，输出当前的隐层状态 d_0，利用式 (2.18) 的注意力机制根据 d_0 和编码器得到的各节点的隐层状态 e_1, e_2, e_3, e_4 可以计算得到选择各个节点的条件概率，此时可选择概率最大的节点 1 作为第一步选择的节点; 在接下来的解码过程中，LSTM 读入上一步 LSTM 的隐层输出和上一步选择节点 1 的特征向量，输

出当前的隐层状态 d_1，根据 d_1 和各节点的特征向量 e_1, e_2, e_3, e_4 计算选择各个节点的概率，利用式 (2.18) 的注意力机制进行该计算，此时可以选择具有最大概率值的节点 4 添加到解当中，按照该方式不断选择节点，直至构造得到一个完整解 $(1, 4, 2, 1)$。

因此，通过该指针网络模型可以实现组合优化问题的建模，通过对该模型参数的训练可以实现问题序列到解序列的准确映射，从而实现组合优化问题的求解。

2.2.3 基于图神经网络的组合优化建模

由于大多数组合优化问题都具有图特征，因此可以采用图神经网络对组合优化问题进行建模，本节对该方法的建模流程进行阐述。其核心思想是根据每个节点的原始信息（如城市坐标）和各个节点之间的关系（如城市之间的距离），利用图神经网络方法计算得到各个节点的特征向量。根据各个节点的特征向量进行节点预测、边预测等任务。

一般将图定义为 $G = (V, E)$，V 代表节点的集合，E 为边的集合。图神经网络通过不断学习节点的特征、邻居节点的特征、边的特征，并将其以各种方法进行聚合，从而最终得到各个节点的特征向量，根据各个节点的特征向量完成预测、分类等任务。以经典 GNN 为例，各个节点的表征根据式 (2.20) 进行更新：

$$\mathbf{h}_v^{(t)} = \sum_{u \in N(v)} f\left(\mathbf{x}_v, \mathbf{x}_{(v,u)}^{\mathrm{e}}, \mathbf{x}_u, \mathbf{h}_u^{(t-1)}\right) \tag{2.20}$$

其中 $\mathbf{h}_v^{(t)}$ 代表节点 v 的表征向量，$N(v)$ 代表 v 的邻居节点的集合，$\mathbf{x}_v$ 是节点 v 的特征，$\mathbf{x}_{(v,u)}^{\mathrm{e}}$ 是与 v 相连的边的特征，$\mathbf{x}_u$ 是邻居节点 u 的特征，$\mathbf{h}_u^{(t-1)}$ 是邻居节点 u 在上一步更新的特征向量。因此该公式根据节点 v 本身的特征、边的特征以及邻居节点的特征对节点 v 的表征向量进行更新，从 $t = 0$ 开始对不断对 $\mathbf{h}_v^{(t)}$ 进行更新直到收敛，从而得到节点 v 的 high-level 特征向量。

根据各个节点的特征向量，可以进行组合优化问题的求解，如针对节点选择问题（如最小顶点覆盖问题），可以将图神经网络得到的节点特征向量以一个全连接神经网络映射到节点选择概率，从而根据概率进行节点的选择；针对边选择问题（如 TSP 问题），可以以两个节点的特征向量作为输入，以一个全连接神经网络映射得到一个选择概率，即该两点之间存在边的概率，从而进行边选择。

值得注意的是，针对路径规划问题，例如旅行商问题，解构造过程存在一定的约束，按照概率进边的选择并不一定可以构成一个完整的哈密顿回路，因此需要辅以搜索方法进行解的构造，可以通过以下两种方法进行组合优化解的构造。

（1）基于自回归的方式构造组合优化的解

根据图神经网络可以计算得到节点和边选择的概率，第一种方法是根据节点

选择的概率，以自回归的方法构造解。具体地，模型首先利用 GNN 计算得到各个节点的向量表示，将各个节点的表征向量通过一个全连接神经网络映射为各个节点的 Q 值。根据 Q 值采用贪婪策略以迭代的方式构造解，即每次选择 Q 值最大的节点添加到当前解当中，直到构造得到完整解，通常以 DQN 强化学习方法对该图神经网络进行训练，从而得到准确的 Q 值估计。

该解构造方式的示意图如图 2.6 所示[36]，即每次从可选节点集合中选择一个节点，根据当前构造的解重新对图神经网络进行计算并更新 Q 值，直到构造得到一个完整解。

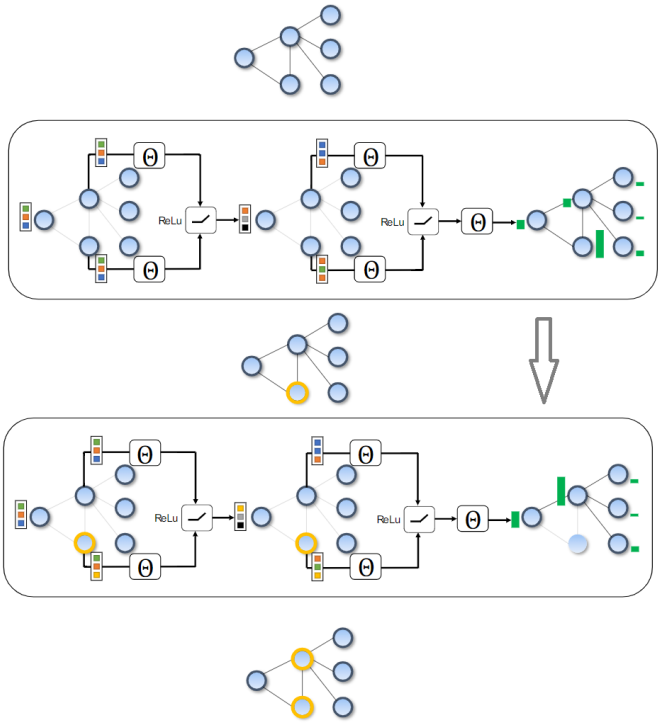


图 2.6　自回归方式构造组合优化解过程示意图

（2）基于非自回归的方式构造组合优化的解

第二种方式是根据图神经网络一次性计算得到所有边选择的概率，而不通过自回归的方式逐步进行边的添加，基于图神经网络的边选择概率计算方法如

式 (2.21) 所示[40]。

$$
\begin{aligned}
x_i^{\ell+1} &= x_i^{\ell} + \mathrm{ReLU}\left(\mathrm{BN}\left(W_1^{\ell} x_i^{\ell} + \sum_{j\sim i} \eta_{ij}^{\ell} \odot W_2^{\ell} x_j^{\ell}\right)\right) \\
\eta_{ij}^{\ell} &= \frac{\sigma\left(e_{ij}^{\ell}\right)}{\sum_{j'\sim i} \sigma\left(e_{ij'}^{\ell}\right) + \varepsilon} \\
e_{ij}^{\ell+1} &= e_{ij}^{\ell} + \mathrm{ReLU}\left(\mathrm{BN}\left(W_3^{\ell} e_{ij}^{\ell} + W_4^{\ell} x_i^{\ell} + W_5^{\ell} x_j^{\ell}\right)\right) \\
p_{ij} &= MLP(e_{ij}^{T})
\end{aligned}
\tag{2.21}
$$

即进行节点特征向量更新的同时，进行边特征向量的更新，在 T 次更新后，将边特征向量映射为一个概率值。该概率值代表在最优解中，各个边属于最优解边集合的概率，此时采用贪婪策略根据该概率进行边的选择一般会出现不可行解的情况，以旅行商问题为例，每次选择一个边，最终所有边不一定可以构成一个哈密顿回路，因此根据该概率分布，进一步以波束搜索、树搜索的方式进行可行解的构造。

波束搜索是广度优先搜索的一种特殊形式，在图的搜索空间较大时，为了提高搜索效率缩减搜索空间，每次只搜索质量较高的一些节点，波束搜索的一个重要参数是波束宽度 L，广度优先搜索遍历完当前所有子节点后进入下一层，而波束搜索在该层只处理质量最高的前 L 个节点，而剪掉其余节点，然后再往下一层拓展，当波束宽度无穷大时，即为广度优先搜索。以旅行商问题为例，给定当前可选边或者节点的集合，选择前 L 个概率最大的边或者节点，对这 L 个边或者节点继续进行扩展，对每条路径均进行存储，直到构造得到完整解，选择最终累计概率最大的解作为最终解。与之相对的则是贪婪策略，即每次只选择概率值最大的边或者节点，从而构造得到最终解，波束搜索可以提高解的质量，虽然相较于广度优先搜索降低了运算量，但仍然需要较长的搜索时间。

由于图神经网络主要用于计算节点的特征向量，因此另外一种方法是将图神经网络和指针网络进行结合，即用图神经网络计算得到的节点特征向量代替指针网络 LSTM 编码器计算得到的各节点的隐层输出向量，仍然采用式 (2.18) 的注意力机制计算每一步的节点选择概率，以自回归的方式逐步构造得到完整解。

2.2.4 强化学习训练方法

通过上述指针网络或者图神经网络对组合优化问题进行建模，可以用参数化的深度神经网络实现组合优化问题从输入到输出的非线性映射，对于深度神经网络模型的参数，可以通过监督式训练算法或者强化学习算法进行训练。

监督式学习方法通过使用“组合优化问题 - 最优解”的标签数据集对神经网

络的参数进行训练，该训练方法存在以下缺陷：1）首先深度神经网络的参数数量巨大，监督式训练需要大量带标签的数据集，但是组合优化问题求解本就耗时，构建不同组合优化问题的标签数据集需要耗费海量时间和存储空间，很难实际应用；2）其次，虽然可以通过计算近似最优解的方式构造标签数据集，但是在该种情况下监督式训练得到的组合优化策略无法超越数据集中的最好结果，即模型最好情况下也只能达到标签数据集的优化能力。在该种情况下，采用无监督的强化学习算法对模型参数进行训练是一个更好的选择。

强化学习通过试错机制不断训练得到最优策略，首先需要将组合优化问题建模为马尔科夫过程，其核心要素为状态、动作以及反馈，以旅行商问题为例，其问题的状态为城市的坐标 s 以及已经访问过的城市，动作为第 t 步选择的城市 π_t，所有动作组成的城市访问顺序 π 即为组合优化问题的解，反馈 r 是路径总距离的负数，即最小化路径长度，策略即为状态 s 到动作 π 的映射，策略通常为随机策略，即得到的是节点选择的条件概率 $p_\theta(\pi|s)$，该随机策略建模为:

$$p_\theta(\boldsymbol{\pi} \mid s) = \prod_{n}^{t=1} p_\theta\left(\boldsymbol{\pi}_t \mid s, \boldsymbol{\pi}_{1:t-1}\right) \tag{2.22}$$

策略由神经网络参数 θ 进行参数化，在马尔科夫过程中，每一步动作的概率为 $p_\theta\left(\boldsymbol{\pi}_t \mid s, \boldsymbol{\pi}_{1:t-1}\right)$，即根据已访问过的节点 $\pi_{1:t-1}$ 和城市坐标 s 计算选择下一步访问各个节点的概率，根据链式法则累乘即可以得到问题输入 s 到节点最终访问顺序 π 的映射 $p_\theta(\pi|s)$，该随机策略即通过上述的指针网络或者图神经网络进行建模，其参数为 θ。

由于旅行商问题的优化目标是最小化总的路径长度 $L(\pi)$，因此可以采用 REINFORCE 强化学习算法对模型参数进行训练，REINFORCE 算法属于策略梯度方法，即对策略进行参数化的近似，在此采用上述深度神经网络对策略进行近似，REINFORCE 算法是以回合制的总反馈作为参数更新的标准，天然符合组合优化问题，因此采用 REINFORCE 强化学习算法对策略参数进行训练优化。

REINFORCE 强化学习算法又称基于蒙特卡洛的策略梯度方法，即不断执行动作直到结束，在一个回合结束之后计算总反馈，然后根据总反馈对策略的参数进行更新，适用于训练组合优化问题的模型，以 TSP 问题为例，总反馈即为路径总长度的负数 $-L(\pi)$。REINFORCE 基于以下公式对策略的参数 θ 进行更新:

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) &= \mathbf{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)}\left[\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid s) L(\boldsymbol{\pi})\right] \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \end{aligned} \tag{2.23}$$

根据链式法则，$p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid s)$ 为每步动作选择概率的累乘，则 $\log p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid s)$ 计算为每步动作选择概率对数的求和，以该值对参数 $\boldsymbol{\theta}$ 计算偏导可得梯度值

$\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid s)$。

通常引入一个基线（baseline）函数 $b(s)$，满足：

$$\begin{aligned} \sum_{a} b(s) \nabla_{\theta} \pi(a \mid s, \theta) &= b(s) \nabla_{\theta} \sum_{a} \pi(a \mid s, \theta) \\ &= b(s) \nabla_{\theta} 1 = 0 \end{aligned} \tag{2.24}$$

因此减去与动作无关的项 $b(s)$，可以减少算法的方差，因此策略参数的更新公式为：

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) &= \mathbf{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)} \left[\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid s) \left(L(\boldsymbol{\pi}) - b(s) \right) \right] \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \end{aligned} \tag{2.25}$$

$L(\boldsymbol{\pi}) - b(s)$ 决定了梯度下降的方向，$b(s)$ 代表策略的平均表现 (baseline)，如果当前策略的表现比”平均”好，则对该策略进行正向激励，反之亦然。

有多种方式对 $b(s)$ 进行估计，运用较多的方法是新增一个 Critic 神经网络对 $b(s)$ 进行估计，即给定一个组合优化问题实例 s，利用 Critic 神经网络估计该问题解的目标函数值。Critic 网络与策略网络同步进行训练，以策略网络训练过程中产生的 $s, L(\boldsymbol{\pi})$ 作为训练集对 Critic 进行训练。REINFORCE 算法通过公式2.25对的梯度进行计算并更新，不断训练从而对 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi} \mid s)$ 进行准确的估计，从而实现组合优化问题序列到解序列的准确映射。

2.2.5 基于深度强化学习的组合优化框架

综上，本节提出基于深度强化学习的组合优化框架，如图 2.7 所示，基本流程如下。

(1) 模型设计

在模型设计阶段，首先根据组合优化问题的类型对模型进行选型。主要考虑组合优化问题是否具有序列特性或者图特征，对于具有序列特性的组合优化问题，可以选择指针网络模型对问题进行建模，尤其对于复杂的路径选择问题，例如 VRP 问题，指针网络模型是更好的选择，通过指针网络模型，基于注意力计算机制，可以得到每一步选择各个节点的概率，从而根据概率，采用贪婪策略、波束搜索、树搜索等方式构造解。

对于具有图特征的问题，例如集合覆盖问题、最大独立集问题、最小顶点最大覆盖问题等，可以选择图神经网络模型对该类问题进行建模，利用图神经网络计算方法计算得到节点的向量表示，进一步采用全连接神经网络等方式将节点向量映射为点或边的选择概率，从而根据点选择 / 边选择概率构造组合优化问题的解。

对于同时具有序列特性和图特性的问题，可以将图神经网络和指针网络的注

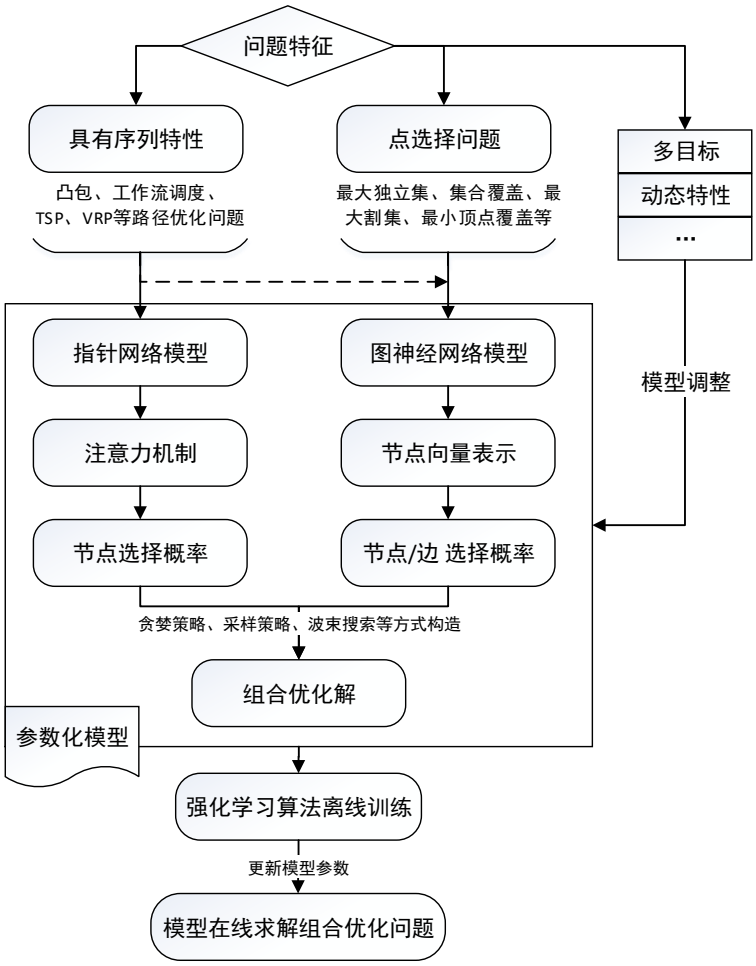


图 2.7　基于深度强化学习的组合优化求解框架

意力机制相结合，将指针网络的编码器替换为图神经网络，从而得到更加准确的节点特征向量，然后以注意力机制进行解的构造，从而提高模型性能。同时，针对不同组合优化问题的其他特征，例如问题可能存在的动态特性或者多目标特性，需要针对性地对神经网络模型进行调整，以适应不同组合优化问题的特点。

（2）模型训练

模型建立完成之后，采用强化学习方法对神经网络模型的参数进行优化。首先，随机生成一定规模的组合优化问题实例 s，利用神经网络模型计算问题 s 的解，并计算该解的目标函数值 $L(\pi)$，根据 s 和 $L(\pi)$，根据问题特征选择基于蒙特卡洛或者时序差分的强化学习方法对模型进行训练。

蒙特卡洛学习方法在回合完成之后进行模型参数的更新，采用最终的总回报值（即组合优化问题真实的目标值）对策略进行更新，对于组合优化目标函数值的估计是无偏的，因此在决策步数少的小规模问题上具有较好的模型训练效果，但是针对大规模问题，该训练方式具有较大的时间复杂度和空间复杂度，训练效

率较低。时序差分学习方法在每步动作执行之后进行策略更新，在大规模问题上训练效率较高，但是对于组合优化问题目标值的估计存在较大偏差，因此需要根据问题的特征进行试验和选型。

（3）构造组合优化解

模型一旦离线训练完成，即可利用该模型进行组合优化问题的在线求解。神经网络模型的输出通常为各个节点 / 边的概率分布 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$，基于 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$，可以通过以下方式进行组合优化解的构造：

（a）贪婪策略。在解构造过程中，每一步都根据可选节点 / 边的概率分布 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$ 选择概率最大的节点 / 边，不断选择直至构造得到完整解，例如求解旅行商问题，指针网络模型每一步输出选择各个节点的概率，每次均选择概率最大的城市添加到当前解当中，最终构造得到一个完整的哈密顿回路。

（b）采样策略。在解构造过程中，每一步从概率分布 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$ 中进行随机采样，不断添加到当前解中从而得到一个完整解。该采样过程进行 N_{sample} 次可得到 N_{sample} 个解，选择其中最好的解作为最终解。可见贪婪策略只构造一个解，而采样策略需要构造多个解，因此采样策略需要更长的计算时间，但是得到的解通常更优。

（c）波束搜索。波束搜索是广度优先搜索的改进方法，即采用搜索树的方式对解空间进行广度优先遍历，但是搜索过程中只保留前 N_{beam} 个最好的部分解，只对该 N_{beam} 个节点进行探索，根据各个节点 / 边的概率值对这 N_{beam} 个搜索树节点进行选择。当 N_{beam} 为无穷大时，波束搜索退化为广度优先搜索，即遍历整个决策空间。

采样策略和波束搜索策略通常能够得到比贪婪策略更优的解，采样数量 N_{sample} 以及波束宽度 N_{beam} 的不同则会导致不同的计算耗时和优化效果，可以根据计算资源、优化性能以及求解实时性的要求进行选择。通过该模型设计、模型训练、解构造的求解框架，即可基于深度强化学习方法实现组合优化问题的在线求解。

2.3 本章小结

本章对基于深度强化学习的组合优化的基本原理进行了阐述，介绍了深度神经网络模型、组合优化问题、强化学习方法等概念，对基于指针网络的组合优化建模方法和基于图神经网络的组合优化建模方法进行了阐述，提出了基于深度强化学习的组合优化基本框架和求解范式。

第三章　基于深度强化学习的覆盖旅行商组合优化方法

当前采用人工智能技术求解组合优化问题的方法大多针对较为简单的组合优化问题，如旅行商问题，对于复杂组合优化问题的研究较少，而且模型能够处理的问题规模通常较小，如大多数当前方法至多能够求解 100 城市的旅行商问题，对于大规模复杂组合优化问题的研究存在欠缺。

本章针对复杂单目标组合优化问题，提出了一种基于深度强化学习的覆盖旅行商组合优化方法，对具有动态特性、解序列长度不定的单目标覆盖旅行商问题进行研究，克服了当前人工智能模型在处理大规模、动态复杂特性问题上的局限，所提方法相对于传统启发式方法能够获得 10 倍以上求解速度的提升，模型表现优于当前业界最优的其他人工智能模型，同时可以拓展到更大规模、不同类型的覆盖旅行商问题上。

3.1　覆盖旅行商问题

旅行商问题（Traveling Salesman Problem，TSP）是运筹学领域经常研究的组合优化问题。给定一组城市的空间位置，旅行商问题的目标是找到一个最短距离的哈密顿回路，即访问每个城市一次并返回出发点。旅行商问题是覆盖旅行商问题（Covering Salesman Probelm，CSP）的一类特殊情况，覆盖旅行商问题是旅行商问题的推广。

在覆盖旅行商问题中，每个城市都有一个预定义的城市覆盖距离，在该距离内所有其他城市都会被覆盖。覆盖旅行商问题即寻找给定城市的一个子集，寻找访问该子集城市的一个最短回路，使得每个城市必须被访问一次或者至少被一个城市覆盖。具体地，针对覆盖旅行商问题，定义一个完全图 $G = (V, E)$，图中每个节点均互相连接，其中 V 是代表城市的节点集合。$E = (\{i, j\} : i, j \in N, i < j)$ 是边的集合，c_{ij} 表示边 $\{i, j\}$ 的权重，通常定义为节点 i 和 j 之间的距离，S_i 为 N 的子集，每个节点 i 可以覆盖 S_i 内的所有城市。在覆盖旅行商问题中，目的是在 V 的子集上找到一个哈密顿回路，使得所有城市被访问一次，或者至少被回路上任何一个城市覆盖一次，该问题的目标是最小化总路径长度。

如果每个城市的覆盖距离为零，则覆盖旅行商问题就约简成了旅行商问题。因此，覆盖旅行商问题是旅行商问题的超集，旅行商问题是覆盖旅行商问题的特例，覆盖旅行商问题也属于 NP 难问题并且比旅行商问题更难解决。

在一些实际应用场景中，由于燃料或人力资源的限制，很难保证每个城市都能像旅行商问题中假设的那样被访问一次。因此，覆盖旅行商问题广泛存在与现实世界的实际应用中，诸如应急管理和救灾规划问题 [93–95]。例如，在医疗服务队

的路线规划问题[93] 中，没有必要对每个村庄进行医疗物质的输送，而是选择部分村庄进行医疗物质的输送，其余村庄前往其最近的存在寻求医疗服务，从而提高输送效率。

覆盖旅行商问题由 Current 和 Schilling [93] 于 1989 年首次提出。他们提出了一种简单的启发式方法来求解该问题，求解过程分为两部分：第一阶段计算可以覆盖所有城市的最少数量的城市，即集合覆盖问题（SCP）；第二阶段可以看作是一个旅行商问题，即在上述选定的城市中找到最短的回路。但是集合覆盖问题通常存在多个解，因此需要多次运行 TSP 算法才能找到最终解，求解效率低且效果差。

自此，学者们开始设计不同的启发式方法来求解覆盖旅行商问题。Golden 等人[96] 提出了两种启发式局部搜索算法（LS1 和 LS2）来求解该问题，LS1 和 LS2 是当前应用最为广泛的方法，并常被用作模型评价的基准算法。作为局部搜索方法，它们都是从一个随机的可行解开始，利用破坏、修复和置换等不同的算子来改进它。LS1 首先根据删除概率从当前解中删除固定数量的节点，然后根据插入概率将新节点逐个地插入到当前的解中，直到解决方案再次可行，这些概率值通过将节点删除或添加到当前解中路径长度的减少或增加来计算得到。相反的，LS2 每次从当前解中删除一个节点，然后将距离被删除节点最近的节点插入到解中，然后采用 Lin-Kernighan 算法和 2-Opt 算法对解进行改进。

Salari 等人[97] 开发了一种基于整数线性规划（ILP）的启发式方法来解决覆盖旅行商问题。它与上述方法[96] 的主要区别在于其应用 ILP 来改进解。这种方法首先应用类似的破坏和修复操作，通过从当前解决方案中删除一些节点并重新插入新节点以形成可行解的方式来减少路径长度。通过求解一个以最小化路径为目标的 ILP 模型，对当前解进一步改进。另外，其他局部搜索方法也应用类似的破坏和修复操作来求解覆盖旅行商问题。例如，Shaelaie 等人[98] 提出了一种变邻域搜索方法，Venkatesh 等人[99] 提出了一种多起点迭代局部搜索算法。

除了局部搜索方法之外，其他基于种群的启发式方法也被相继提出。Salari 等人[100] 结合蚁群优化（ACO）算法和动态规划来求解覆盖旅行商问题。它还结合了各种本地搜索方法，例如删除、插入和 3-OPT 搜索。实验表明，这种方法在大规模问题上存在优势。Tripathy 等人[101] 提出了一种重新设计的交叉算子的遗传算法（GA），但模型表现不如 LS1 和 LS2。文献中还存在其他不同类型的遗传算法[98, 102] 用于求解覆盖旅行商问题，Pandiri 等人[103] 提出了一种人工蜂群算法等等。

启发式方法是近几十年来求解覆盖旅行商问题的主要方法，在性能方面，上述启发式算法在大多数情况下与早先提出的 LS1/LS2 并不存在明显的优势，一些启发式方法可能在几个问题实例上优于 LS1/LS2，但是仍然属于迭代型优化算法，

存在求解耗时时间长等缺陷。近年来开始出现基于机器学习求解组合优化问题的一系列方法，一定程度上克服了传统算法的局限性，但是当前研究大多针对诸如旅行商问题的小规模简单组合优化问题，而覆盖旅行商问题相对于 TSP 问题具有更多复杂特性，传统基于深度强化学习的组合优化方法在求解覆盖旅行商问题上面临多种挑战，首先覆盖旅行商问题解序列的长度可变，其次城市访问过程中节点覆盖状态会发生动态变化，如何对城市访问过程中节点覆盖情况的动态变化进行建模是一个需要解决的问题，模型不仅需要捕捉城市空间位置表示的静态特征，并且需要捕捉城市访问过程中覆盖状态变化导致的动态特征。

3.2 基于深度神经网络的覆盖旅行商问题建模

由问题分析可知，覆盖旅行商问题具有明显的序列特性，并且节点的状态在决策过程中会发生动态变化，因此根据第二章的求解框架，适合采用指针网络模型架构对该问题进行建模，并且需要将动态特性的处理机制加入到神经网络的模型结构设计当中。

针对具有 n 个城市的 CSP 问题 s，模型的参数化表示如下，在 CSP 中，城市 i 的特征 $\mathbf{x}_i$ 由其地理位置定义，即其 x 坐标和 y 坐标，解 $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_k, k \leq n)$ 是城市子集的序号组合，解需要满足所有的城市都应该被访问或者被覆盖。为了对 CSP 问题进行求解，需要设计参数为 θ 的深度神经网络模型，该模型以覆盖旅行商问题 s 的城市位置作为输入，输出序列 $\boldsymbol{\pi}$，该模型策略定义为 $p(\boldsymbol{\pi}|s)$，可以实现 s 到 $\boldsymbol{\pi}$ 的映射：

$$p_\theta(\boldsymbol{\pi}|s) = \prod_{t=1}^{k} p_{\boldsymbol{\theta}}\left(\pi_t|s, \boldsymbol{\pi}_{1\sim t-1}\right), k \leq n. \tag{3.1}$$

由于选择下一个城市的概率与之前已选择城市 $\boldsymbol{\pi}_{1\sim t-1}$ 的信息密切相关，即具有明显的序列特性，因此本章采用指针网络模型架构对覆盖旅行商问题进行建模，具体地，对模型的编码器和解码器结构均进行了特殊设计，提出了一种基于动态嵌入的注意力模型，在编码器，采用了基于多头注意力机制的静态嵌入用于处理问题的静态特征，并设计了一种动态嵌入用于处理问题的动态特征；在解码器，结合了循环神经网络和注意力机制对解码器进行了重新设计，以提高模型表现。接下来对该深度神经网络模型的结构进行详细介绍。

3.2.1 模型整体架构

本节对模型的整体架构和模型中的一些基本元素进行介绍。

3.2.1.1 指针网络模型架构

采用深度学习方法对组合优化问题建模，首先需要对问题的输入进行处理，以适合深度神经网络的输入结构，因此，给定节点 i 的二维输入（例如其位置的二维坐标），首先需要将其映射为一个高维向量，具体地，定义节点 i 的初始特征向量为维度为 d_h 的高维向量，是节点 i 的问题输入（例如其位置的二维坐标）到 d_h 维的线性映射：

$$\mathbf{X}_i = \mathbf{W}^{\mathbf{x}} x_i \tag{3.2}$$

本章设定 $d_h = 128$，该过程通过一个基于全连接神经网络的线性映射实现节点特征向量的初步提取。

本章采用的指针网络模型架构如图 3.1 所示。它的基本结构由编码器和解码器串联而成。编码器用于将节点嵌入编码为 *Vector*，而解码器用于将 *Vector* 解码为输出序列。编码器或解码器均为神经网络，本章中解码器为循环神经网络 (RNN)，编码器的具体设计将在后文介绍。在使用编码器对输入进行编码时，为每个城市 i 生成隐层向量 e_i。这里，e_i 可以解释为城市 i 的特征向量。编码器的输出是 *Vector*。

如图 3.1 所示，*Vector* 用作解码器 RNN 的输入，解码过程顺序进行。*Vector* 作为初始隐层向量 d_0，用于选择第一个访问的城市。下一个隐层向量 d_1 是解码器的循环神经网络进行计算得到。在解码步骤 t 中，城市选择的概率由编码器状态 d_{t-1} 和每个城市的特征向量 e_i 计算。RNN 读取选择城市的节点特征向量并输出当前解码器状态 d_t，然后用于下一个解码步骤。城市解码概率的计算通过如下方式计算。

$$\begin{aligned} u_i^t &= v^T \tanh\left(W_1 e_i + W_2 d_t\right) \quad i \in (1, \ldots, n) \\ P\left(y_{t+1} | y_1, \ldots, y_t, X\right) &= \operatorname{softmax}\left(u^t\right) \end{aligned} \tag{3.3}$$

其中 v, W_1, W_2 是可学习的参数。在步骤 t 中，可以选择概率最大的城市作为要访问的城市。

例如，如图 3.1 所示，为了在解码步骤 $t = 1$ 中选择第一个访问的城市，u_1^1, u_2^1, u_3^1 根据式 (3.3) 由 (d_0, e_1)、(d_0, e_2) 和 (d_0, e_3) 计算。选择 u^1 最大的一个，即 u_2^1 作为第一个城市，即城市 2。通过将城市 2 的节点嵌入输入解码器 RNN，可以得到 d_1。在选择下一个城市的过程中，根据在 (d_1, e_1)、(d_1, e_2) 和 (d_1, e_3) 上应用注意力计算公式，可以得到城市选择概率，此时可以选择概率最大的城市 1。该过程一直循环直到构成一个完整解。通过离线方式对参数 v, W_1, W_2 进行训练可以实现问题的求解。

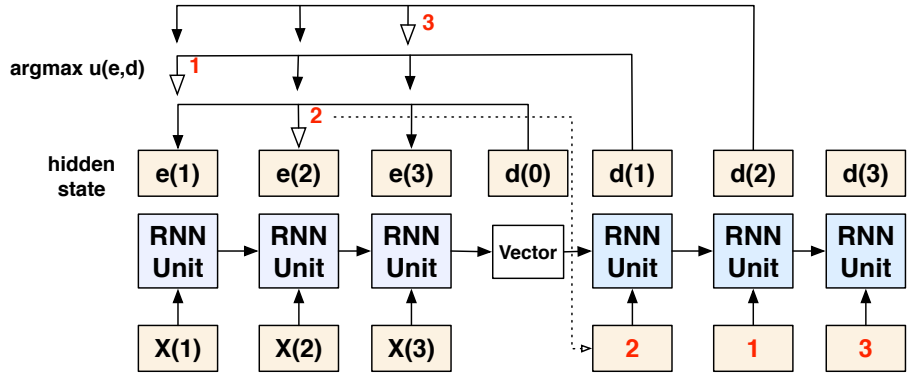


图 3.1　求解覆盖旅行商问题的编码器 - 解码器架构

3.2.1.2　自注意力机制

在上面的指针网络中，编码器的主要作用是在编码器中得到每个城市 i 的特征向量 e_i。近年来随着注意力机制的发展，自注意力（Self-attention）计算模块可以更好地提取问题的特征。自注意力机制是 Transformer 模型[104] 的基本组件，它是当前机器翻译等序列学习领域应用最广泛的模型之一。

为了对覆盖旅行商问题的特征进行更加有效的提取，本章将自注意力机制应用到编码器的设计中，计算每个节点的自注意力值作为节点的特征向量。通过该方式，每个节点 i 的特征向量不仅涵盖了自身的特征，还涵盖了该节点与其他节点之间的关系，这对于覆盖旅行商问题是尤为重要的，因为各个节点之间的距离关系是一个重要的特征，因此，自注意力机制可以更加有效地捕捉问题的结构模式，例如地理位置特征。

传统注意力机制可以通过另一种方式进行描述，例如式 (3.3) 中使用的注意力机制，第 t 步城市选择步骤中的 d_t 可以看作是一个“查询 *query*”，它包含已访问城市的信息。每个城市 i 的特征向量 e_i 可以看作是“键 *key*”，注意力操作就是通过 *query* 对所有城市的 *key* 进行查询，如果 *key* 与城市 i 的 *query* 值更匹配，则城市 i 将获得更多的注意力，在式 (3.3) 中注意力值被表示城市选择的概率。

与式 (3.3) 中的注意力计算方法不同，自注意力机制使用 *Scaled Dot-Product Attention* 注意力计算方式：注意力值由 query 值 $\mathbf{q}_t$ 和 key 值 $\mathbf{k}_i$ 的矩阵乘法 $\mathbf{q}_t^T\mathbf{k}_i$ 计算得到。自注意力机制的工作原理是计算“输入序列”中每个节点相对于其他节点的注意力值，并将节点 i 的注意力值作为其特征向量。通过该方式，与原始序列相比，处理后的序列不仅包含各节点自身的特征，还包含该节点与其他节点之间的关系特征。

具体地，自注意力由 *query* 和 *key-value* 键值对进行计算，*query* 和 *key-value* 键值对均由输入序列进行线性变换后构成，因此称为“自”注意力，如果 *value* 的 *key* 与 *query* 更匹配，则为给与 *value* 更多“注意力”。具体计算方法如下，首先计

算 *query*、*key* 和 *value* 值，即输入序列的线性变换：

$$\mathbf{Q} = W^Q\mathbf{X}, \quad \mathbf{K} = W^K\mathbf{X}, \quad \mathbf{V} = W^V\mathbf{X}. \tag{3.4}$$

其中 W^Q，W^K，W^V 为全连接神经网络的权重值，第 i_{th} 个节点的 *query*、*key* 和 *value* 为：

$$\mathbf{q}_i = W^Q\mathbf{X}_i, \quad \mathbf{k}_i = W^K\mathbf{X}_i, \quad \mathbf{v}_i = W^V\mathbf{X}_i. \tag{3.5}$$

第 i 个节点自注意力值计算过程如图 3.2 所示。首先，第 i 个节点与其他节点的关系特征由第 i 个节点的 *query* 与所有其他节点的 *keys* 的匹配度函数计算得到：$\mathbf{q}_i\mathbf{K}^T$。获得的匹配度值作为权重并通过 softmax 函数进行归一化。注意力值最终计算为 *values* 值 $\mathbf{V}$ 的加权和，其中权重为 $\mathbf{q}_i\mathbf{K}^T$。因此，第 i 个节点的注意力值是 $\mathbf{q}_i\mathbf{K}^T\mathbf{V}$，它与它的节点初始特征向量 $\mathbf{X}_i$ 具有相同的维度（d_h）。通过自注意力计算，第 i 个节点的注意力值不仅存储了它自己的特征，还存储了它与其他节点的关系特征。

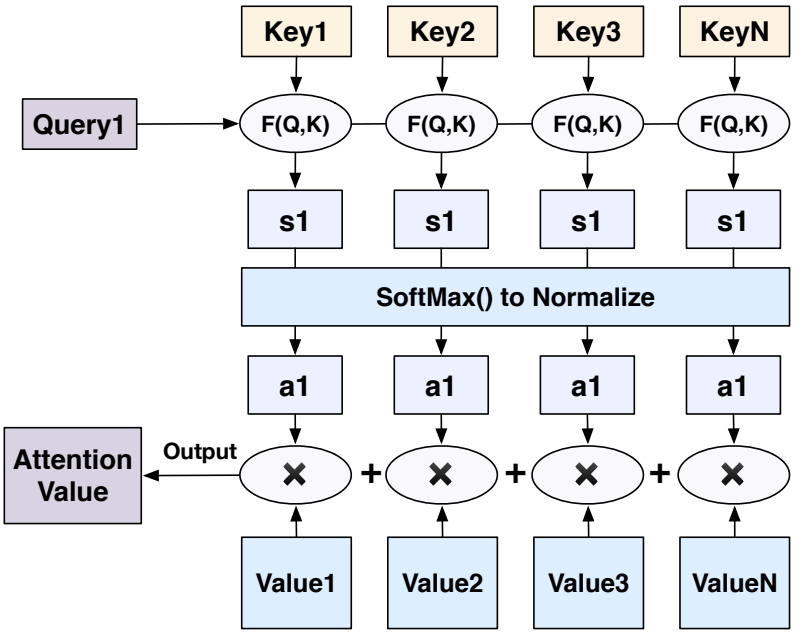


图 3.2　自注意力计算过程示意图

在实际基于 GPU 的编程实现中，通过矩阵乘法同时计算所有节点的注意力值：

$$\text{Attention}\,(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{3.6}$$

其中 $\sqrt{d_k}$ 是缩放因子，d_k 是 *key* 的维度。

3.2.1.3 多头注意力机制

可以将上述注意力计算过程执行多次，从而得到多个具有不同表现的特征，这被称作多头注意力（Multi-Head Attention）。首先，将 d_h 维特征向量截断为 M 个 $\frac{d_h}{M}$ 维度的特征向量，每个特征向量采用不同的线性变换参数 W^Q，W^K，W^V 得到不同的 *query*、*key* 和 *value*，从而通过自注意力机制得到 M 个子特征向量，最后，将 M 个子特征向量进行拼接，得到最终的节点特征向量，该向量仍为 d_h 维，该过程即为多头注意力计算方法。

通过在编码器中使用该多头注意力机制计算方法，可以提取到组合优化问题更加多样的特征，从而捕捉问题丰富的结构模式特征。

3.2.2 编码器结构

模型由编码器和解码器构成，编码器的目的是为了输出每个节点的特征向量，即产生每个节点的 *key* 值。本章所提模型的 *key* 由两部分组成：静态嵌入和动态嵌入。

3.2.2.1 静态嵌入

静态嵌入代表组合优化问题的静态特征，例如覆盖旅行商问题中城市的地理位置信息。静态嵌入的构造方法类似于文献 [105, 106] 中针对 TSP 问题的编码过程，本章采用上节介绍的多头注意力机制来构造静态嵌入，具体构造过程如图 3.3 所示，详细介绍如下：

静态嵌入的计算架构由 N 个顺序连接的“注意力层”组成，每个“注意力层”都具有相同的结构，通过串联的多个“注意力层”将节点特征向量顺序地进行 N 次处理。每个注意力层包含两个子层：第一个是多头注意力层，即采用上节介绍的多头注意力机制进行节点特征向量的计算；第二个是基于全连接神经网络的前馈层，用于增加模型深度。此外，每个子层都添加了残差连接[107] 和层归一化，因此，每个子层的输出为 $Norm(x + Sublayer(x))$。

计算静态嵌入通过以下两个步骤：

Step 1. 首先将 d_x 维城市坐标 ($d_x = 2$) 映射到 d_h 维节点初始特征向量 ($d_h = 128$)：

$$\mathbf{h}_i^{(0)} = W^{\mathbf{x}}\mathbf{x}_i + \mathbf{b}^{\mathbf{x}}. \tag{3.7}$$

$W^{\mathbf{x}}$ 和 $\mathbf{b}^{\mathbf{x}}$ 是可学习的参数。假设有 n 个城市，则节点初始特征向量的形状为 $n \times d_h$。

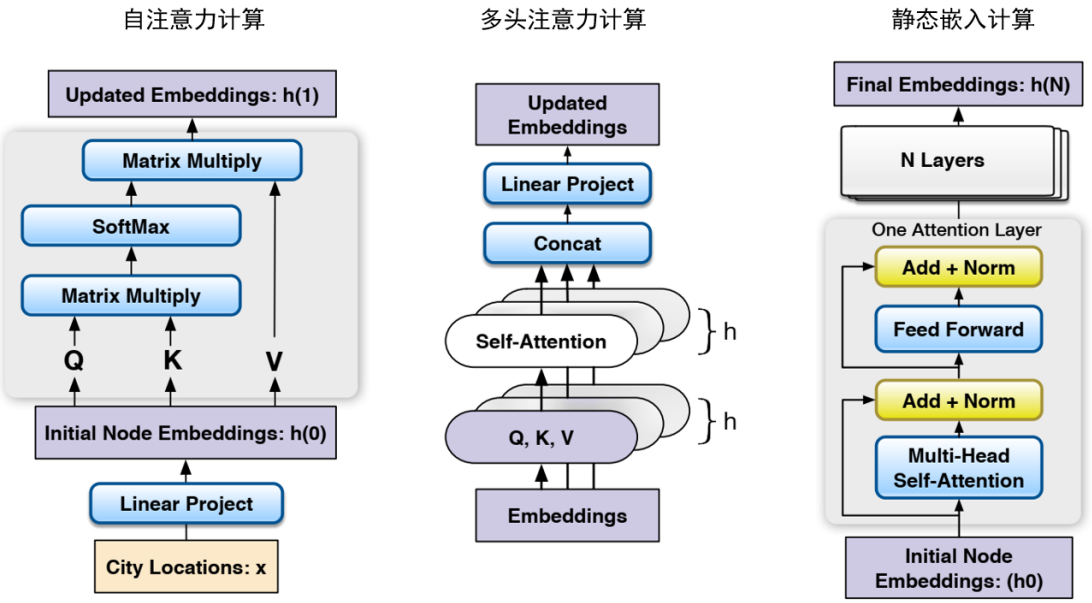


图 3.3　基于多头注意力机制的静态嵌入计算方法

Step 2. 如图 3.3 所示，节点初始特征向量 $h^{(0)}$ 通过 N 注意力层进行处理和更新。ℓ_{th} 层的节点特征向量计算如下：

$$\begin{aligned} \mathbf{h}_{tmp} &= Norm^{\ell}\left(\mathbf{h}^{(\ell-1)} + \mathrm{MHA}^{\ell}\left(\mathbf{h}^{(\ell-1)}\right)\right) \\ \mathbf{h}^{(\ell)} &= Norm^{\ell}\left(\mathbf{h}_{tmp} + \mathrm{FF}^{\ell}\left(\mathbf{h}_{tmp}\right)\right) \end{aligned} \quad . \tag{3.8}$$

N_{th} 层输出的向量 $h^{(N)}$ 是最终的节点特征向量，其中 $h_i^{(N)}$ 是城市 i 的 d_h 维静态特征向量。

3.2.2.2　动态嵌入

节点的动态嵌入用于捕捉组合优化问题的动态特征。

节点的静态嵌入 $h^{(N)}$ 可以直接作为旅行商问题[106] 的 *key* 值。然而，与旅行商问题不同，在解码步骤中，覆盖旅行商问题中城市的状态不是静态不变的，这基于如下事实：一旦某个城市被覆盖，它被选择访问的可能性就会降低，因为只需要被覆盖就可以满足问题的约束；如果某个城市被覆盖的次数增加，它被访问的可能性会进一步降低。因此在覆盖旅行商城市选择的马尔可夫决策过程中，每一步动作均会引起部分节点覆盖状态的改变，而传统旅行商问题节点状态是不变的。因此，将覆盖状态的概念引入到 *key* 的构造中以进行 CSP 问题的优化至关重要。

为了有效处理问题的动态特征，本章定义一个引导向量，每个节点 i 的引导向量 g_i 表征该节点的覆盖状态，并且 g_i 在每个解码步骤中进行动态更新。这区别于传统基于深度强化学习的组合优化模型，例如指针网络模型中，节点状态在整

个解码过程中保持不变，而本章所提方法通过设计动态嵌入来对问题的动态特征进行有效提取，在每步解码过程中对节点状态进行动态调整，从而有效提高模型的求解精度。

具体地，节点动态嵌入的计算过程如图 3.4 所示，详细介绍如下：

假设在解码步骤 t 中选择要访问的城市为 π_t，则 $C(\pi_t)$ 被定义为被 π_t 覆盖的城市集合。对以下数据进行计算：计算 $C(\pi_t)$ 中的节点的数量，计为 $N_{C(\pi_t)}$；计算 $C(\pi_t)$ 中的所有节点到 π_t 的欧几里得距离，并据此进行排序，$C(\pi_t)$ 中的每个城市都有一个排序索引 c_i，范围从 1 到 $N_{C(\pi_t)}$，其中 $c_i = 1$ 表示城市 i 最接近 π_t，反之亦然。

在解码过程开始时，所有城市的 g_i 值被初始化为 1：

$$g_i = 1, i = 1, \cdots, n. \tag{3.9}$$

在每个解码步骤 t 中，g_i 按照如下公式进行动态更新：

$$g_i = g_i \times \frac{c_i}{N_{C(\pi_t)}}, i \in C(\pi_t), i = 1, \cdots, n. \tag{3.10}$$

根据式 (3.10) ，城市 i 的动态特征由值 g_i 表示，g_i 根据节点的访问状态进行更新，参考图 3.4 ，一旦节点 A 被访问，节点 A 周边节点（即 A 所覆盖的节点）的 g_i 值则会相应减小，减小的量根据它们与节点 A 之间的距离决定，离节点 A 最近的节点的 g_i 值最小，如果某节点 i 再次被其他节点覆盖，则该节点的 g_i 会进一步降低。

因此，该引导向量包含每个节点的覆盖状态：它是否被覆盖，以及它与覆盖它节点之间的距离，引导向量可以有效体现节点的覆盖信息，从而指导下一个城市的选择，例如，如果某节点与被访问过城市之间的距离很近，那么该节点被选中的概率可能会比较低。

但是在某些情况下，距离被访问城市近的节点也可能是较好的选择，不能根据这个简单的经验观察对城市选择策略直接进行调整，相反地，应该通过参数化的模型对该启发式经验知识进行学习，从而得到更加精确的策略。因此，模型没有直接根据节点 i 的 g_i 值显式地降低城市 i 的选择概率，而通过神经网络的参数来学习如何利用 g_i 调整策略选择的概率，因此，将 g_i 通过神经网络参数 W_G 线性投影为 d_h 维的动态嵌入 G_i：

$$G_i = g_i W_G \tag{3.11}$$

由于 G_i 在解码过程中动态变化，因此 G_i 称为动态嵌入。静态嵌入 $h^{(N)}$ 表示城市的静态状态，即它们的空间位置，而动态嵌入 G 表示它们的动态状态。然后

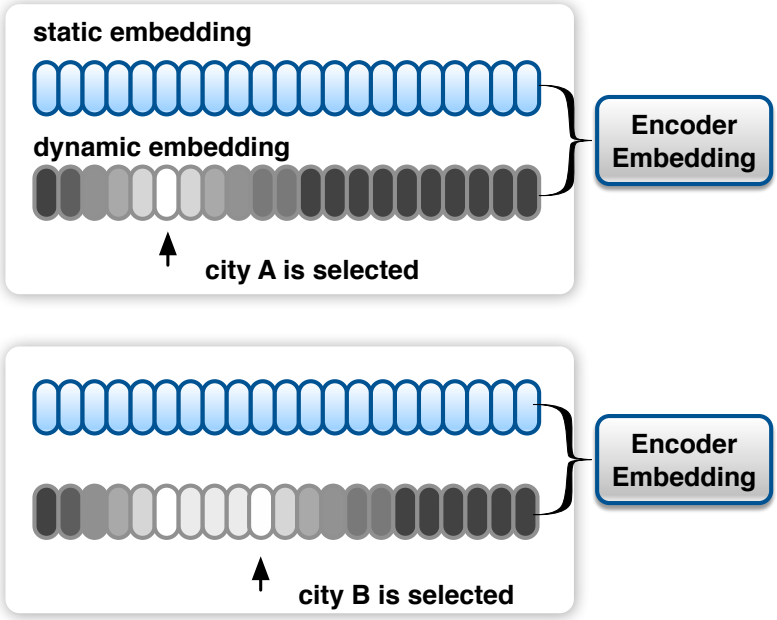


图 3.4　节点动态嵌入更新示意图

根据 $h^{(N)}$ 和 G 构造最终的 key:

$$\mathrm{k}_i = W^K h_i^{(N)} G_i, i = 1 \cdots n \tag{3.12}$$

其中 $\mathbf{k}_i$ 表示城市 i 的 key，W^K 是大小为 $(d_h \times d_h)$ 的可学习权重，因此节点状态在解码过程中动态变化，既包含其静态位置信息，又包含其动态覆盖特征。通过这种方式，模型可以捕获问题的静态和动态模式，从而学习到更好的策略。

3.2.3　解码器结构

解码器主要用于 *query* 值的构建，*query* 值表征覆盖旅行商问题当前解的状态，*query* 值用于和编码器的 *key* 值进行匹配度的计算，最匹配的 *key* 所代表的节点将被选择，因此 *query* 值需要能够准确表征当前解的状态。

传统基于深度强化学习的组合优化模型的解码器一般为循环神经网络。本章所提模型的解码器由两部分组成：循环神经网络和注意力机制。

3.2.3.1　循环神经网络

循环神经网络用于在第 t 步解码过程输出 d_t，包含了当前已访问节点的信息，要选择下一个要访问的城市，自然地，需要考虑之前访问过的城市信息，由于循环神经网络能够有效处理此类序列信息，因此采用循环神经网络对其进行建模，LSTM 是当前应用最广泛的循环神经网络模型，本章采用使用 LSTM 循环神经网络对 *query* 值进行构建，LSTM 模型以初始隐含层状态 h_0、第 t 步的输入 x_t 作为

模型输入，输出每步的隐层状态 h_t，h_t 的计算过程 f_{LSTM} 如下：

$$\begin{aligned} f_t &= \sigma_g\left(W_f x_t + U_f h_{t-1} + b_f\right) \\ i_t &= \sigma_g\left(W_i x_t + U_i h_{t-1} + b_i\right) \\ o_t &= \sigma_g\left(W_o x_t + U_o h_{t-1} + b_o\right) \\ \tilde{c}_t &= \sigma_c\left(W_c x_t + U_c h_{t-1} + b_c\right) \\ c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\ h_t &= o_t \circ \sigma_h\left(c_t\right) \end{aligned} \tag{3.13}$$

其中 W、U、b 等代表 LSTM 神经网络的参数，本章将编码器所计算得到的节点特征向量的均值 $\overline{\mathbf{h}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_i^{(N)}$ 作为解码器 LSTM 的初始隐藏状态，即 $d_0 = \overline{\mathbf{h}}$。同时，将参数 $\mathbf{v}_1$ 作为解码器 LSTM 的第一个输入，$\mathbf{v}_1$ 为可学习的神经网络权重。在步骤 $t > 1$，解码器 LSTM 的输入是上一步所选择的节点 π_{t-1} 的编码器特征向量 $\mathbf{h}_{\pi_{t-1}}^{(N)}$，因此解码器的计算过程如下：

$$o_t, d_t = f_{LSTM}\left(I_t, d_{t-1}\right), d_0 = \overline{\mathbf{h}} \tag{3.14}$$

$$I_t = \begin{cases} \mathbf{v}_1 & t = 1 \\ \mathbf{h}_{\pi_{t-1}}^{(N)} & t > 1 \end{cases} \tag{3.15}$$

d_t 是解码器 LSTM 在解码步骤 t 的隐层状态，它包含了先前选择的城市 $\pi_{1\sim t-1}$ 的信息。由于 π_t 的选择很大程度上取决于 $\pi_{1\sim t-1}$，因此解码器所输出的 d_t 自然地可以用来构建 *query*。

3.2.3.2 注意力机制

只采用 LSTM 可以实现 *query* 的构造，但是已访问城市 $\pi_{1\sim t-1}$ 并不是决定 π_t 的唯一因素。为了构建 *query*，还应该考虑已访问的城市与其他城市之间的关系。例如，在旅行商问题中，离上次访问的城市更近的城市可能有更多的机会被选择，但是在覆盖旅行商问题中，离上次访问的城市更近的城市由于被覆盖，被选择的概率也会受到影响，因此存在一个复杂的关系，不需要人为地对这种影响关系进行经验总结，而可以采用神经网络模型对该模式进行自动发现，因此为了使得表征当前问题状态的 *query* 更加精确，本章使用注意力机制进一步对 d_t 进行处理，方法如下所示：

$$\mathrm{q}_t = Attention(d_t, K_1, V_1) = \mathrm{softmax}\left(\frac{d_t K_1^T}{\sqrt{d_k}}\right) V_1, \tag{3.16}$$

其中 K_1 和 V_1 是编码器静态嵌入 $h^{(N)}$ 的线性变换。因此，用于选择 π_t 的 *query* 值 q_t 被定义为 d_t 和 K_1, V_1 键值对的注意力值。通过对 d_t 添加额外的注意力

操作，获得的 *query* 包含更丰富的信息：先前已访问城市的信息（由 d_t 表示）；以及已访问城市和其他城市（由 $h^{(N)}$ 表示）之间的关系特征。该注意力操作能够进一步提取已访问城市和其他城市之间的位置关系，从而实现更加有效的查询。

3.2.4 注意力机制计算条件概率

上述内容对 *query* 和 *key* 的构造进行了介绍，根据第 t 步的 *query* 值 q_t 和所有节点的 *key* 值，可以计算第 t 步城市选择的概率，即在所有 $key_i, i = 1 \dots n$ 中，最匹配 q_t 的将被选为下一个要访问的城市。该计算过程通过注意力机制实现。

具体地，在第 t 步解码过程中，本章采用 Scaled Dot-Product Attention 方法基于 *query-key* (q_t, k_i) 计算所有城市 $i = 1 \cdots n$ 的选择概率：

$$u_i^t = \frac{\mathbf{q}_t^T \mathbf{k}_i}{\sqrt{d_k}}, \tag{3.17}$$

其中 $\sqrt{d_k}$ 是缩放因子，$d_k = \frac{d_h}{M}$。M 是多头注意力机制中注意力计算的次数。同时采用 mask 机制来避免被访问过的城市被再次选择：

$$u_i^t = \begin{cases} -\infty & \text{if } j \in \pi_{1 \sim t-1} \\ \frac{\mathbf{q}_t^T \mathbf{k}_i}{\sqrt{d_k}} & \text{otherwise.} \end{cases} \tag{3.18}$$

采用 softmax 算子对上述概率进行正则化后，已选择城市的选择概率则为零。最终城市选择的条件概率是通过如下 softmax 运算实现：

$$p_i = \frac{e^{u_i^t}}{\sum_{j=1}^{n} e^{u_j^t}}, i = 1, \cdots, n \tag{3.19}$$

假设使用贪婪策略根据计算得到的条件概率进行组合优化决策，则在解码步骤 t，选择概率最大的节点 p_i 进行访问，如果在训练阶段，通常根据条件概率进行动作采样，即按照条件概率分布随机选择一个节点，在模型在线应用阶段则按照贪婪策略进行动作选择。

综上，本章根据覆盖旅行商问题的特性设计了一种基于动态嵌入的注意力模型，模型由编码器、解码器和注意力机制组成，编码器将组合优化问题的特征作为输入，并输出节点特征向量。解码过程顺序进行，直到所有城市都被访问或覆盖为止，选择的节点构成问题的解 $(\pi_0, \pi_1, \cdots)$。

3.3 模型训练方法

给定一个组合优化问题实例 s，上节内容定义了由参数 $\boldsymbol{\theta}$ 参数化的深度神经网络模型，该模型可以通过式 (3.18) 和式 (3.19) 产生条件概率分布 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$。解 $\boldsymbol{\pi}|s$ 可以通过从 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$ 条件概率分布中采样得到。

由于组合优化问题存在一个优化目标，例如覆盖旅行商问题的优化目标是最小化路径总长度 $L(\pi)$，因此适合使用回合制的 REINFORCE 算法 [108] 来对上述深度神经网络模型进行训练，该算法使用基于回合制的蒙特卡洛方法来计算奖励值，即一直执行该策略，直到构造得到一个完整解，此时计算总的路径长度的负数作为奖励（$-L(\pi)$），即最小化路径长度。基于该定义，策略参数可以通过 *state-action-reward* 来不断更新。

具体地，给定动作 $\boldsymbol{\pi}|s$ 和相应的奖励值 / 损失函数 $L(\pi)$，可以基于 REINFORCE 算法 [108]，通过梯度下降来更新模型的参数 $\boldsymbol{\theta}$：

$$\begin{aligned} \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) &= \mathbf{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)} \left[\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s) L(\boldsymbol{\pi}) \right] \\ \boldsymbol{\theta} &\leftarrow \boldsymbol{\theta} + \nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}). \end{aligned} \tag{3.20}$$

组合优化解的构造需要从概率分布中采样大量动作，将总的路径长度作为奖励，由于采样的不确定性以及每个回合中存在的大量采样过程，不同回合得到的奖励存在很大的方差，该方差会使策略参数的梯度下降方向产生冲突，影响收敛速度。为了减少策略参数更新的方差，通常引入基线值 $b(s)$ 来重写式 (3.20) 中的策略梯度更新公式：

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) = \mathbf{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)} \left[\nabla \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s) (L(\boldsymbol{\pi}) - b(s)) \right]. \tag{3.21}$$

$b(s)$ 可以看作是用来评价策略的平均表现，如果当前策略的表现优于该平均水平 $b(s)$，$L(\boldsymbol{\pi}) - b(s)$ 为负数，则该策略受到正向激励（最小化该损失函数），如果当前策略的表现劣于该平均水平 $b(s)$，$L(\boldsymbol{\pi}) - b(s)$ 为正数，则该策略受到负向激励。当前主流的方法是训练一个额外的神经网络来逼近 $b(s)$，即 Critic 网络。它的参数是通过训练过程中收集的 $(s, L(\boldsymbol{\pi}))$ 进行训练。然而，文献 [106] 指出，同时训练由神经网络参数化策略和 Critic 神经网络网络会产生收敛慢的问题，因此本章采用文献[106] 中提出的 greedy rollout baseline 方法来计算 $b(s)$，它在实验中被证明优于基于 Critic 神经网络的训练方法。

具体地，不需要设计额外的神经网络来计算 $b(s)$，本章中，给定一个组合优化问题实例 s，$b(s)$ 定义为利用基准策略 $p_{\boldsymbol{\theta}^*}$ 对该问题实例 s 进行求解得到的组合优化解的路径长度，求解过程中对概率分布的采样方式为贪婪策略。基准策略 $p_{\boldsymbol{\theta}^*}$ 是在训练过程中迄今为止最好的模型，即在每个训练周期结束时在验证集上进行策略评价，一旦当前策略优于当前的基准策略，则用当前模型替换基准策略，从而使得基准策略为训练至今最优的模型，$b(s)$ 即当前最优的策略表现，通过该基准表现对策略梯度更新进行优化。

算法3.1概述了该改进的 REINFORCE 训练算法，算法采用 Adam 优化器 [109]。

通过这种方式，如果采样的解决方案 $\boldsymbol{\pi}$ 比当前最优模型的贪婪表现更好，那么该策略被加强，反之亦然。因此该方法能够使得策略参数稳定地向最小化路径长度的方向进行更新。

算法 3.1 基于 REINFORCE 的强化学习训练方法

算法输入： 训练集 $\mathcal{X}$, 一次训练的样本数（batch size）B, 训练轮数（number of epochs）, N_{epoch}, 每轮训练步数 N_{steps}

初始化当前策略的神经网络参数和基准策略的神经网络参数：$\boldsymbol{\theta}, \boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
for $epoch \leftarrow 1 : N_{epoch}$ **do**
    for $step \leftarrow 1 : N_{step}$ **do**
        从训练集 $\mathcal{X}$ 中随机产生 B 个批训练个体 $s_i, i \in 1, \ldots, B$。
        $\boldsymbol{\pi}_i \leftarrow p_{\boldsymbol{\theta}}(s_i)$. 以随机采样的方式执行该策略，即对神经网络参数进行前向传播计算得到条件概率分布，以随机采样的方式选择动作。
        $\boldsymbol{\pi}_i^* \leftarrow p_{\boldsymbol{\theta}^*}(s_i)$. 以贪婪采样的方式执行基准策略。
        计算当前策略和基准策略产生组合优化解的路径长度 $L(\boldsymbol{\pi}_i) - L(\boldsymbol{\pi}_i^*)$。
        $\nabla \mathcal{L} \leftarrow \sum_{i=1}^{B} \left(L(\boldsymbol{\pi}_i) - L(\boldsymbol{\pi}_i^*)\right) \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}_i)$
        计算 $\nabla \mathcal{L}$，根据 Adam 优化算法对参数 $\boldsymbol{\theta}$ 进行更新。
    end for
    在验证集上对当前策略 $p_{\boldsymbol{\theta}}$ 和基准策略 $p_{\boldsymbol{\theta}^*}$ 进行验证。
    if $p_{\boldsymbol{\theta}}$ 表现优于 $p_{\boldsymbol{\theta}^*}$ **then**
        $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
    end if
end for

3.4 实验结果与讨论

为了验证本章所提基于深度强化学习的复杂单目标组合优化方法在求解覆盖旅行商问题上的有效性，本节在 20、50、100、200 和 300 规模的覆盖旅行商问题上进行实验验证，首先将该方法与当前业界最优的基于深度强化学习的组合优化方法进行实验比较，并进一步与当前在覆盖旅行商问题上使用最广泛的启发式基准方法进行比较，从而对所提方法的有效性进行全面的验证。

3.4.1 实验设置

3.4.1.1 实验对比算法

选取以下三个基于深度强化学习的组合优化方法与本章所提方法（Attention-dynamic）进行实验对比：

- 指针网络模型 [92]（PN）。
- 具有动态嵌入的指针网络模型 [110]（PN-dynamic）。
- 注意力模型 [106]（Attention）。

指针网络模型 [92] 是第一个可以有效解决组合优化问题的深度学习模型，该方法在当前基于深度强化学习的组合优化研究中作为常用作基准方法。具有动态嵌入的指针网络模型[110] 旨在解决在决策过程中问题状态发生改变的 VRP 问题，有与 CSP 相似的动态特征，因此选取该模型作为第二个对比模型。注意力模型 [106] 是当前最先进的基于深度强化学习的组合优化方法，它在解决各种组合优化问题上取得了当前最优的效果。因此选择上述算法进行实验对比。

在深度学习技术出现之前，启发式方法是解决 CSP 的主要方法。因此，除了深度学习方法，还采用启发式方法进行实验对比。LS1 和 LS2 [96] 是当前覆盖旅行商问题研究中常用作比较基准的方法 [96–98, 100]。由于所有深度学习模型都在 Python 平台上运行，并且网络上没有公开的 LS1/LS2 源代码，因此基于 Python 语言对 LS1/LS2 进行了实现，从而保证实验对比的公平性。代码严格按照文献[96] 编写，并且进行了开源[1]。

3.4.1.2 模型超参数设置

表 3.1 列出了模型训练的超参数，针对本章所提模型和所有基于深度学习的对比算法，均使用如下超参数：批量大小为 256、节点向量维度为 128、编码器 - 解码器的隐层向量维度为 128。均使用 Adam 优化器 [109] 以 10^{-4} 的固定学习率进行模型训练。

在训练阶段，随机生成 320,000 个覆盖旅行商问题作为训练集，模型训练执行 50 个轮次，覆盖旅行商问题按照如下方式随机生成：城市位置从 [0,1] 的均匀分布中随机采样，每个城市可以覆盖的邻近城市的数量设置为 7（number of covering, NC）。NC 在模型训练时设置为固定数量 7，测试时可以设置为任意数量，这也进一步验证了模型的泛化能力，泛化能力的实验验证在第3.4.4章中进行了讨论。论文使用相同的参数和数据集来训练所有神经网络模型，并采用原论文 [96] 中推荐的超参数运行 LS1 和 LS2 方法。

3.4.1.3 模型实现方法

所有模型均采用 Python 实现，实验均在 Python 中进行，实验平台的参数为：GTX 2080Ti GPU 和 Intel 64GB 16 Cores i7-9800X 的 CPU，并且模型代码开源 [2] 以复现实验结果并促进未来的研究。

[1]https://github.com/kevin031060/CSP_Attention/tree/master/LS

[2]https://github.com/kevin031060/CSP_Attention

表 3.1　实验参数设置

参数	值	参数	值
批量大小	256	隐层维度	128
训练轮次	50	多头注意力特征个数	8
每轮次的训练集规模	320000	编码器层数	3
优化器	Adam	学习率	1e-4

3.4.1.4　模型评估方法

本章采用随机生成的 100 个覆盖旅行商问题对模型的性能进行评估，性能指标为 100 个解的平均路径长度。由于对比算法 LS1 和 LS2 需要较长的执行时间，因此只生成 100 个实例进行测试，文献[105] 证明，针对基于深度学习的组合优化方法，在得到解之后再通过一个简单的局部搜索可以有效地提高神经组合模型获得的解的质量，因此本节采用简单的局部搜索对模型解进行进一步改进，从而在不严重增加求解耗时的情况下提高解的质量，因此算法运行时间是神经网络模型的推理时间和局部搜索的执行时间的总和。

通过上述的实验设置进行实验，以验证所提出方法的学习能力、优化能力和泛化能力。

3.4.2　模型学习能力验证

首先验证本章所提方法（Attention-dynamic）在模型训练阶段的学习能力。

图 3.5 比较了本章所提方法、PN、PN-dynamic 和 Attention 模型在训练过程的模型收敛性能，采用独立于训练集的验证集的平均路径长度作为在训练阶段模型评估的指标。图 3.6 进一步展示了训练后期各个模型的指标值。

可见 Attention-dynamic 在收敛速度和收敛能力方面明显优于当前的基于深度强化学习的组合优化方法。PN 和 Attention 模型收敛能力明显弱于本章所提方法，因为它们没有考虑城市覆盖状态的动态信息；虽然 PN-dynamic 模型考虑了问题的动态特征，但该方法收敛速度明显慢于 Attention-dynamic，因为本章使用的多头注意力机制可以有效地提取覆盖旅行商问题的特征，从而有助于模型更快的收敛，并且取得比 PN-dynamic 模型更强的收敛能力。通过对比可以发现本章所提方法在训练过程上展现出更加稳定、样本效率更高的优势，具有很强的收敛速度和收敛能力。

表 3.2 概述了所有模型的训练时间，可见基于指针网络的模型不仅性能较差，而且需要较长的训练时间。本章所提方法可以节省高达 75% 的训练时间，同时显示了比 PN-dynamic 模型更强的收敛性能。使用单个 RTX2080Ti GPU 在 CSP50、

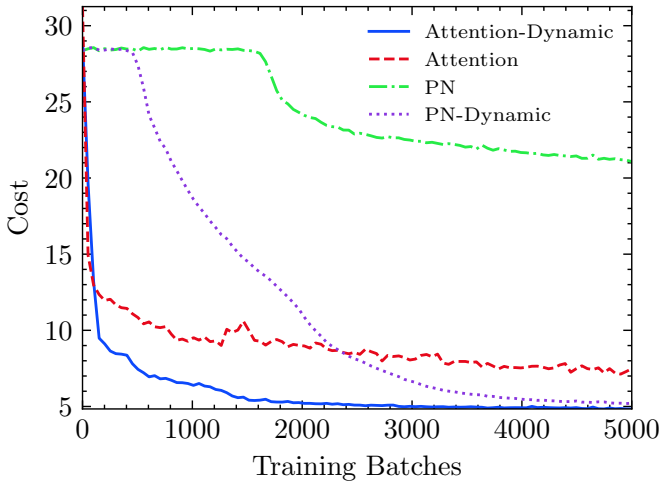


图 3.5　验证集对比模型训练表现

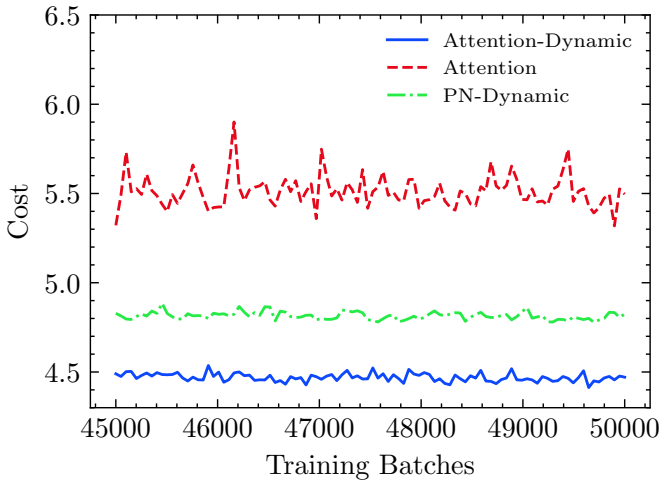


图 3.6　验证集对比模型训练表现（最终阶段）

表 3.2　训练 100 个 mini-batch 训练时间对比

Method	CSP50	CSP100
PN	85.6	176.2
PN-dynamic	107.5	237.8
AM	26.3	45.9
AM-dynamic	32.6	52.6

CSP100 和 CSP200 实例上训练本章所提方法分别需要 3.6 小时、5.8 小时和 11.3 小时。

3.4.3 模型求解能力验证

为了评估模型的优化性能，本节对所提模型（Attention-dynamic）在小规模的CSP任务（CSP20、CSP50、CSP100）和大规模的任务（CSP150、CSP200、CSP300）的模型表现进行评估，采用如下实验方法：在CSP50实例上训练的模型用于求解随机生成的CSP20和CSP50测试问题；在CSP100实例上训练的模型用于求解随机生成的CSP100和CSP150测试问题；在CSP200实例上训练的模型用于求解随机生成的CSP200和CSP300测试问题。通过实验发现，如果使用在CSP20上训练的模型来求解CSP300问题时，基于深度学习的模型性能均会显著下降，因此本节采用上述的实验设置进行模型性能评估。

采用如下指标对模型的优化性能进行评估：1）Cost：模型输出解所计算得到的平均路径长度；2）Gap：该模型的Cost与最佳求解器Cost之间的最优性差距；3）Time：模型求解时间。

表3.3和表3.4分别列出了本章所提方法和对比算法在小规模CSP任务（CSP20、CSP50、CSP100）和大规模CSP任务（CSP150、CSP200、CSP300）上的性能，表的第一部分列出了启发式对比方法的实验结果，第二部分列出了深度学习模型的实验结果，实验结果分析如下。

3.4.3.1 与传统深度学习方法实验对比结果

由表3.3和表3.4可见，本章所提方法在所有覆盖旅行商测试问题上均优于传统的基于深度强化学习的组合优化方法PN、PN-dynamic和Attention，并且优化能力的提升明显。传统深度学习方法与与启发式求解器的优化能力存在很大差距，Attention-dynamic极大地缩小了该优化能力的差距，虽然与启发式求解器之间仍然存在优化能力的差距，但是求解速度获得了极大地提升。

3.4.3.2 与传统启发式方法实验对比结果

进一步地，对本章所提方法结合局部搜索（AM-dynamic(LS)）与传统启发式搜索方法进行对比，本章所提方法的求解时间为深度神经网络模型的推理时间和局部搜索时间的总和。实验对比结果如表3.3和表3.4的第三部分所示。可见，进行简单的局部搜索，可以在不急剧增加求解时间的情况下提升组合优化解的质量，该结果与[105]中实验结果一致。

表3.3和表3.4的实验结果表明，本章所提方法可以实现接近LS1等传统启发式求解器的优化性能（除CSP20之外，本章方法在所有CSP测试问题上与传统启发式算法的最优性差距不超过2%）。尽管仍然存在优化能力的差距，但本章所提方法求解速度比启发式方法快得多：在使用额外局部搜索的情况下，它比传统启发式算法LS1&LS2快20到40倍，在不使用局部搜索的情况下，比LS1&LS2快

表 3.3　本章所提方法在小规模 CSP 问题上与传统方法的实验对比结果

Method	CSP20			CSP50			CSP100		
	Cost	Gap	Time/s	Cost	Gap	Time/s	Cost	Gap	Time/s
LS1	1.87	8.09%	4.24	2.57	0.00%	15.04	3.58	0.00%	88.85
LS2	1.73	0.00%	6.55	2.67	3.89%	21.61	3.69	3.07%	107.98
PN	4.27	146.82%	0.019	7.85	205.45%	0.027	12.05	236.59%	0.042
PN-dynamic	2.15	24.28%	**0.011**	3.29	28.02%	**0.015**	4.68	30.73%	**0.033**
AM	2.23	28.90%	0.014	3.87	50.58%	0.041	5.02	40.22%	0.064
AM-dynamic	**1.89**	**9.25%**	0.014	**2.75**	**7.00%**	0.032	**4.08**	**13.97%**	0.063
AM-dyna (LS)	**1.85**	**6.94%**	**0.5**	**2.64**	**2.72%**	**0.71**	**3.65**	**1.96%**	**2.66**

表 3.4　本章所提方法在大规模 CSP 问题上与传统方法的实验对比结果

Method	CSP150			CSP200			CSP300		
	Cost	Gap	Time/s	Cost	Gap	Time/s	Cost	Gap	Time/s
LS1	4.28	0.00%	331.45	4.86	0.00%	755.77	5.93	0.00%	2637.55
LS2	4.49	4.91%	362.28	5.16	6.17%	695.77	6.34	6.91%	2428.75
PN	16.25	279.67%	0.087	22.54	363.79%	0.117	29.68	400.51%	0.228
PN-dynamic	6.47	51.17%	**0.042**	8.44	73.66%	**0.076**	11.17	88.36%	0.128
AM	6.11	42.76%	0.102	8.28	70.37%	0.162	10.52	77.40%	0.291
AM-dynamic	**5.19**	**21.26%**	0.071	**6.01**	**23.66%**	0.082	**7.62**	**28.50%**	**0.106**
AM-dyna (LS)	**4.34**	**1.40%**	**7.99**	**4.93**	**1.44%**	**4.93**	**5.98**	**0.84%**	**85.08**

数千倍。

值得注意的是，本章所提方法是在与测试问题不同的数据集上进行训练的，上述结果是在没有重新训练的情况下直接将模型应用于具有不同城市位置、维度的测试问题上，因此本章所提方法和传统启发式求解器之间存在轻微性能差距是合理的。此外，实验结果表明了本章所提方法在快速求解能力和泛化能力上的优势。

另外，启发式求解器通常需要一个停止搜索的标准，例如设置进行固定次数的搜索或者当目标值持续一定时间不发生改变则停止搜索，在这种情况下会造成额外的求解时间，因此需要进行更加公平的实验对比。同时，求解时间和优化能力之间的权衡很难进行量化，虽然极大缩短了求解时间，但是同时也造成了优化能力的下降。

因此，为了更加公平、直观、科学地对本章所提方法的有效性进行验证，本节进一步设计了控制实验，将所有对比算法执行到相同的优化水平，对其相同优化能力下的求解时间进行比较。具体实验设置如下：将本章所提方法 AM-dynamic(LS)获得的目标值设置为基准 Cost，启发式方法在优化过程中的目标值一旦达到或超过该基准 Cost 就立即停止，此时记录它们的算法求解时间，该求解时间 StopTime 即为达到相同优化能力所需要的求解时间，实验结果在表 3.5 中列出。

从实验结果可以明显看出，如果所有模型都达到相同的优化水平，本章所提方法比传统启发式求解器快 10 倍以上，证明了本章所提方法比传统方法具有明显的优势。

表 3.5　对比算法在达到相同优化水平所需求解时间的对比

	LS1		LS2		AM-dynamic (LS)	
	Cost	StopTime	Cost	StopTime	Cost	StopTime
CSP50	2.619	2.35	2.642	11.14	2.641	**0.78**
CSP100	3.632	27.69	3.742	71.43	3.656	**2.71**
CSP150	4.332	128.64	4.476	290.66	4.337	**8.07**
CSP200	4.93	327.59	5.086	559.48	4.932	**22.14**
CSP300	5.98	1295.65	6.26	2697.78	5.972	**85.25**

3.4.3.3　实际问题数据集实验对比结果

上述为本章所提方法在随机生成的虚拟测试问题上的表现，本节对 Attention-dynamic 方法在实际数据集上的有效性进行测试。

文献中通常采用 TSPLIB 构建覆盖旅行商问题[96–98]，具体构建方法为，将 CSP 测试问题的城市位置设置为 TSPLIB 数据集 [3] [111] 中给定的城市位置，设置每个节点能够覆盖的周围节点的数量（NC）为 7。

为了进行公平的实验对比，仍然采用上节所述的控制实验，即通过执行对比算法至相同的优化水平，比较各个算法的执行时间。表 3.6 展示了该实验结果。可以看出，所提出的方法在大多数 TSPLIB 实例上都超过了传统启发式算法 LS1，在小规模实例上，启发式方法的效果会相对好一些，但是在大规模实例上需要非常长的求解时间。

值得注意的是，该实验所用模型是在从均匀分布生成的问题实例上训练的，而 TSPLIB 实例不是均匀分布的，这种数据分布的差异是本章所提模型在 TSPLIB 实例上的性能比在随机生成的实例上的性能稍差的原因，但是仍然可以看出，本

[3]http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/

章所提方法具有明显的求解速度和优化能力的优势。

表 3.6　本章所提方法在 TSPLIB 数据集上与传统方法的实验对比结果

	NC=7			NC=11		
Methods		LS1	AM-dynamic		LS1	AM-dynamic
Instances	Cost	Time/s	Time/s	Cost	Time/s	Time/s
ulysses22	2.253	**0.24**	0.63	1.916	2.71	**0.41**
berlin52	2.812	**0.19**	0.44	2.678	**0.19**	0.39
st70	2.925	3.13	**1.05**	3.22	**0.3**	0.45
pr76	3.488	5.27	**1.39**	3.158	**0.3**	0.46
eil76	3.142	30.63	**1.18**	3.051	**0.29**	0.49
rat99	3.722	1.16	2.23	3.23	2.09	**1.52**
rd100	3.622	13.35	**1.61**	2.991	42.01	**1.39**
kroA100	3.738	2.34	**1.63**	2.974	16.25	**1.53**
kroB100	3.704	2.42	**2.38**	2.976	29.29	**1.76**
kroC100	3.791	**1.82**	1.54	2.984	47.28	**1.58**
kroD100	3.619	5.9	1.41	2.972	13.28	**1.24**
kroE100	3.951	**3.21**	3.36	2.911	2.93	**1.5**
eil101	3.726	7.77	**2.8**	2.996	21.05	**1.82**
lin105	3.854	23.66	**1.2**	3.495	1.11	1.31
pr124	3.804	5.89	**2.26**	3.386	23.25	**1.72**
ch130	4.551	7.16	**3.52**	3.681	24.69	**2.06**
pr136	4.048	56.11	**3.76**	4.007	**1.03**	1.73
pr144	4.902	11	**2.11**	4.519	2.94	**2.08**
kroA150	4.343	9.08	**4.34**	3.572	39.86	**3.85**
kroB150	4.318	30.89	**4.53**	3.821	**1.64**	2.55
ch150	4.124	64.6	**5.07**	3.587	8.41	**4.82**
pr152	4.771	**2.61**	4.1	4.051	26.18	**1.81**
u159	4.522	216.34	**7.1**	3.729	99.22	**4.22**
rat195	5.146	25.97	**11.22**	4.159	355.49	**8.16**
d198	3.823	91.84	**73.36**	3.474	118.69	**73.97**
kroA200	5.09	230.92	**9.12**	4.346	**5.06**	5.33
kroB200	4.845	430.32	**11.03**	3.981	43.11	**6.22**
gr202	4.199	152.49	**39.5**	3.477	94.98	**19.71**
ts225	6.088	533.87	**15.47**	5.061	**4.14**	5.78
tsp225	5.184	23.79	**15.87**	4.288	19.79	**11.87**

pr226	4.596	19.93	**7.86**	4.102	10.55	**5.02**
pr264	4.634	200.33	**158.78**	4.234	237.57	**135.15**
a280	5.763	38.39	**30.6**	4.517	450.32	**20.09**
pr299	6.352	29.9	**15.83**	5.076	830.77	**17.23**
linhp318	6.143	525.8	**29.78**	5.125	874.1	**23.12**
lin318	6.143	514.02	**29.03**	5.125	873.01	**23.35**
rd400	7.08	1730.51	**242.52**	5.613	1182.48	**65.34**
pcb442	7.241	1927.87	**622.64**	5.771	1023.9	**588.03**

3.4.4 模型泛化能力验证

本节对模型泛化能力进行验证，探索模型一旦训练完成，能否对不同类型的覆盖旅行商问题进行求解。具体地，首先将模型在 NC=7 的 CSP 训练集上进行训练（每个城市覆盖其七个最近的邻居城市），然后将该模型用于求解以下不同类型的覆盖旅行商问题：

- 首先利用在 NC=7 的问题上训练得到的模型求解具有不同 NC 的 CSP 任务。表 3.7 和表 3.8 分别给出了不同规模的 NC=7、11 和 15 的 CSP 测试问题的实验结果，为每个问题随机产生 100 个测试实例用于测试，实验结果取平均值。
- 在上述 CSP 任务中，每个城市都覆盖了相同数量的周围城市。进一步地，对每个城市可以覆盖不同数量城市的 CSP 任务进行测试。为此，随机生成每个城市的 NC 值，在 NC=7 训练集上训练得到的模型用于求解该问题。表 3.9 和表 3.10 给出了该测试问题在不同规模上的实验结果，每个问题均随机产生 100 个测试实例用于测试。
- 仍然使用该模型，对另一类 CSP 问题进行求解：每个城市都有一个固定的覆盖半径，半径内的所有城市都被覆盖，而不是覆盖指定数量的城市。并且对固定覆盖半径和可变覆盖半径的 CSP 问题均进行测试。固定覆盖半径的 CSP 问题的半径设置为固定值 0.2，可变覆盖半径的 CSP 问题的半径从 [0,0.25] 中随机产生。表 3.9 和表 3.10 给出了该测试问题在不同规模上的实验结果，每个问题均随机产生 100 个测试实例用于测试。

表 3.7 和表 3.8 的实验结果表明，当模型泛化到具有不同 NC 值的 CSP 任务时，本章所提方法的性能略有下降，对于 NC=11 和 NC=15 的测试问题，优化精度差距分别超过了 3% 和 2%。但是与传统启发式方法相比，本章所提方法仍然能够得到优化精度差距很小的近似最优解，同时所需的求解时间要少得多。

可以发现，在 NC=15 的 CSP100 测试问题上，本章所提方法存在一个较大的优化精度差距，这与 NC=7 的 CSP20 测试问题的实验结果相似（与其他测试实例相比，模型在该测试问题下存在较大的 6.94% 的优化精度差距），可见在小规模问题上传统启发式方法具有更好的表现，而在大规模问题上本章所提方法具有更强的性能。

表 3.7　模型在不同固定 NC 值的小规模 CSP 问题上的泛化表现

NC	Method	CSP100			CSP150		
		Cost	Gap	Time/s	Cost	Gap	Time/s
7	LS1	3.58	0.00%	88.85	4.28	0.00%	331.45
	LS2	3.69	3.07%	107.98	4.49	4.91%	362.28
	AM-dynamic (LS)	3.65	1.96%	2.66	4.34	1.40%	7.99
11	LS1	2.94	0.00%	69.6	3.51	0.00%	230.38
	LS2	3.03	3.06%	55.92	3.69	5.13%	144.77
	AM-dynamic (LS)	3	2.04%	2.69	3.61	2.85%	5.92
15	LS1	2.58	0.00%	57.88	3.09	0.00%	170.93
	LS2	2.64	2.33%	35.69	3.21	3.88%	96.37
	AM-dynamic (LS)	3.02	17.05%	1.12	3.17	2.59%	4.22

表 3.8　模型在不同固定 NC 值的大规模 CSP 问题上的泛化表现

NC	Method	CSP200			CSP300		
		Cost	Gap	Time/s	Cost	Gap	Time/s
7	LS1	4.86	0.00%	755.77	5.93	0.00%	2637.55
	LS2	5.16	6.17%	695.77	6.34	6.91%	2428.75
	AM-dynamic (LS)	4.93	1.44%	4.93	5.98	0.84%	85.08
11	LS1	4.01	0.00%	557.82	4.93	0.00%	1015.65
	LS2	4.22	5.24%	343.93	5.22	5.88%	810.56
	AM-dynamic (LS)	4.15	3.49%	8.29	5.02	1.83%	28.45
15	LS1	3.53	0.00%	214.22	4.25	0.00%	1379.24
	LS2	3.65	3.40%	425.66	4.44	4.47%	720.74
	AM-dynamic (LS)	3.6	1.98%	12.32	4.33	1.88%	35.26

表 3.9 和表 3.10 的实验结果表明，本章所提方法可以成功地泛化到到不同类型的覆盖旅行商问题上。

表 3.9　模型在其他不同类型小规模 CSP 问题上的泛化表现

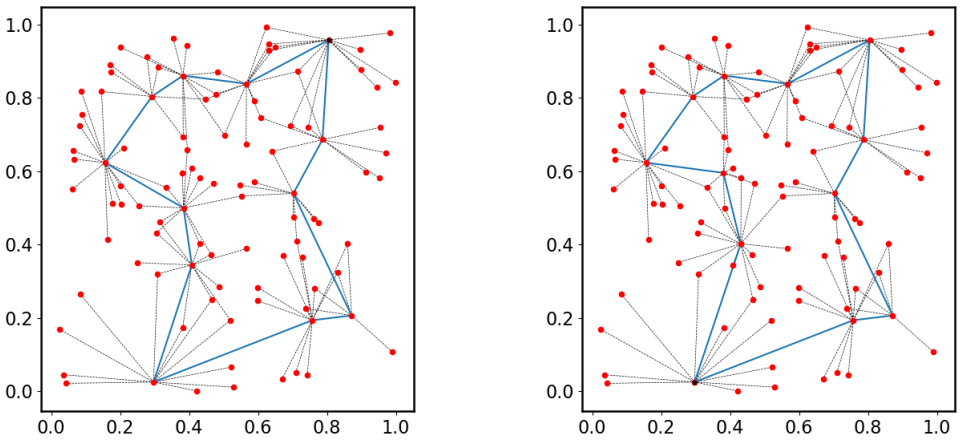
CSP tasks	Method	CSP50			CSP100		
		Cost	Gap	Time/s	Cost	Gap	Time/s
可变 NC	LS1	2.34	0.00%	5.62	2.96	0.00%	18.76
	LS2	2.36	0.85%	5.33	3.02	2.03%	22.53
	AM-dynamic (LS)	2.54	8.55%	0.36	3.01	1.69%	1.56
固定半径	LS1	2.94	0.00%	17.63	2.96	0.00%	49.1
	LS2	3.13	6.46%	21.4	3.04	2.70%	57.58
	AM-dynamic (LS)	2.96	0.68%	1.52	2.99	1.01%	1.48
可变半径	LS1	3.71	7.85%	17.84	3.41	0.59%	53.09
	LS2	3.76	9.30%	20.53	3.62	6.78%	47.24
	AM-dynamic (LS)	3.44	0.00%	1.43	3.39	0.00%	9.56

表 3.10　模型在其他不同类型大规模 CSP 问题上的泛化表现

CSP tasks	Method	CSP150			CSP200		
		Cost	Gap	Time/s	Cost	Gap	Time/s
可变 NC	LS1	3.65	0.00%	93.25	4.14	0.00%	175.68
	LS2	3.77	3.29%	67.65	4.27	3.14%	128.65
	AM-dynamic (LS)	3.7	1.37%	2.82	4.2	1.45%	6.52
固定半径	LS1	2.93	0.00%	119.34	2.93	0.00%	237.54
	LS2	3.01	2.73%	87.78	2.96	1.02%	124.57
	AM-dynamic (LS)	2.99	2.05%	2.42	2.97	1.37%	4.55
可变半径	LS1	3.29	0.00%	81.28	3.17	0.63%	115.38
	LS2	3.56	8.21%	63.33	3.46	9.84%	85.21
	AM-dynamic (LS)	3.31	0.61%	6.35	3.15	0.00%	3.45

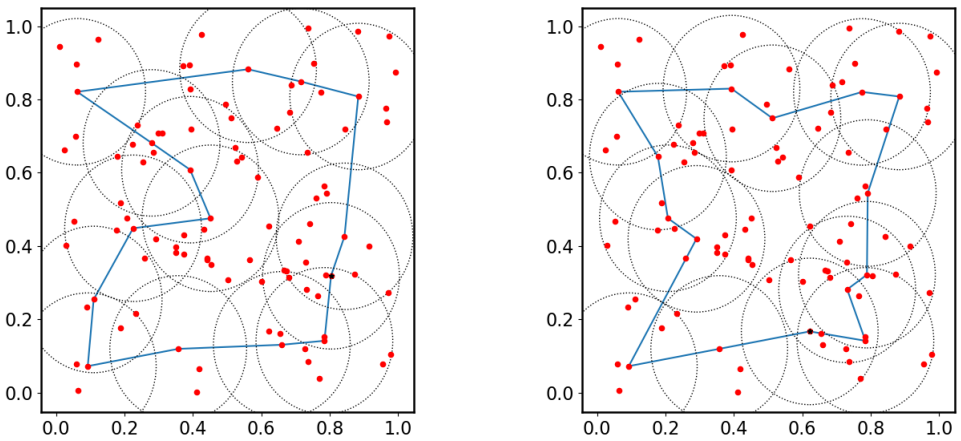
首先，对于不同城市可以覆盖不同城市数量的 CSP 任务，本章所提模型仍然表现出了良好的性能。对于除 CSP50 之外的所有 CSP 测试实例，本章方法的最优性差距始终在 2% 以内。本章所提方法在 CSP50 问题上出现的性能差的问题与上部分的实验发现一致，即在小规模问题上传统方法表现更好。

图 3.7 对该类型 CSP 问题进行了可视化，分别展示了本章所提方法以及传统 LS1 方法求解每个城市可以覆盖随机数量城市的 CSP 问题的结果，该实验中，每个城市的 NC 值随机生成，可以看出，两种方法得到的路径只有两个城市的顺序不同，这导致了本章提出方法的性能比 LS1 略差。但是从实验结果可以看出，本



(a) Attention-dynamic-LS (3.03)　　(b) LS1 (2.96)

图 3.7　覆盖城市数量可变的覆盖旅行商问题的解路径对比



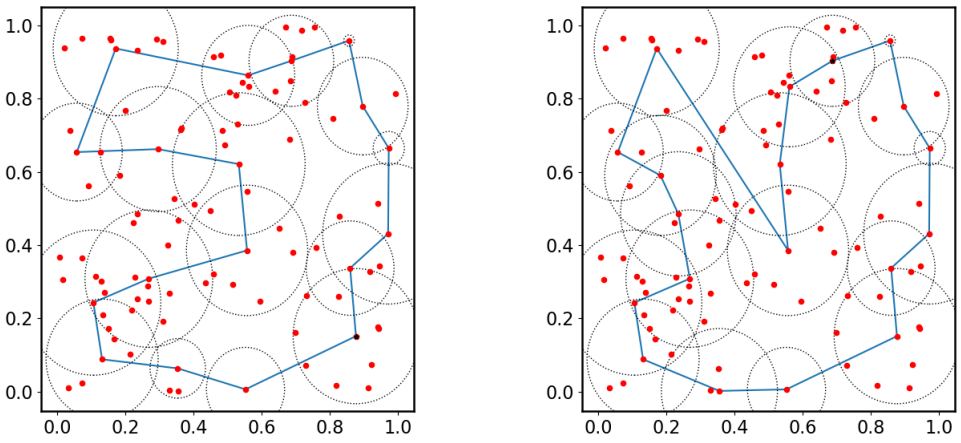
(a) Attention-dynamic-LS (3.28)　　(b) LS1 (3.16)

图 3.8　固定覆盖半径的覆盖旅行商问题的解路径对比

章所提方法的求解时间远快于传统启发式方法，只需要牺牲较小的优化精度。

其次，从表 3.9 和表 3.10 的实验结果可以看出，模型可以成功地泛化到覆盖半径可变的 CSP 问题以及固定覆盖半径的 CSP 问题，尽管该模型没有针对这些类型的 CSP 任务进行训练，但它仍然可以成功地对这些问题进行求解，在大部分问题上均可以得到近似最优解，求解速度远快于传统方法。图 3.8 和图 3.9 分别可视化了固定覆盖半径和可变覆盖半径 CSP 问题的测试结果，可以看出，本章所提方法和传统启发式方法得到的解具有很大相似性，但是本章所提方法的求解速度具有明显优势。

可见，本章所提方法能够得到覆盖旅行商问题的近似最优解，求解速度远快



(a) Attention-dynamic-LS (3.26)　　(b) LS1 (3.41)

图 3.9　可变覆盖半径的覆盖旅行商问题的解路径对比

于传统启发式方法，并且仅在 NC=7 的 CSP 问题上训练的模型可用于求解各种类型的 CSP 任务，具有很强的泛化能力和在线优化优势。这种泛化能力主要是通过论文所设计的动态嵌入来实现的，多种类型的覆盖旅行商问题均可以抽象为动态嵌入的动态特征：一旦一个城市被覆盖，它被选中的概率就会由神经网络模型参数动态调整，从而能够解决多种覆盖旅行商问题。

通过以上实验，可以看出本章所提方法具有以下优势：

- **优化能力**：针对具有动态复杂特性的大规模覆盖旅行商问题，本章所提出的方法明显优于近年来业界最优的基于深度强化学习的组合优化方法。
- **求解速度**：本章所提方法相对于传统启发式求解器，在牺牲较少优化性能的情况下，求解速度可以提升 20 倍以上；在达到相同优化水平的情况下，求解速度快 10 倍以上。实现了算法求解时间和解的最优性之间的权衡。
- **学习能力**：传统的求解器通常依赖大量专家经验进行启发式规则设计，无法学习问题实例之间的统计相似性。所提出的方法可以对相似问题之间的特征进行学习，从而具有很强的泛化能力，模型一旦训练完成，可以用来求解任意生成的具有不同规模、不同城市空间坐标的新问题，只要新问题与训练集属于相同的类别。
- **泛化能力**：所提出的方法还可以泛化到不同类型的覆盖旅行商问题，一旦模型训练好，可以用来求解以下不同类型的覆盖旅行商问题：每个城市覆盖相同个数的城市、每个城市覆盖任意随机个数的城市、相同半径内的城市均被覆盖、任意随机半径内的城市均被覆盖等，而无需重新训练模型。

3.5 本章小结

当前基于深度强化学习的组合优化方法主要针对小规模、简单的组合优化问题进行研究，本章对基于深度强化学习的复杂单目标组合优化方法进行研究，针对具有复杂动态特性的覆盖旅行商问题进行研究，提出了一种基于深度强化学习的覆盖旅行商组合优化方法，设计了一种基于动态嵌入的注意力模型，采用多头注意力机制处理问题的静态特征，采用动态嵌入处理问题的动态特征，并采用改进的 REINFORCE 强化学习算法以无监督的方式进行离线训练。通过大量实验对比可以发现，本章所提方法求解速度快、泛化能力强。与传统求解器相比，达到相同优化精度的情况下，求解速度快 10 倍以上，并且显著优于当前最先进的组合优化深度学习方法，同时，模型一旦训练完毕，可以求解任意其他类型的覆盖旅行商问题，无需重新训练，具有很强的应用潜力，适用于在线路径优化等实时决策应用。

第四章　基于深度强化学习的多目标旅行商组合优化方法

当前采用深度学习技术求解组合优化问题的方法均针对“单目标组合优化问题”，很少有研究采用深度学习方法对传统“多目标组合优化问题”进行求解，而多目标问题广泛存在与实际生产、交通、通信、军事等实际领域中，本章率先对基于深度强化学习的多目标组合优化方法开展了研究，提出了一种基于深度强化学习的多目标旅行商组合优化方法，填补了该领域的空白。

4.1　多目标旅行商问题

多目标组合优化问题（Multi-objective Combinatorial Optimization Problem, MCOP）广泛存在于实际应用当中，多目标组合优化问题即需要同时优化两个或多个目标的组合优化问题。不失一般性，该问题定义如下:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_M(\mathbf{x})) \\ \text{s.t.} \quad & \mathbf{x} \in D, \\ & G(x) \geq 0 \end{aligned} \tag{4.1}$$

其中 $\mathbf{f}(\mathbf{x})$ 由 M 个不同的目标函数组成，D 代表整数域的决策空间。由于 M 个目标通常相互冲突，因此多目标优化通常需要找到一组在各个目标上互相权衡的解，即帕累托最优解。

多目标组合优化问题的一个典型的例子是多目标旅行商问题（MOTSP），给定 n 个城市和 M 个从城市 i 到 j 的目标函数，需要找到最小化 M 个目标函数的一组帕累托最优解，多目标旅行商问题的定义如下，寻找一条长度为 n 的路径，该路径经过 n 个城市各一次，以同时最小化 M 个不同的目标函数:

$$\min z_k(\rho) = \sum_{i=1}^{n-1} c^k_{\rho(i),\rho(i+1)} + c^k_{\rho(n),\rho(1)}, k = 1, \ldots, M \tag{4.2}$$

其中 $c^k_{\rho(i),\rho(i+1)}$ 是从城市 $\rho(i)$ 到城市 $\rho(i+1)$ 的第 k 个目标函数, 在实际应用中，目标函数可以定义为路径长度、安全指数或者舒适度等。单目标旅行商问题是典型的 NP 难问题，因此多目标旅行商问题的求解更加困难。

在过去的几十年中，多目标进化算法 (Multi-objective Evolutionary Algorithms, MOEAs) 被广泛用于求解多目标组合优化问题，它们采用基于种群的方式进行解空间的探索，从而可以通过一次算法运行得到一组非支配解集。NSGA-II [112] 和 MOEA/D [113] 是该领域最广泛应用的多目标进化算法，这两种算法也已成功应用于求解多目标旅行商问题[114–116]。

其他启发式搜索方式例如 Lin-Kernighan 搜索 [117] 和 2-opt 局部搜索 [118] 等也

常被应用于求解该类问题，这些方法基于专家经验，设计不同的启发式算子对多目标旅行商问题进行求解，这类方法包括帕累托局部搜索方法 (PLS) [119]、多目标遗传局部搜索算法 (MOGLS) [120] 以及其他类似的变体[121–123]，更多求解多目标组合优化问题的方法可参考综述 [124]。

多年来，进化算法和其他启发式搜索算法是求解多目标组合优化问题的主要方法。然而，这些算法采用迭代的方式进行解空间的探索，面临多种局限[124–126]：首先，为了寻找近似最优解，特别是在问题维数较大的情况下，通常需要大量的迭代进行种群更新或迭代搜索，从而导致算法求解慢甚至无法求解；其次，一旦问题稍有变化，例如改变了多目标旅行商问题的城市坐标，需要重新执行算法来求解该问题，因此很难进行在线实时优化，并且算法通常只针对某个特定问题，算法的设计需要大量专家知识和试错成本[127]。值得注意的是，“多目标 (深度) 强化学习”的概念[91, 128] 很早就被提出并且存在部分文献对其进行研究, 但是其研究的是如何设计具有多个奖励信号的强化学习算法, 例如如何利用强化学习算法控制潜艇寻找目标[128], 其中需要最大化寻找到目标的数量和最小化寻找的时间, 其研究的主体是多目标强化学习算法, 而不是如何利用强化学习方法解决传统的多目标优化问题。

在该背景下，本章对基于深度强化学习的多目标组合优化方法开展了研究，提出了一种基于深度强化学习的多目标旅行商组合优化方法，采用端到端的方式输出帕累托最优解，无需迭代搜索，从而可以实现多目标组合优化问题的快速求解。

4.2 基于深度神经网络的的多目标旅行商问题建模

由于旅行商问题具有典型的序列特征，因此可以通过指针网络模型对旅行商问题进行建模，但是多目标旅行商问题具有明显的挑战，需要设计一个有效的机制能够以端到端的方式输出问题的帕累托最优解集，而不是仅仅输出单个解，为了解决该问题，本章采用分解策略和基于领域的参数迁移策略对多目标组合优化问题进行建模，首先采用分解策略[113] 将多目标优化问题分解为多个子问题，每个子问题都基于指针网络模型建模为一个神经网络，然后根据基于邻域的参数迁移策略和强化学习训练算法对所有子问题的神经网络参数进行协同优化，从而实现多目标组合优化问题的求解。

4.2.1 分解策略

分解策略是解决多目标优化问题的一种经典方法，大量多目标优化算法采用分解策略进行方法设计，例如 MOGA[129]、MOEA/D、MOEA/DD [130] 和 NSGA-III

[131]。本章采用分解策略对多目标旅行商问题进行建模，将多目标旅行商问题分解为一组标量的优化子问题，以协同优化的方式对子问题进行求解，得到针对每个子问题的最优解，所有子问题的解构成原问题的的帕累托前沿（Pareto Front, PF）。

采用广泛使用的 Weighted Sum [132] 方法对原问题进行分解，也可以应用其他方法，例如 Chebyshev 方法和基于惩罚的边界交叉（PBI）方法 [133, 134] 对多目标优化问题进行分解。本章采用的分解方法如下：首先给定一组均匀分布的权重向量 $\lambda^1, \ldots, \lambda^N$，例如针对双目标旅行商问题，权重向量为 $(1,0),(0.9,0.1),\ldots,(0,1)$，这里 $\lambda^j = (\lambda_1^j, \ldots, \lambda_M^j)^T$，其中 M 代表目标的数量。因此原多目标旅行商问题可以通过 Weighted Sum 方法转换为 N 个标量优化子问题，第 j 个子问题的目标函数如下[113] 所示：

$$\text{minimize}\, g^{ws}(x|\lambda_i^j) = \sum_{i=1}^{M} \lambda_i^j f_i(x) \tag{4.3}$$

因此帕累托前沿可以由通过解决 N 个子问题得到的 N 个解构成，该过程如图 4.1 所示，虚线代表子问题优化的方向, 通过式 (4.3) 将原问题分解为 N 个子问题，并进一步采用深度神经网络对子问题进行建模，从而实现多目标组合优化问题的求解。

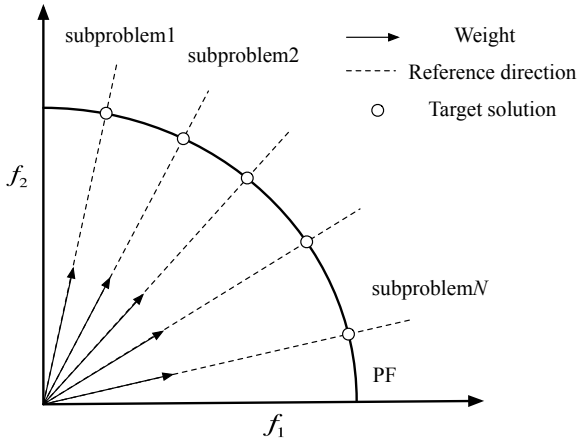


图 4.1　分解策略示意图

4.2.2　基于邻域的参数迁移策略

为了采用深度强化学习方法解决每个子问题，需要将子问题建模为深度神经网络，然后对 N 个标量优化子问题通过基于邻域的参数迁移策略以协同优化的方式进行求解，基于领域的参数迁移策略介绍如下

根据式 (4.3) 可以观察到，对于两个相邻的子问题，由于它们的权重向量是相似的，因此它们的最优解相似[113]，根据该事实，每个子问题都可以借鉴其相邻子

问题的信息来解决。

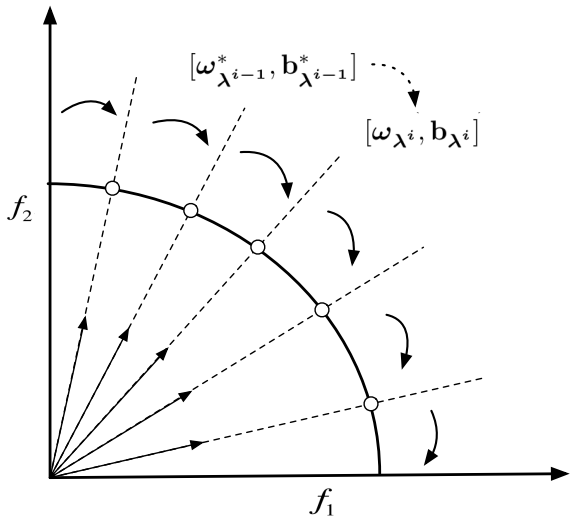


图 4.2　基于邻域的参数迁移策略示意图

由于本章中的子问题被建模为深度神经网络，因此第 $(i-1)$ 个子问题的神经网络模型参数可以表示为 $[\boldsymbol{\omega}_{\boldsymbol{\lambda}^{i-1}}, \mathbf{b}_{\boldsymbol{\lambda}^{i-1}}]$，$[\boldsymbol{\omega}^*, \mathbf{b}^*]$ 表示已经最优的神经网络模型参数，$[\boldsymbol{\omega}, \mathbf{b}]$ 表示尚未优化的神经网络模型参数。假设第 $(i-1)$ 个子问题被求解，即其神经网络参数已优化至接近最优，那么可以将第 $(i-1)$ 个子问题的模型参数 $[\boldsymbol{\omega}^*_{\boldsymbol{\lambda}^{i-1}}, \mathbf{b}^*_{\boldsymbol{\lambda}^{i-1}}]$ 设置为第 i 个子问题神经网络模型的初始参数，从该初始参数开始进行训练，即将邻域子问题的神经网络模型参数迁移到下个相邻子问题上，从而极大降低模型训练时间，存在多种迁移方式，本章中采用最基础的参数迁移方法，即神经网络参数按顺序地从前一个子问题迁移到下一个子问题，该过程如图 4.2 所示。

如果不采用该基于邻域的参数迁移策略，训练 N 个建模为神经网络模型的子问题将会需要大量时间，无法进行实际应用，本章所采用的基于邻域的参数迁移策略使得该多目标组合优化方法能够实际应用。

因此，每个子问题都由深度神经网络进行建模，子问题通过强化学习方法进行求解，所有子问题都可以通过参数迁移策略进行协同优化，通过得到的神经网络模型集合即可对帕累托前沿进行端到端的输出。结合分解策略和基于邻域的参数迁移策略，基于深度强化学习的多目标组合优化方法流程如算法 4.1 所示。

该方法的明显优势是其模块化和易用性，不仅可以应用于多目标旅行商问题中，还可以将任意基于深度学习的单目标组合优化模型[106, 135] 集成到该方法里，只需要对子问题的模型进行替换，便可以求解任意的多目标组合优化问题，一旦模型训练完毕，可以通过神经网络模型直接输出问题的帕累托前沿。分解策略和参数迁移策略是该方法的外环结构，接下来需要对分解后的标量子问题进行建模。

算法 4.1 基于深度强化学习的多目标组合优化方法流程

算法输入: 基于深度神经网络模型对子问题进行参数化建模，模型参数为 $\mathcal{M} = [\boldsymbol{w}, \mathbf{b}]$, 权重向量为 $\lambda^1, \ldots, \lambda^N$

算法输出: 神经网络模型最优参数 $\mathcal{M}^* = [\boldsymbol{w}^*, \mathbf{b}^*]$

$[\boldsymbol{\omega}_{\boldsymbol{\lambda^1}}, \mathbf{b}_{\boldsymbol{\lambda^1}}] \leftarrow Random\_Initialize$
for $i \leftarrow 1 : N$ **do**
    if $i == 1$ **then**
        $[\boldsymbol{\omega}^*_{\boldsymbol{\lambda^1}}, \mathbf{b}^*_{\boldsymbol{\lambda^1}}] \leftarrow Optimize([\boldsymbol{\omega}_{\boldsymbol{\lambda^1}}, \mathbf{b}_{\boldsymbol{\lambda^1}}], g^{ws}(\boldsymbol{\lambda^1}))$
    else
        $[\boldsymbol{\omega}_{\boldsymbol{\lambda^i}}, \mathbf{b}_{\boldsymbol{\lambda^i}}] \leftarrow [\boldsymbol{\omega}^*_{\boldsymbol{\lambda^{i-1}}}, \mathbf{b}^*_{\boldsymbol{\lambda^{i-1}}}]$
        $[\boldsymbol{\omega}^*_{\boldsymbol{\lambda^i}}, \mathbf{b}^*_{\boldsymbol{\lambda^i}}] \leftarrow Optimize([\boldsymbol{\omega}_{\boldsymbol{\lambda^i}}, \mathbf{b}_{\boldsymbol{\lambda^i}}], g^{ws}(\boldsymbol{\lambda^i}))$
    end if
end for
return $[\boldsymbol{w}^*, \mathbf{b}^*]$
给定多目标旅行商问题的输入，通过模型 $[\boldsymbol{w}^*, \mathbf{b}^*]$ 直接输出问题的帕累托前沿

4.2.3 多目标旅行商子问题建模

多目标旅行商问题具有序列特性，因此可以采用指针网络模型[135] 对多目标旅行商问题的子问题进行建模，并采用强化学习算法对模型参数进行训练。本节对经典指针网络[135] 模型进行修改，以适应多目标组合优化问题的子问题结构，实现对子问题的建模。

首先介绍该子问题神经网络网络模型的输入和输出结构: 输入是 $X \doteq \{s^i, i = 1, \cdots, n\}$，其中 n 是城市的数量。每个 s^i 由一个元组 $\{s^i = (s^i_1, \cdots, s^i_M)\}$ 表示。s^i_j 是用于计算第 j 个目标函数的第 i 个城市的属性。例如，$s^i_1 = (x_i, y_i)$ 表示第 i 个城市的 x 坐标和 y 坐标，用于计算两个城市之间的距离。

以双目标问题为例，假设两个目标函数均由欧几里得距离[124] 定义。那么子问题的模型输入结构如图 4.3 所示，即输入是四维的，对于 n 个城市则有 $4 \times n$ 个输入，模型的输出是城市的排列 $Y = \{\rho_1, \cdots, \rho_n\}$。

进一步地，采用条件概率的链式法则将输入 X 映射到输出 Y:

$$P(Y|X) = \prod_{t=1}^{n} P(\rho_{t+1}|\rho_1, \cdots, \rho_t, X_t). \tag{4.4}$$

首先选择任意城市作为 ρ_1，在每个解码步骤 $t = 1, 2, \cdots$ 中，从可用城市集合 X_t 中选择 ρ_{t+1}，每次访问城市时都会更新可用城市 X_t，其中式 (4.4) 代表根据 $\rho_1, \cdots, \rho_t$ 选择下一个城市的概率。

根据上述多目标旅行商子问题的模型输入、输出定义，本节采用改进的指针

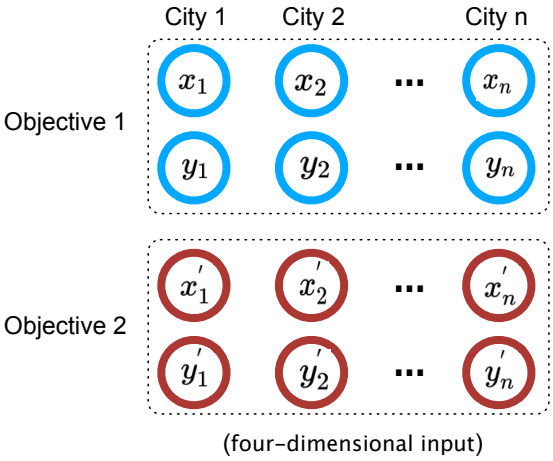


图 4.3　子问题模型输入结构

网络模型对子问题进行建模[135]，编码器将输入序列编码为包含问题信息的定长向量，根据该定长向量，解码器将向量解码为输出序列，模型的架构如图 4.4 所示，其中左侧部分是编码器，右侧部分是解码器，模型详述如下。

4.2.3.1　编码器设计

编码器用于将输入序列压缩为向量，传统编码器为循环神经网络，但是由于城市的坐标并不存在序列信息 [135]，即输入序列中城市的顺序没有意义，因此没有必要采用循环神经网络对问题输入进行编码，而且循环神经网络的训练和输出耗时严重，针对本章所提出的多目标组合优化框架，由于存在大量神经网络参数，因此降低神经网络训练和计算的耗时是一个重要的问题，因此本节不采用循环神经网络作为编码器。而采用一个简单的全连接神经网络将问题输入映射为节点特征向量，从而降低模型复杂度、降低计算成本。

具体地，采用双层全连接神经网络将子问题的输入编码为维度为 d_h 的高维节点特征向量 e，本节中 $d_h = 128$，如图 4.4 所示，输入通道的数量等于子问题输入的维度。例如，针对两个目标均定义为欧几里得距离的双目标旅行商问题，问题的输入为四维，如图 4.3 所示，因此输入通道的数量是 4，编码器最终得到一个 $n \times d_h$ 向量，其中 n 表示城市数量。值得注意的是，该全连接神经网络参数在所有城市之间共享，因此无论有多少个城市，每个城市都共享相同的参数集，这些参数将城市信息编码为高维向量，因此编码器对问题的规模具有鲁棒性。

4.2.3.2　解码器设计

解码器用于将存储输入信息的高维向量以顺序的方式展开为输出序列。与编码器不同，解码器中需要采用循环神经网络进行建模，因为需要汇总先前选择城市 $\rho_1, \cdots, \rho_t$ 的信息，以便做出 ρ_{t+1} 的决策。因此需要采用具有序列记忆能力的循环神经网络。

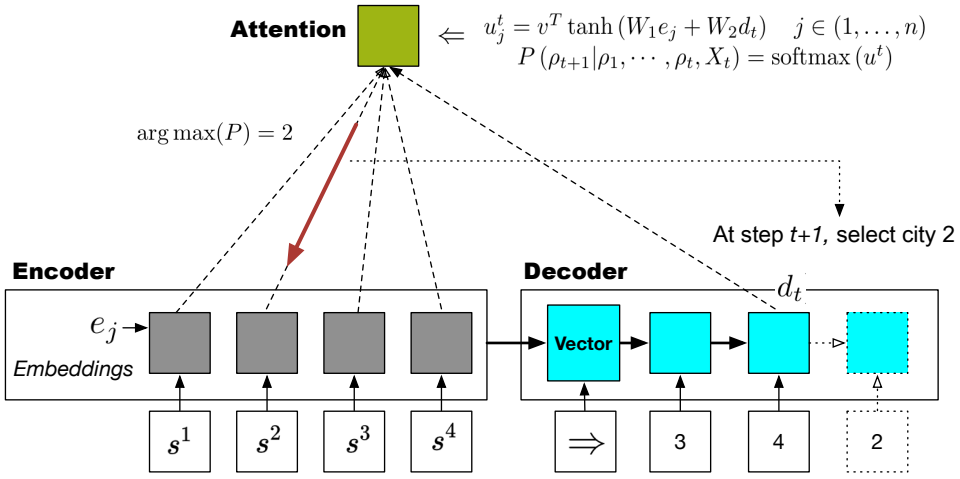


图 4.4　基于指针网络的子问题模型结构

但针对多目标组合优化问题，需要尽量缩小模型计算复杂度以降低 N 个神经网络模型的训练和推理的计算压力，因此本节采用 GRU 循环神经网络[88] 作为解码器，该模型具有与 [135] 中原始指针网络采用的 LSTM 循环神经网络相似的性能，但 GRU 的参数更少，GRU 循环神经网络的计算方法如下：

$$\begin{aligned} z_t &= \sigma_g\left(W_z x_t + U_z h_{t-1} + b_z\right) \\ r_t &= \sigma_g\left(W_r x_t + U_r h_{t-1} + b_r\right) \\ \hat{h}_t &= \phi_h\left(W_h x_t + U_h\left(r_t \odot h_{t-1}\right) + b_h\right) \\ h_t &= \left(1 - z_t\right) \odot h_{t-1} + z_t \odot \hat{h}_t \end{aligned} \tag{4.5}$$

其中 W, U, b 代表该模型的参数，h_t 为解码器每一步的输出，用来计算节点选择的概率。需要注意的是，GRU 模型不直接用于输出子问题的解序列，解码器在第 t 步解码过程中输出的隐层状态 d_t 用于存储先前选择城市 $\rho_1, \cdots, \rho_t$ 的信息，然后 d_t 和编码器得到的节点特征向量 $e_1, \cdots, e_n$ 一起用于计算下一步城市选择的条件概率 $P\left(y_{t+1}|\rho_1, \cdots, \rho_t, X_t\right)$。这个计算是通过注意力机制来实现的，如图 4.4 所示，为了在步骤 $t+1$ 选择下一个城市，首先通过解码器获得隐层状态 d_t，代表当前解的状态，基于注意力机制结合 $e_1, \cdots, e_n$ 计算下一步选择各个城市的概率。

4.2.3.3 注意力机制

采用如下注意力计算方法对下一步节点选择的概率进行计算：

$$
\begin{aligned}
u_j^t &= v^T \tanh\left(W_1 e_j + W_2 d_t\right) \quad j \in (1, \ldots, n) \\
a_t &= \operatorname{softmax}\left(u^t\right) \\
c_t &= \sum_{i=1}^{n} a_t^i e^i \\
\tilde{u}_t^j &= v_c^T \tanh\left(W_c\left[e_j; c_t\right]\right) \quad j \in (1, \ldots, n) \\
\mathrm{P}\left(\rho_{t+1} \mid \rho_1, \cdots, \rho_t, X_t\right) &= \operatorname{softmax}\left(\tilde{u}_t^j\right)
\end{aligned}
\tag{4.6}
$$

其中 v, W_1, W_2, W_c 是全连接神经网络的参数。

传统指针网络直接采用 a_t 作为节点选择的概率，但是由于本章所设计的编码器结构采用简单的全连接神经网络进行节点特征向量的提取，因此会降低特征的表示精度，因此本章在注意力计算过程中，借鉴自注意力机制的计算方法，即对输入信息进行注意力加权从而提高节点特征向量表示能力，利用 a_t 对原节点特征向量 e 进行加权得到加权后向量 c_t，从而获得更加丰富的节点特征向量表示，最后利用参数化的神经网络 W_c 对 c_t 和 e 进行整合，从而计算得到最终的节点选择概率。

该注意力计算过程相对于增加了一次注意力计算深度，中间层借鉴了基于自注意机制的特征提取的想法提高了节点特征向量的表示能力，通过该过程，只增加了 W_c 参数，没有对模型复杂度进行很大的增加，弥补了为降低模型复杂度对编码器进行简单设计而导致的精度损失。

在解码过程中，d_t 是计算 $P\left(\rho_{t+1}|\rho_1, \cdots, \rho_t, X_t\right)$ 的关键变量，因为它存储了已选城市 $\rho_1, \cdots, \rho_t$ 的信息，对于每个城市 j，其 $\tilde{u}_j^t$ 由 d_t 及其编码器隐层状态 e_j 计算 ，如图 4.4 所示。使用 softmax 算子对 $\tilde{u}_1^t, \cdots, \tilde{u}_n^t$ 进行归一化，最终可以最终获得在步骤 t 选择每个城市 j 的概率，可以采用贪婪策略选择下一个城市，例如在图 4.4 中，城市 2 的 $P\left(\rho_{t+1}|\rho_1, \cdots, \rho_t, X_t\right)$ 最大，因此被选为下一个访问城市。但在训练过程中，不采用贪婪策略选择概率最大的城市，而是通过从模型输出的条件概率分布中进行随机抽样。

通过以上方法，采用分解策略将多目标旅行商问题分解为多个标量子问题，采用改进的指针网络模型对子问题进行建模，最后结合基于邻域的参数迁移策略和强化学习方法对子问题模型进行训练，从而得到最终的模型。

4.3 模型训练方法

借鉴邻域子问题的相似特征，采用基于邻域的参数迁移策略对模型训练过程进行加速，每个子问题模型均采用强化学习方法进行训练，从而以协同优化的方式实现所有子问题模型的训练。子问题模型采用类似于文献 [135, 136] 所采用的 Actor-critic 强化学习方法进行训练。但是由于文献 [135, 136] 的方法是用来训练单目标旅行商模型，而本章需要对多目标旅行商问题的子问题进行训练，因此对模型训练过程进行调整，具体训练算法如算法 4.2 所示，模型训练过程介绍如下。

训练过程中需要同时对两个神经网络进行训练：1）Actor 神经网络，即本章所提出的改进的指针网络模型，用于给出选择下一个动作的概率分布；2）Critic 神经网络，给定一个问题实例，用于评估该问题的预期目标值，即路径长度。Critic 神经网络采用与 Actor 神经网络编码器相同的架构，然后增加两个全连接层将编码器的隐层状态映射为单维输出，用于对问题的目标值进行近似。

训练以无监督的方式进行，在训练期间，从分布 $\{\Phi_{\mathcal{M}_1}, \cdots, \Phi_{\mathcal{M}_M}\}$ 中随机生成多目标旅行商问题实例，这里 $\mathcal{M}$ 表示城市的不同输入特征，比如城市的位置或者城市的安全指数。例如，对于双目标旅行商问题，假设两个目标都由欧几里得距离定义，则 $\mathcal{M}_1$ 和 $\mathcal{M}_2$ 都是坐标值，这种情况下 $\Phi_{\mathcal{M}_1}$ 和 $\Phi_{\mathcal{M}_2}$ 均为 $[0,1] \times [0,1]$ 的均匀分布。

对 Critic 神经网络的参数 ϕ 与 Actor 神经网络的参数 θ 进行优化的过程如下。首先，对于每个子问题，按照算法4.1，基于参数迁移策略对当前子问题模型的参数进行初始化。其次，每回合从 $\{\Phi_{\mathcal{M}_1}, \cdots, \Phi_{\mathcal{M}_M}\}$ 中采样 N 个问题实例进行训练。对于每个训练问题实例，使用具有当前参数 θ 的 Actor 网络来生成当前问题的解，并且根据该解和目标函数计算方法，计算其 M 个目标值 R_i^k，以当前子问题的权重向量 λ 计算加权求和后的当前子问题的目标值 R_i^k。同时，使用具有当前参数 ϕ 的 Critic 网络对该问题实例 X_0^k 的目标值进行近似 $V\left(X_0^k;\phi\right)$。最后，基于策略梯度方法，根据式 (4.7) 进行 Actor 神经网络以及 Critic 神经网络参数的更新：

$$
\begin{aligned}
d\theta &\leftarrow \frac{1}{N}\sum_{k=1}^{N}\left(R^k - V\left(X_0^k;\phi\right)\right)\nabla_\theta \log P\left(Y^k \mid X_0^k\right)\\
d\phi &\leftarrow \frac{1}{N}\sum_{k=1}^{N}\nabla_\phi\left(R^k - V\left(X_0^k;\phi\right)\right)^2\\
\theta &\leftarrow \theta + \eta d\theta\\
\phi &\leftarrow \phi + \eta d\phi
\end{aligned}
\tag{4.7}
$$

$V\left(X_0^n;\phi\right)$ 代表由 Critic 神经网络计算得到的第 n 个子问题的目标函数近似值，通过减少实际目标值和近似值之间的差异来优化 Critic 神经网络的参数。

算法 4.2 子问题模型 Actor-Critic 强化学习训练算法

算法输入： $\theta,\phi \leftarrow$ 按照算法 4.1 对子问题模型参数基于参数迁移策略进行初始化，当前子问题的权重向量 λ

算法输出： 子问题模型最优参数 θ^*,ϕ^*

for $iteration \leftarrow 1,2,\cdots$ **do**
  从分布 $\{\Phi_{\mathcal{M}_1},\cdots,\Phi_{\mathcal{M}_M}\}$ 中生成 T 个多目标旅行商问题训练实例
  for $k \leftarrow 1,\cdots,T$ **do**
    $t \leftarrow 0$
    while not terminated **do**
      根据模型输出的条件概率分布 $P\left(\rho_{t+1}^k|\rho_1^k,\cdots,\rho_t^k,X_t^k\right)$ 进行节点选择决策 ρ_{t+1}^k
      更新 X_t^k 至 X_{t+1}^k
    end while
    对于多目标优化问题的 M 个目标
    for $i \leftarrow 1,\cdots,M$ **do**
      计算当前解的第 i 个目标函数值 R_i^k
    end for
    根据子问题权重向量以 Weighed Sum 方式计算 $R^k=\sum_{i=1}^{M}\lambda_i R_i^k$
  end for
  $d\theta \leftarrow \frac{1}{N}\sum_{k=1}^{N}\left(R^k-V\left(X_0^k;\phi\right)\right)\nabla_\theta \log P\left(Y^k|X_0^k\right)$
  $d\phi \leftarrow \frac{1}{N}\sum_{k=1}^{N}\nabla_\phi\left(R^k-V\left(X_0^k;\phi\right)\right)^2$
  $\theta \leftarrow \theta+\eta d\theta$
  $\phi \leftarrow \phi+\eta d\phi$
end for

一旦子问题模型训练完毕，就可以通过对神经网络模型的前向传播直接输出多目标旅行商问题的帕累托最优解集。编码器前向计算的时间复杂度为 $O(d_h n)$，其中 d_h 为隐含层向量的维度，解码器前向计算的时间复杂度为 $O(d_h^2 n)$，其中 $O(d_h^2)$ 是循环神经网络的近似时间复杂度。因此，使用该方法解决多目标组合优化问题的近似时间复杂度为 $O(Nnd_h^2)$，其中 N 是子问题的数量，由于神经网络模型的前向传播速度非常快，因此总是可以在合理的时间内获得问题的解。

4.4 实验结果与讨论

4.4.1 实验设置

为了验证本章基于深度强化学习的多目标旅行商组合优化方法（DRL-MOA）的有效性，分别在 2、3、5 个优化目标的旅行商问题上进行测试，并对高达 500 个节点的问题进行测试，所有实验均在单个 GTX 2080Ti GPU 上进行，代码基于 Python 编写，并且进行开源 [1] 以对实验结果进行复现。同时，用于对比的多目标进化算法均在标准软件平台 PlatEMO[2][137] 上实现，采用 MATLAB 编写，用于对比的 MOGLS 启发式算法采用 Python[3] 编写，所有对比算法都在具有 64GB 内存的 Intel 16 核 i7-9800X CPU 上运行。

4.4.1.1 测试问题

为了验证所提方法的有效性，采用以下两种多目标旅行商问题进行测试 [124]：

欧几里得测试问题：欧几里得测试问题是求解 MOTSP [124] 的常用测试问题，该测试问题的两个目标函数都由欧几里德距离定义：第一个目标由两个城市 i, j 实际地理坐标之间的欧式距离定义；城市 i 和城市 j 的第二个目标值由另一组虚拟坐标定义，该目标值代表城市 i, j 之间的诸如安全系数、舒适度等指标。因此输入是四维的，输入结构如图 4.3 所示。

混合测试问题：为了测试本章所提模型具有适应不同输入结构的能力，进一步对具有三维输入的混合测试问题进行模型验证。这里，第一个目标函数仍然由代表真实城市位置的两点之间的欧几里得距离定义，它是一个具有 x 坐标和 y 坐标的二维输入。此外，从城市 i 到 j 的第二个目标函数由一维输入定义，因此混合测试问题具有三维输入。

4.4.1.2 测试方法

作为无监督的训练方法，强化学习在训练过程中只需要组合优化问题的输入信息和目标值的计算方法，不需要最优路径作为标签。因此从 $[0, 1]$ 的均匀分布中，随机生成四维输入的欧几里得实例和三维输入的混合实例，训练集所有问题的城市规模为 40。

使用 TSPLIB 库[111] 中的标准 TSP 测试问题 kroA 和 kroB 构建欧几里得测试实例 kroAB100、kroAB150 和 kroAB200，这是常用的多目标旅行商问题的测试实例[123, 124]，kroA 和 kroB 是两组不同的城市位置，用于计算由欧几里得距离定义的

[1] https://github.com/kevin031060/RL_TSP_4static

[2] http://bimk.ahu.edu.cn/index.php?s=/Index/Software/index.html

[3] https://github.com/kevin031060/Genetic_Local_Search_TSP

目标函数。对于混合测试实例，采用从 $[0,1]$ 均匀分布中随机生成的 40、70、100、150 和 200 三维问题输入构建得到混合实例的测试集。

模型在 40 个城市的 MOTSP 实例上进行训练，用于求解 40 个、70 个、100 个、150 个、200 个以及 500 个城市的多目标旅行商问题。

4.4.1.3 实验参数设置

子问题的模型参数设置如表 4.1 所示。D_{input} 表示输入的维度，例如欧几里德测试实例的 $D_{input}=4$。在解码器中使用隐层维度为 128 的单层 GRU 循环神经网络，Critic 神经网络的隐层维度也设置为 128。

表 4.1 子问题模型参数设置，MLP 为全连接神经网络

Actor 神经网络 (指针网络)	
Encoder:	MLP(D_{input}, 128)
Decoder:	GRU(hidden size=128, number of layer=1)
	Attention(No hyper parameters)
Critic 神经网络	
MLP(D_{input}, 128)	
MLP(128, 20)	
MLP(20, 20)	
MLP(20, 1)	

使用 Adam 优化器 [109] 以 0.0001 的学习率和大小为 200 的 mini-batch 训练 actor 和 critic 神经网络模型，采用 *Xavier* 初始化方法 [138] 初始化第一个子问题的权重。

进一步地，从第二个子问题的训练开始，子问题的权重采用基于邻域的参数迁移策略进行初始化。

此外，训练不同类型的模型需要不同规模的测试集。与混合 MOTSP 问题相比，欧几里得 MOTSP 问题的输入维度更大，因此模型需要更多的参数来优化，每次迭代需要更大规模的训练集。因此，生成 500,000 个问题实例来训练欧几里德 MOTSP 问题，生成 120,000 个实例来训练混合 MOTSP 问题。所有问题实例均从 $[0,1]$ 的均匀分布中生成，并用于 5 个 epoch 的训练，由于采用了基于邻域的参数迁移策略，因此只需要少量回合的训练模型即可收敛。训练混合 MOTSP 问题大约需要 3 个小时，训练欧几里得 MOTSP 问题大约需要 7 个小时，模型训练成功

后，可以直接输出问题的帕累托前沿。

本节将 DRL-MOA 方法与文献中常用的多目标进化算法 NSGA-II 和 MOEA/D 进行实验对比，并与求解多目标旅行商问题的经典启发式方法 MOGLS 进行实验对比。NSGA-II 和 MOEA/D 的最大迭代次数分别设置为 500、1000、2000 和 4000。NSGA-II 和 MOEA/D 的种群规模设置为 100。DRL-MOA 的子问题数也设置为 100，此外，在最终得到的问题的解集中只保留了非支配解。

4.4.2 混合测试问题实验结果

首先在 40 城市规模的混合类型双目标 TSP 问题上进行模型训练，然后使用该模型来求解 40、70、100、150 和 200 城市规模的混合双目标 TSP 测试问题。

图 4.5、4.6、4.7、4.8 可视化了所有对比算法在各个测试问题上得到的帕累托前沿。可以观察到，本章设计的模型虽然只在 40 规模的问题上进行了训练，但是可以有效地扩展到具有不同城市数量的多目标组合优化问题上，在 70 个、100 个、150 个和 200 个城市规模的测试问题上表现出了良好的性能。此外，表 4.2 给出了各个算法的 Hypervolume（HV）性能指标和求解时间，所有实验结果都通过独立执行五次算法并取平均值得到。

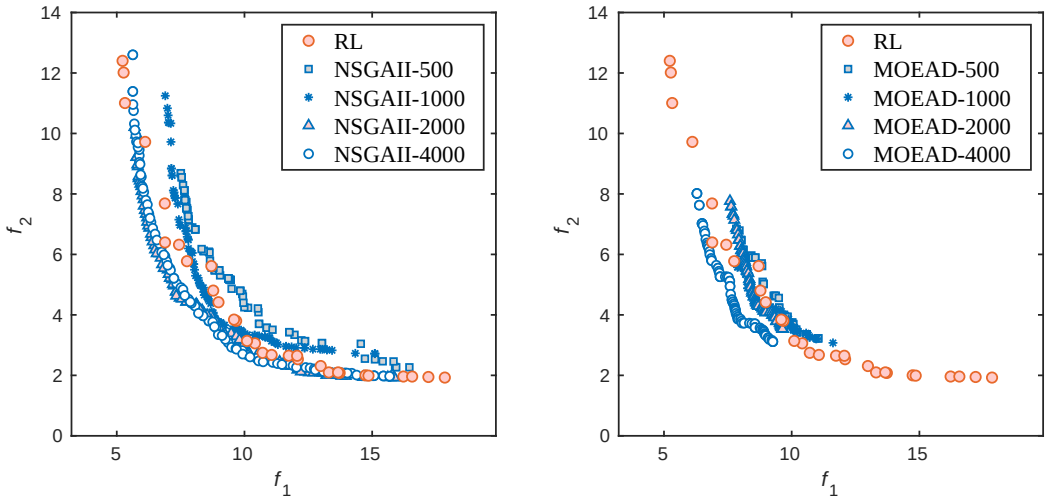
如图 4.5 所示，所有对比算法都可以很好地解决小规模多目标旅行商问题，针对传统的多目标进化算法，在 40 城市规模的测试问题上，可以通过增加迭代次数使得 NSGA-II 和 MOEA/D 表现出很好的收敛能力，但是增加迭代次数会导致额外的求解时间。例如 4000 次迭代对于 MOEA/D 需要 130.2 秒，对于 NSGA-II 需要 28.3 秒，而本章所提方法只需要 2.7 秒，因此本章所提方法在达到相似的优化水平的情况下，可以极大地降低求解时间。

从图 4.6、4.7、4.8 可以看出，随着城市数量的增加，NSGA-II 和 MOEA/D 逐渐难以收敛，而 DRL-MOA 表现出明显更好的收敛能力。

对于图 4.6 中的 100 城市规模的测试问题，MOEA/D 通过在 140.3 秒内进行 4000 次迭代，可以在收敛能力方面表现出比其他方法稍好的性能，然而本章所提方法输出的帕累托前沿与 MOEA/D 相比具有更好的多样性。

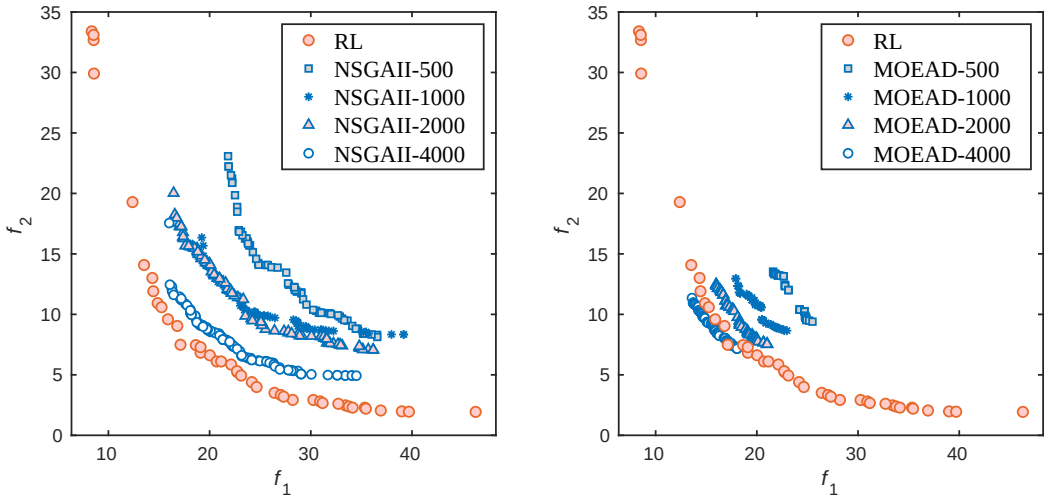
对于图 4.7 和图 4.8 中给出的 150、200 节点的测试问题，本章所提方法在收敛性和多样性方面明显优于 NSGA-II 和 MOEA/D。尽管 NSGA-II 和 MOEA/D 进行了 4000 次迭代搜索（相当多的迭代次数，文献中通常只进行几百次迭代），但 DRL-MOA 仍然表现出比他们更加好的优化性能。

如表 4.2 所示，与其他算法相比，DRL-MOA 总是能够得到最优的 HV 指标值，并且本章所提方法求解速度更快，该实验结果验证了本章所提方法 DRL-MOA 的有效性。相对于传统的多目标进化算法，它的优化能力和求解时间不会随着城市数量的增加而出现明显恶化。相比之下，在求解大规模多目标组合优化问题



(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.5　算法在 40 城市混合类型双目标 TSP 问题上的帕累托前沿



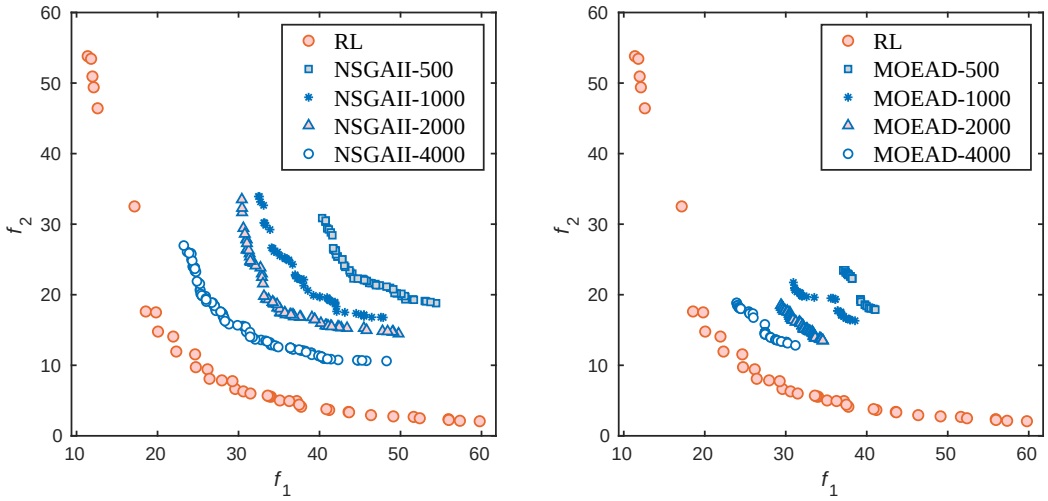
(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.6　算法在 100 城市混合类型双目标 TSP 问题上的帕累托前沿

时，NSGA-II 和 MOEA/D 无法在短时间内实现算法收敛。并且本章所提方法与 NSGA-II 和 MOEA/D 相比，能够获得多样性更加好的帕累托前沿。

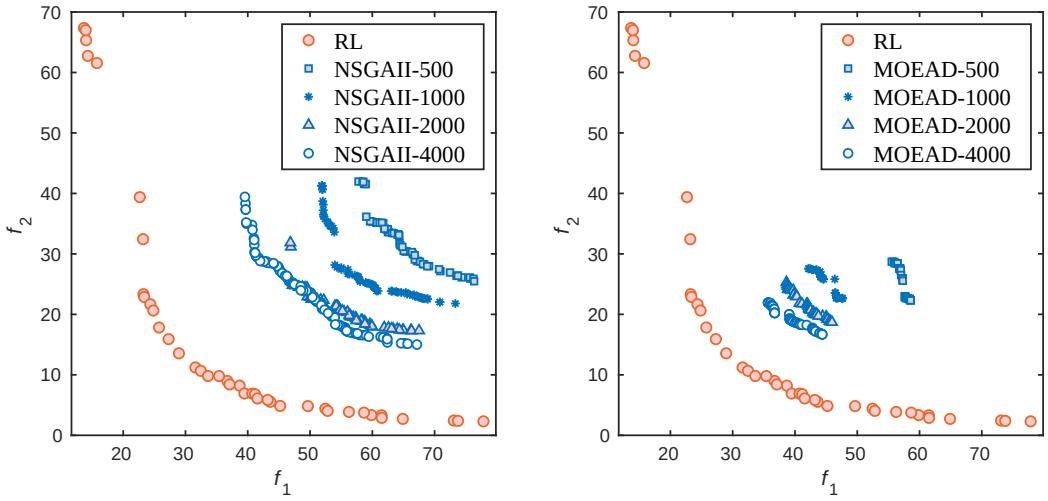
4.4.3　欧几里得测试问题实验结果

其次对第二类欧几里得问题进行测试，同样的，在 40 节点的双目标 TSP 问题上进行模型训练，该模型用于求解 40、70、100、150 和 200 城市规模的测试问题，对于 100、150 和 200 城市规模的测试问题，采用经典的的 kroAB100、kroAB150 和 kroAB200 测试集 [124]，表 4.3 给出了多目标优化的 HV 指标和求解时间。



(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.7　算法在 150 城市混合类型双目标 TSP 问题上的帕累托前沿



(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.8　算法在 200 城市混合类型双目标 TSP 问题上的帕累托前沿

图 4.9、图 4.10 和图 4.11 可视化了 DRL-MOA 算法在 kroAB100、kroAB150 和 kroAB200 测试问题上的解。对于 kroAB100，通过将 NSGA-II 和 MOEA/D 的迭代次数增加到 4000，可以达到与 DRL-MOA 相似的收敛水平，MOEA/D 的性能稍好一些，但是 MOEA/D 在多样性方面表现最差，并且需要大量求解时间。

当城市数量增加到 150 和 200 时，DRL-MOA 在收敛性和多样性方面都显著优于 NSGA-II 和 MOEA/D，从图 4.10 和图 4.11 可见，尽管 NSGA-II 和 MOEA/D 进行了 4000 次迭代，但它们的优化性能与 DRL-MOA 仍然存在明显差距。

从表 4.3 给出的 HV 指标可见，DRL-MOA 在所有测试问题上都表现最好，并

表 4.2　算法在不同规模混合双目标 TSP 问题上的实验结果

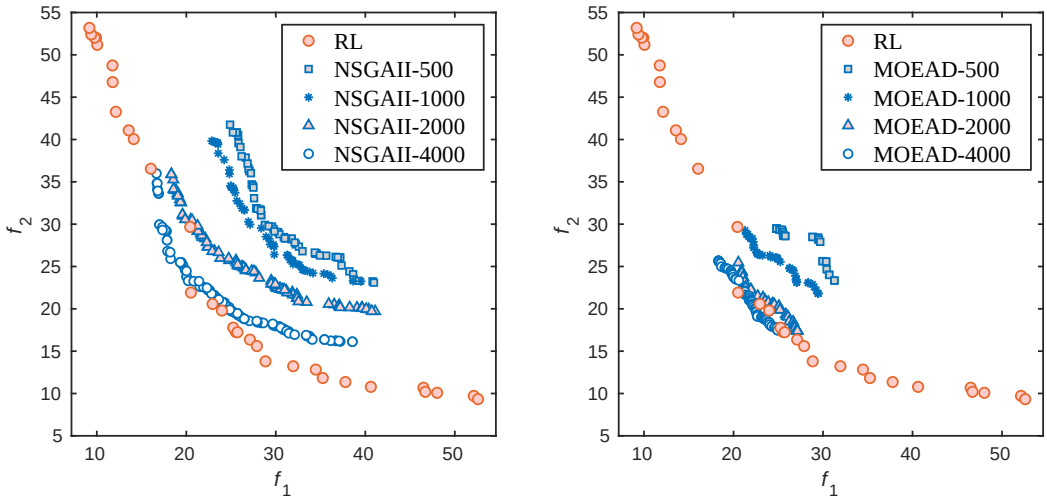
	40-city		70-city		100-city		150-city		200-city	
	HV	Time/s	HV	Time/s	HV	Time/s	HV	Time/s	HV	Time/s
NSGAII-500	1282	4.1	3866	4.2	7186	4.6	15158	5.7	26246	6.5
NSGAII-1000	1345	7.0	4042	9.6	7717	8.7	16313	12.6	27557	11.9
NSGAII-2000	1366	13.3	4146	16.8	8218	16.3	17283	21.4	29206	23.3
NSGAII-4000	1404	28.3	4434	32.7	8597	33.2	18267	40.5	31647	51.2
MOEA/D-500	1251	17.0	3878	17.7	7367	18.5	15796	20.5	26548	21.8
MOEA/D-1000	1305	34.5	4048	35.2	7796	35.9	16838	40.6	28851	41.9
MOEA/D-2000	1324	65.2	4166	68.5	8261	73.2	17833	79.4	30785	85.5
MOEA/D-4000	1346	**130.2**	4235	**136.0**	8471	**145.2**	18644	**157.6**	32642	**169.2**
DRL-MOA	1398	2.7	4668	4.7	9647	6.6	22386	10.1	40354	12.9

表 4.3　算法在不同规模欧几里得双目标 TSP 问题上的实验结果

	40-city		70-city		100-city		150-city		200-city	
	HV	Time/s	HV	Time/s	HV	Time/s	HV	Time/s	HV	Time/s
NSGAII-500	1498	3.8	4446	4.1	8738	4.3	18487	5.2	31430	5.9
NSGAII-1000	1547	7.2	4643	7.7	9119	8.1	19491	9.6	33424	11.0
NSGAII-2000	1587	13.4	4798	15.7	9577	15.6	20116	20.1	35261	23.3
NSGAII-4000	1596	26.7	4874	29.5	9816	31.8	21395	40.2	36375	54.1
MOEA/D-500	1485	16.4	4438	17.1	8851	18.5	18941	20.3	32540	21.2
MOEA/D-1000	1494	33.6	4576	34.3	9256	36.5	19897	39.7	34842	42.4
MOEA/D-2000	1525	65.2	4703	69.5	9594	71.7	20723	78.6	36253	84.8
MOEA/D-4000	1512	**130.3**	4781	**135.2**	9778	**141.7**	21522	**156.9**	37687	**168.2**
DRL-MOA	1603	2.6	5150	4.5	10773	6.3	24567	9.4	44110	12.9

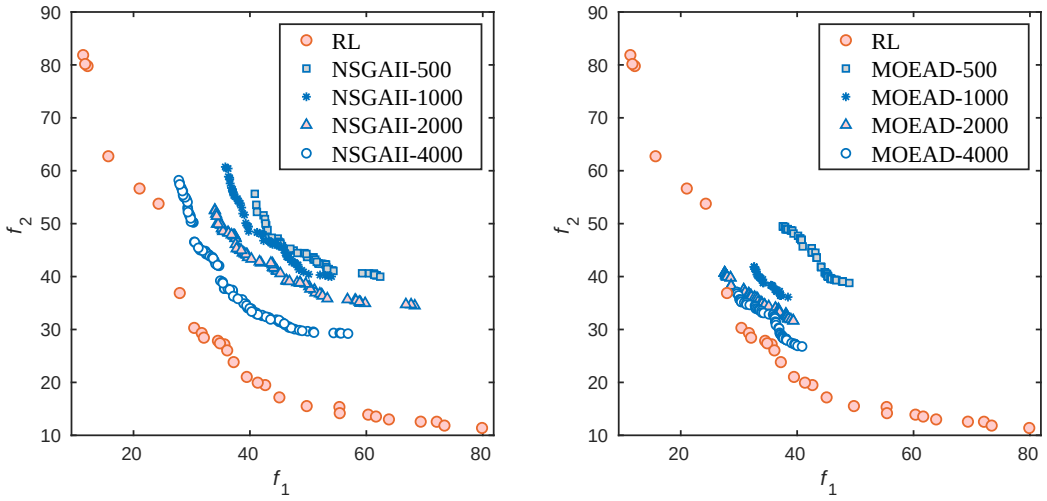
且 DRL-MOA 的运行时间远低于比较 MOEA。增加 MOEA/D 和 NSGA-II 的迭代次数可以提高它们的优化性能，但会导致大量的求解时间。MOEA/D 需要 150 多秒才能达到可接受的收敛水平，NSGA-II 进行 4000 次迭代大约 30 秒，但是 NSGA-II 的性能始终是对比算法中最差的。而本章所提方法总是可以在最短的求解时间内达到最好的优化水平，实现多目标旅行商问题的快速精准求解。

最后，对模型在 500 城市规模的测试问题上的性能进行测试。该模型仍然是在 40 城市规模的测试问题上进行训练，它用于求解 500 城市规模的测试问题，图 4.12 给出了该实验结果。可见，DRL-MOA 的优化性能和求解速度明显优于经典的多目标进化算法，而且该性能差距比在小规模问题上的差距更大。



(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.9　算法在 100 城市 KroAB100 问题上的帕累托前沿



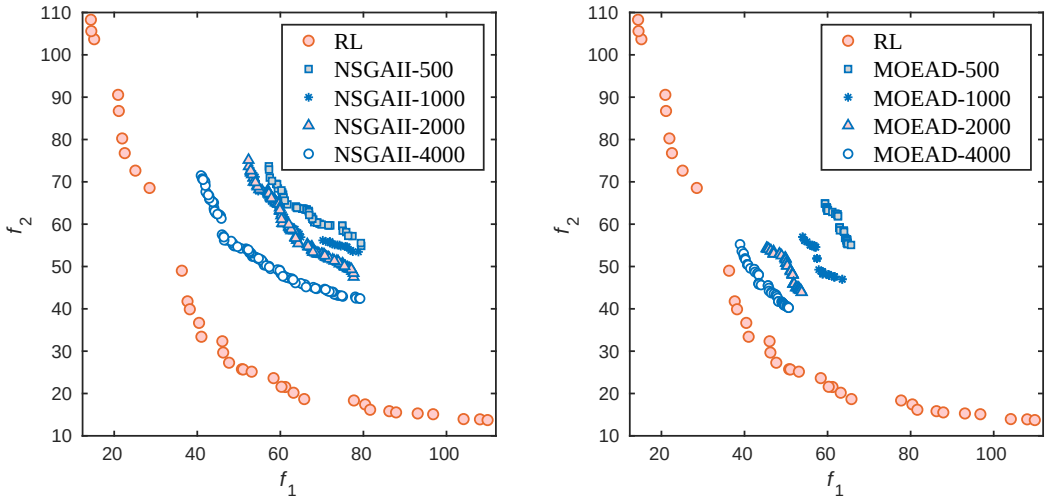
(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.10　算法在 150 城市 KroAB150 问题上的帕累托前沿

4.4.4　高维多目标测试问题实验结果

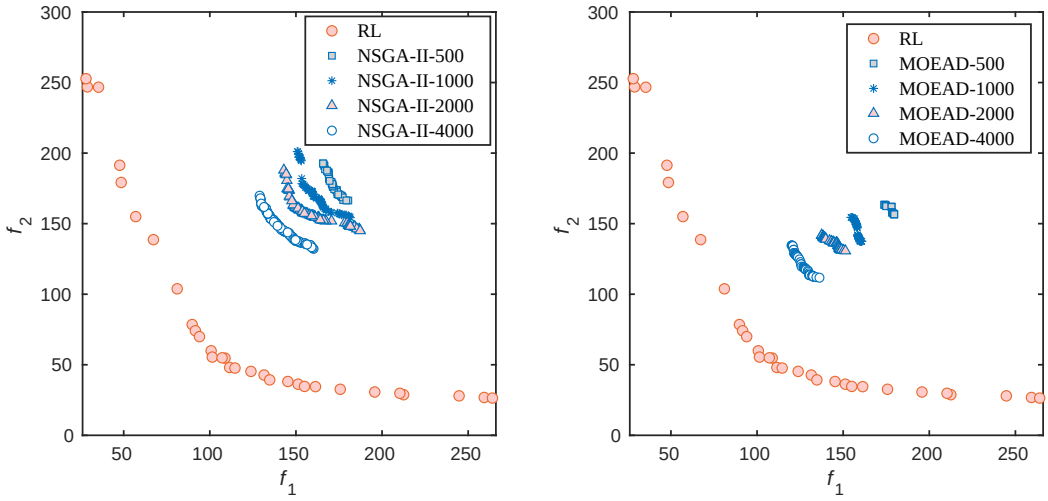
上述实验均为两目标旅行商问题的实验结果，本节进一步对本章所提方法在高维多目标问题上的性能进行测试，具体地，在三目标和五目标的旅行商问题上进行实验对比。模型仍然在 40 城市规模的测试问题上进行训练，并用于求解 100 个 200 城市规模的测试问题。三目标的测试问题构建方法为：类似于混合测试问题，组合两个二维输入和一个一维输入。五目标的测试问题构建方法为：类似于混合测试问题，组合三个二维输入和两个一维输入。

三目标和五目标的实验对比数据在表 4.4 中给出，三目标的实验结果在



(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.11　算法在 200 城市 KroAB200 问题上的帕累托前沿



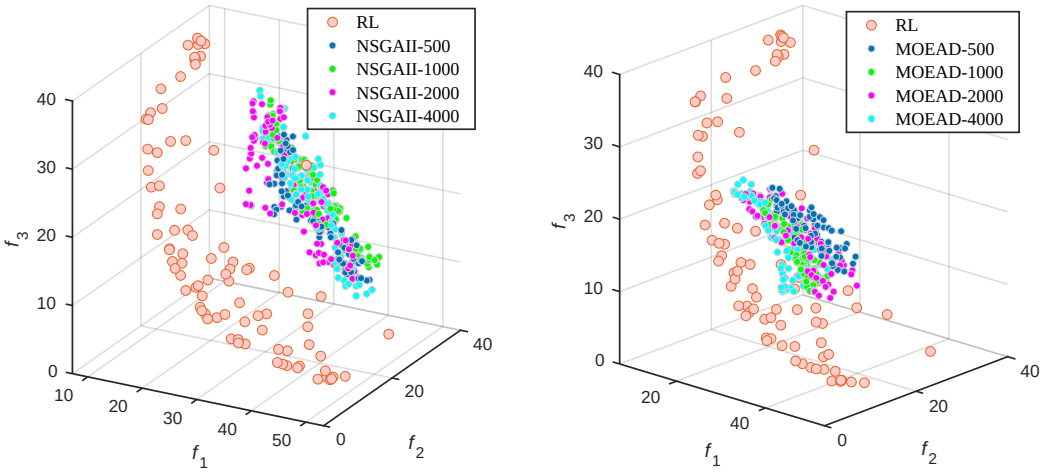
(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.12　算法在 500 城市欧几里得双目标 TSP 问题上的帕累托前沿

图 4.13 和图 4.14 中进行可视化。可以观察到，DRL-MOA 在所有 100 和 200 城市的 TSP 问题上都明显优于传统的多目标进化算法，并且 DRL-MOA 在 200 节点测试问题上的表现明显优于在 100 节点测试问题上的表现，可见本章所提方法在大规模、高维多目标组合优化问题上具有较好的优化性能。

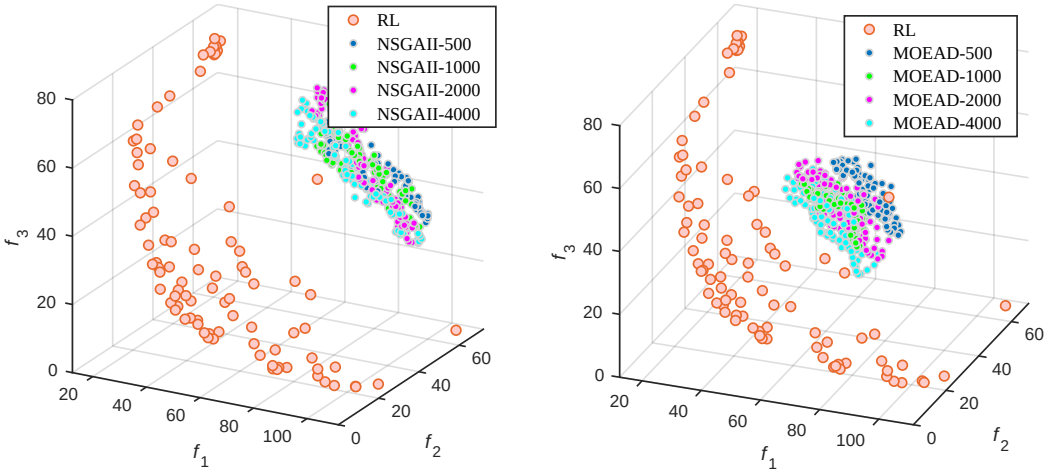
4.4.5　与启发式搜索方法的实验对比结果

本节将 DRL-MOA 与启发式局部搜索方法进行进一步的实验对比。上述多目标进化算法是求解多目标优化问题的通用算法，由于组合优化问题的特殊性质，



(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.13　算法在 100 城市三目标 TSP 问题上的帕累托前沿



(a) DRL-MOA 和 NSGA-II　　(b) DRL-MOA 和 MOEA/D

图 4.14　算法在 200 城市三目标 TSP 问题上的帕累托前沿

为提高求解组合优化问题的优化精度，文献中也存在一些专门为了求解多目标组合优化的启发式局部搜索算法。Ishibuchi 和 Murata 首次提出了一种用于专门求解多目标组合优化问题的的多目标遗传局部搜索算法（MOGLS）[139]，该方法是求解多目标组合优化问题的经典方法，Jaszkiewicz 等人[140] 进一步对 MOGLS 进行改进，专门用于求解多目标旅行商问题，其性能明显优于原始的 MOGLS 算法。因此，为了进一步评估本章所提方法的性能，本节将 DRL-MOA 与改进的 MOGLS 进行实验对比研究。

近年来局部搜索方法已得到广泛的研究，学者们根据专家经验设计了大量启发式算子来提升算法在特定问题上的求解精度，这些算法针对某一类特定的问题

表 4.4　算法在不同规模三目标和五目标 TSP 问题上的实验结果

	3-objective CSP		5-objective CSP	
	100-city	200-city	100-city	200 -city
NSGAII-500	7.63E + 05	5.09E + 06	5.00E + 09	1.31E + 11
NSGAII-1000	7.72E + 05	5.41E + 06	5.19E + 09	1.42E + 11
NSGAII-2000	8.25E + 05	5.56E + 06	5.51E + 09	1.57E + 11
NSGAII-4000	8.53E + 05	5.98E + 06	6.12E + 09	1.64E + 11
MOEA/D-500	7.74E + 05	5.63E + 06	5.55E + 09	1.53E + 11
MOEA/D-1000	8.17E + 05	6.02E + 06	6.19E + 09	1.55E + 11
MOEA/D-2000	8.58E + 05	6.26E + 06	6.39E + 09	1.68E + 11
MOEA/D-4000	8.82E + 05	6.49E + 06	7.15E + 09	1.78E + 11
DRL-MOA	1.13E + 06	9.42E + 06	1.19E + 10	4.06E + 11

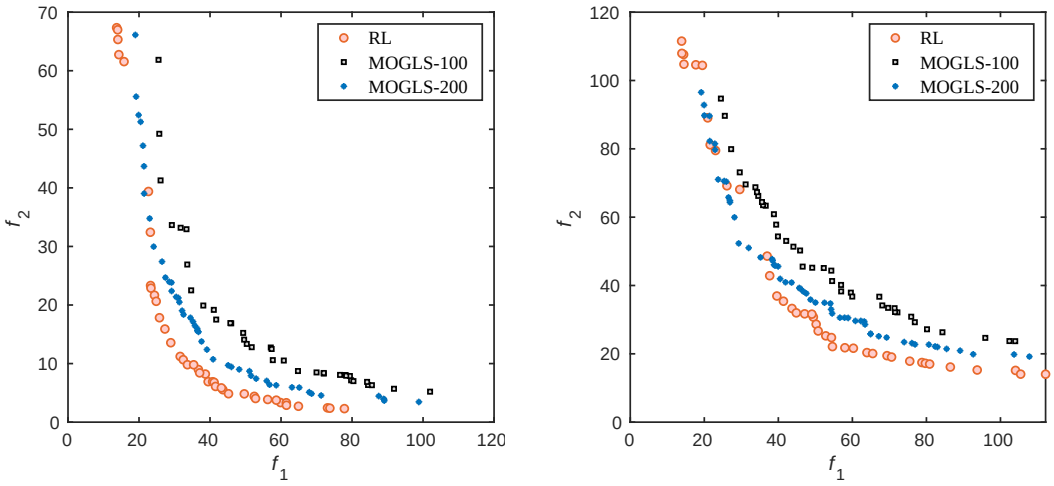
进行了大量的算法设计和调试过程，因此能够有效提高算法性能。而本章所提方法的主要优势是其求解速度和泛化能力，因此没有与大量此类局部搜索方法进行广泛的实验对比，只和代表性的 MOGLS 算法进行对比。

本节在 Python 平台严格按照文献 [140] 对改进 MOGLS 算法进行实现[4]。算法参数（包括临时种群大小、初始解数量）与文献 [140] 中的设置一致，算法中的局部搜索使用标准的 2-opt 算法，如果局部搜索达到了预先指定的迭代次数 N_{LS}，则算法终止，N_{LS} 用于控制求解时间和模型性能的平衡 [139]。本节分别对参数为 $N_{LS} = 100, 200, 300$ 的 MOGLS 算法进行实验。

第三章指出通过局部搜索可以进一步对深度神经网络模型输出的解进行提升[105]，因此实验中对 DRL-MOA 输出的解也进行一次 2-opt 局部搜索以提升解的质量，即 DRL-MOA+LS，由于只需要执行一次 2-opt 局部搜索，因此只会增加数秒的求解时间，而不像 MOGLS 需要进行大量 2-opt 过程。表 4.5 给出了 MOGLS-100、MOGLS-200、MOGLS-300、DRL-MOA 和 DRL-MOA+LS 的实验对比结果，同时图 4.15 中可视化了 MOGLS-100、MOGLS-200 和 DRL-MOA 方法的帕累托前沿。

可见，DRL-MOA+LS 在所有测试问题上都优于改进 MOGLS 算法，同时需要更少的求解时间。同样地，即使改进 MOGLS 算法运行了 1000 秒，DRL-MOA 也可以在所有的 200 节点测试问题上取得比改进 MOGLS 算法更好的优化性能，而 DRL-MOA 只需要约 13 秒。尽管 MOGLS 在 100 节点测试问题上的表现略好于 DRL-MOA，但 DRL-MOA 始终可以在 7 秒内获得优化精度差距不大的结果。相对于传统的启发式局部搜索方法，本章所提 DRL-MOA 方法结合局部搜索可以在更少的求解时间内获得更好的优化效果，证明了本章所提方法的有效性。

[4]https://github.com/kevin031060/Genetic_Local_Search_TSP



(a) 混合测试问题帕累托前沿　　(b) 欧几里得测试问题帕累托前沿

图 4.15　算法在 200 城市双目标 TSP 问题上的帕累托前沿

表 4.5　算法在不同类型多目标旅行商问题上的实验对比结果

		100-city		200-city	
		HV	Time/s	HV	Time/s
混合问题	MOGLS-100	1848	143.3	5800	394.6
	MOGLS-200	1986	254.9	6516	738.8
	MOGLS-300	2037	**349.8**	6794	**1073.7**
	DRL-MOA	2022	6.6	6911	12.9
	DRL-MOA+LS	2073	12.7	6988	23.2
		HV	Time/s	HV	Time/s
欧几里得问题	MOGLS-100	3127	138.1	13799	374.8
	MOGLS-200	3362	246.3	15093	697.1
	MOGLS-300	3450	**348.1**	15716	**1005.6**
	DRL-MOA	3342	6.4	15750	12.9
	DRL-MOA+LS	3474	12.4	16117	24.1

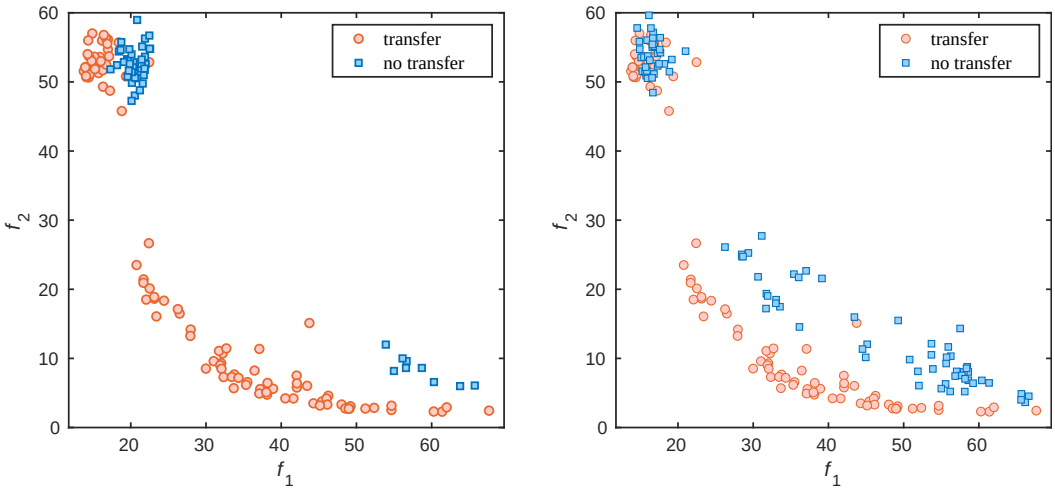
4.4.6　参数迁移策略有效性验证

本节对基于邻域的参数迁移策略的有效性进行实验验证。首先，对使用和不使用参数迁移策略的模型训练情况进行对比，该两种情况下，都采用包含 120,000 个 20 节点 MOTSP 问题的训练集进行 5 回合的训练。图 4.16 (a) 可视化了这两种情况下模型输出的帕累托前沿。

通过结果可以明显的发现，如果不使用参数迁移策略进行训练，模型的性能

会非常差，解没有收敛并且解的多样性很差，这是因为如果不进行参数迁移，从头开始对每个子问题进行训练，各个子问题的模型几乎无法收敛。

此外，本节进一步将训练集的规模扩大到了 240,000，不采用参数迁移策略进行 10 回合的训练，即模型的训练时间是以前的四倍。该结果展示在图 4.16 (b) 中。可以看出，在没有参数迁移策略的情况下，即使模型训练规模增加了 4 倍，它仍然表现出较差的性能。因此将参数迁移策略应用于训练是有效且关键的，否则不可能在可接受的训练时间内得到可用的模型。



(a) 进行相同规模的训练　　(b) 无参数迁移模型的训练规模增加 4 倍

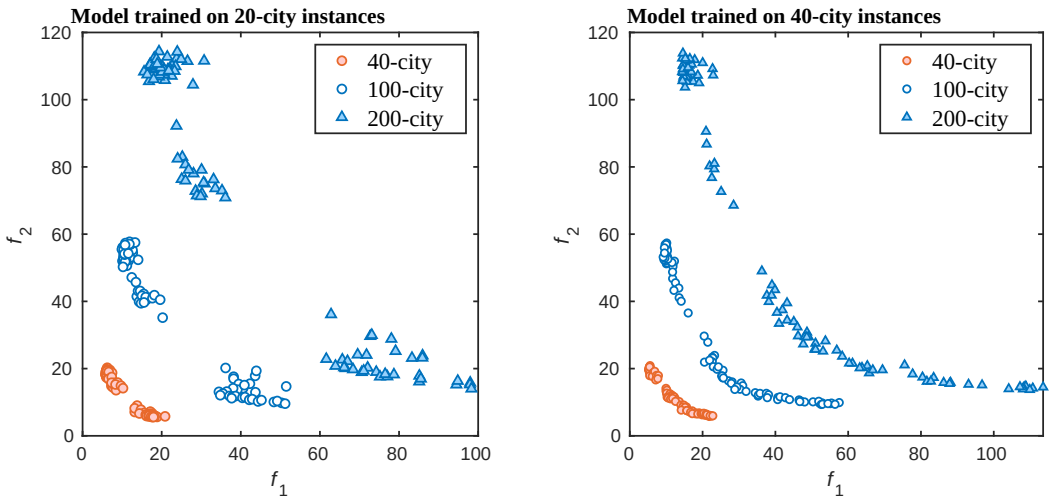
图 4.16　采用和不采用参数迁移策略训练得到的模型的表现

4.4.7　训练集规模分析

上述模型均在 40 城市规模的 MOTSP 问题上进行训练。本节对训练集中节点规模对模型训练效果的影响进行实验分析。图 4.17 (a) 和 (b) 分别给出了 20 和 40 节点规模的训练集得到的模型的性能。

可见，在 20 城市规模训练集上得到的模型明显比在 40 城市规模的训练集上得到的模型表现更差。前者的解多集中在少量几个区域，非支配解较少。这主要是由于下列原因导致，在 40 城市规模数据集上进行训练时，训练每个样本的过程中进行了 40 个城市选择的决策，是在 20 城市规模训练集上训练的两倍，因此，如果两个模型都使用 $120,000$ 个样本进行训练，40 城市规模训练集上进行了 $120,000 \times 40$ 次决策，是前者的两倍，因此在 40 城市规模训练集上得到的模型更优。鉴于此，可以通过增加训练集的规模或者提高训练次数来提高性能。

综上，与经典的多目标优化方法相比，DRL-MOA 展现出一些新的特性。首先，该方法通过离线训练、在线应用的方式，无需迭代搜索，可以直接输出问题



(a) 20 城市规模训练集的模型　　(b) 40 城市规模训练集的模型

图 4.17　20、40 规模训练集得到的模型求解 40、100、200 规模问题

的帕累托前沿，有效提高了求解速度，从而可以实现多目标优化问题的实时在线求解；其次，该方法展现了很强的泛化能力，一旦模型训练完成，就可以用于求解任意该类型的新问题，不需要重新训练。通过在 2、3、5 个目标、高达 500 个城市规模问题上的实验评估，验证了该方法在高维多目标、大规模组合优化问题上的明显优势，在解的收敛性和多样性方面明显优于传统多目标优化方法，实现了多目标组合优化问题“稳、快、准”的求解。

4.5　本章小结

本章对基于深度强化学习的多目标组合优化方法进行研究，在国际上首次提出了一种基于深度强化学习的多目标组合优化方法，该方法可以直接输出多目标优化问题的帕累托最优解集，计算耗时低，而传统多目标组合优化方法如果不使用大量迭代搜索和 / 或大种群规模的解搜索过程，很难在短时间内对大规模问题进行求解[124–126]，通过在高达 500 城市规模、多达五个目标的多目标旅行商问题上进行的实验测试，验证了本章所提方法的有效性，在求解时间和优化能力上优于传统的多目标进化算法和局部搜索算法，采用端到端的方式输出帕累托最优解，无需迭代搜索，从而可以实现多目标组合优化问题的快速求解，并且在优化能力和泛化能力上具有显著优势。

第五章　基于深度强化学习的分支定界组合优化方法

求解组合优化问题的方法主要包括：1）能够获得问题近似解的近似方法，包含启发式方法以及近似算法；2）以及能够获得问题理论最优解的精确算法，该两类方法在不同领域的应用场景具有不同的应用需求，前两章所提方法都是对单目标和多目标问题进行近似求解，而分支定界法是组合优化另外一类重要的方法，该方法能够保证得到问题的最优解，是当前 CPLEX 等主流优化求解器的核心算法，在大量场景下存在应用需求，并且分支定界法不需要针对问题类型进行专门的设计，只需要将组合优化问题建模为混合整数规划形式，便可以对任意组合优化问题进行求解，因此是当前求解通用组合优化问题优化软件的核心方法，但是该类方法面临求解速度极慢的问题。本章进一步采用深度学习方法对分支定界法进行研究，以提高其优化速度，提出基于深度强化学习的分支定界组合优化方法，从而形成系统、全面的基于深度强化学习的组合优化解决方案。

5.1　分支定界法

分支定界法由于其通用性和优化能力，是当前 SCIP、CPLEX（IBM ILOG CPLEX）、Gurobi 和 Xpress（FICO Xpress）等知名优化求解器的核心算法，只需要将组合优化问题建模为混合整数线性规划形式，便可以采用分支定界法对任意组合优化问题进行求解，分支定界法可以确保得到组合优化问题的最优解，或者在受限运算时间内提供现有解和最优解对应的目标函数值的差距（optimality gap），这在理论上和实践中都为决策提供了有价值的信息，具有广泛的应用场景。

在实践中，大多数组合优化问题可以表述为混合整数线性规划形式 (Mixed-integer linear programs, MILPs)，在这种情况下，可以采用分支定界 (B&B) [141] 对该 MILP 进行精确求解。分支定界递归地将解空间划分为搜索树，并不断计算当前解的最优性边界（optimality bound）以修剪不可能包含最优解的子树，这个迭代过程需要对以下内容进行决策：1）节点选择：选择下一个要进行处理的节点；2）变量选择：选择用于划分节点搜索空间的决策变量[142]。

分支定界算法运行过程中，选择一个合适的节点或者变量能够大幅降低解搜索的空间和时间，多年来，上述节点选择、变量选择的决策过程主要通过专家设计的启发式方法进行实现的，通过对不同策略进行试验设计，从而找到一个可靠的启发式[143]。然而，当前节点选择变量选择的启发式规则在运算量和效果方面仍然存在矛盾，例如通过计算每个节点的 Strong Branching (SB) 分数是进行变量选择一个经典的方法，相对于当前的混合启发式变量选择方法，SB 方法能够实现 65% 的搜索树规模的缩减，但是会导致 44% 的运算时间的增加。现有的分支定界

方法仍然面临着求解时间长的问题，如何进行有效的节点选择、变量选择的决策仍然是几十年来悬而未决的课题，采用人工设计的方式对选择决策进行启发式规则设计存在很大局限性，需要大量的专家经验和试错成本，近年来随着人工智能技术的发展，深度学习模型能够有效提前问题的特征，并做出相应决策，从而替代人力的启发式规则设计过程，为提高分支定界法的效率提供了可能性。本章对分支定界法进行介绍，对其中可采用人工智能技术改进的决策问题进行分析。

5.1.1 混合整数线性规划建模

采用分支定界法对组合优化问题求解，首先需要将组合优化问题建模为混合整数线性规划 (MILP)，其形式为:

$$\begin{aligned} \min \quad & \mathbf{c}^\top \mathbf{x} \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{l} \leq \mathbf{x} \leq \mathbf{u} \\ & x_i \in \mathbb{Z}, \quad i \in \mathcal{I} \\ & \mathbf{c} \in \mathbb{R}^n, \mathbf{A} \in \mathbb{R}^{m \times n}, \mathbf{b} \in \mathbb{R}^m, \mathbf{l}, \mathbf{u} \in \mathbb{R}^n, \end{aligned} \tag{5.1}$$

其中问题的目标是找到一组最优的决策变量 $\mathbf{x}$ 以最小化目标函数，其中 $\mathbf{c}$ 作为目标系数向量。该 MILP 有 m 个约束和 n 个决策变量，部分决策变量是整数，$\mathcal{I} \subseteq \{1, \ldots, n\}$ 是整数决策变量的索引，$\mathbf{A}, \mathbf{b}$ 是约束的系数矩阵和约束的右侧向量，$\mathbf{l}, \mathbf{u}$ 是决策变量的上下界约束。

通过去掉所有的整数约束，可以将 MILP 松弛为一个线性规划问题（Linear program, LP），在最小化问题中，由于去掉了一些约束，求解该线性规划问题能够获得原 MILP 问题的下界，该 MILP 问题所有可行解的目标值均不可能优于该下界。

5.1.2 分支定界法流程

分支定界法从求解原始 MILP 问题的线性松弛开始，即忽略所有的整数约束对问题进行求解，求解得到的提供了问题的下界。如果 $\mathbf{x}^*$ 满足原 MILP 所有的整数约束，则该解就是原 MILP 问题的最优解，算法终止。通常得到的解一般不满足所有的整数约束，在该情况下，在不满足整数约束的决策变量集合 $\mathcal{C} = \{i \mid x_i^\star \notin \mathbb{Z}, i \in \mathcal{I}\}$ 中选择一个变量 i，在该变量 i 上进行分支，将以下两个约束分别添加到上一步的 LP 问题，得到两个子问题。

$$x_i \leq \lfloor x_i^\star \rfloor, x_i \geq \lceil x_i^\star \rceil, \quad \exists i \in \mathcal{I} \mid x_i^\star \notin \mathbb{Z}, \tag{5.2}$$

其中 $\lfloor x_i^\star \rfloor$ 是指小于 $x_i^\star$ 的最大整数值，$\lceil x_i^\star \rceil$ 是大于 $x_i^\star$ 的最小整数值，这里 i 被称为 分支变量。

通过在决策变量 i 上进行分支，可以构建得到两个新的 LP 子问题，它们代表分支定界法搜索树的叶子节点。因此，下一步是选择一个叶子节点，并重复上述步骤（对 LP 问题进行求解，寻找不满足整数约束的决策变量，进行分支），选择叶子节点的决策过程即“节点选择”问题，选择不满足整数约束的整型变量进行分支的决策过程即“变量选择”问题。

该过程不断迭代进行，一旦找到满足原 MILP 问题所有整数约束的一个可行解 $\hat{x}$，它就提供了该 MILP 问题的上界，如果找到的可行解的目标值优于当前的上界（算法运行时初始上界为 $+\infty$），则更新上界；另一方面，当上界更新后，如果某个叶子节点对应的 LP 子问题的解的目标值比当前上界更差，则该叶子节点将被剪枝、不再被继续处理，因此通过该剪枝过程，可以极大缩小解搜索的空间。

如果解搜索过程中满足以下条件，则叶子节点停止分支：1）叶子节点的 LP 子问题无解；2）叶子节点的 LP 问题找到了可行解，更新上界；3）叶子节点的 LP 子问题的解的目标值比当前上界差。当所有的叶子节点都停止分支，即不存在待解决的子问题，返回具有最优目标值的可行解，即为该混合整数线性规划问题的最优解。

5.1.3 分支规则

节点选择问题是提高分支定界法求解效率主要研究的问题，即在候选变量集合 $\mathcal{C} = \{i \mid x_i^\star \notin \mathbb{Z}, i \in \mathcal{I}\}$ 中选择一个不满足整数约束的整型变量 i 进行分支。现有的方法通常对候选变量集合 $\mathcal{C}$ 中的每个变量进行打分，从而选择分数最大的决策变量进行分支，当前最常用的打分准则是，对变量进行分支之后该节点对应的子问题下界的变化，基于该准则，多年来学者们设计了多种分支规则能够提高分支定界法的效率。

其中，Strong branching（SB）规则[144] 能够生成最小规模的分支定界搜索树，该规则显式地对子问题上下界的变化进行衡量，从而选择最佳的分支变量。其计算方法如下，对于当前节点 N 对应的 LP 子问题，其 LP 解为 $\mathbf{x}^*$，对应的 LP 目标函数值为 z^*，从候选集合 $\mathcal{C}$ 中选择变量 i 进行分支，可以得到两个 LP 子问题 N_i^- 和 N_i^+，对应的各自的 LP 目标函数值为 z_i^{*-} 和 z_i^{*+}，如果 LP 子问题 N_i^- 和 N_i^+ 无可行解，则 z_i^{*-} 和 z_i^{*+} 设置为一个极大值。从而可以计算得到对该节点进行分支后，目标函数值的变化值 δ_i^- 和 δ_i^+，SB 分数通过如下公式进行计算：

$$SB_i = \text{score}\left(\max\left\{\delta_i^-, \epsilon\right\}, \max\left\{\delta_i^+, \epsilon\right\}\right) \tag{5.3}$$

其中 ϵ 是一个极小值，通常采用乘法函数进行分数计算，即 $\text{score}(a,b) = a \times b$，SB 分支规则对候选集合 $\mathcal{C}$ 中所有变量计算 SB 分数，选择 SB 分数最大的变量进行分支，由于计算每个 SB 分数需要求解两个 LP 问题，该分支规则虽然能够很大限度地缩小搜索树的规模，但是每次分支决策需要耗费大量的运算时间，因此算法的总求解时间仍然很长。

鉴于 SB 方法分支决策的长耗时，计算 Pseudocost （PC）分数替代 SB 分数是当前优化求解器常用的方法，PC 分数将之前分支过程中变量 i 的目标值的平均变化作为下一步分支过程中 i 目标值变化的估计，因此不需要对实际分支之后的两个 LP 子问题进行求解，可以很大的缩短计算时间。具体地，令历史分支过程中变量 i 的目标值的平均变化为 Ψ_i^- 和 Ψ_i^+，PC 分数计算如下：

$$PC_j = \text{score}\left(\left(x_i^* - \lfloor x_i^* \rfloor\right)\Psi_i^-, \left(\lceil x_i^* \rceil - x_i^*\right)\Psi_i^+\right) \tag{5.4}$$

其中 $x_i^* - \lfloor x_i^* \rfloor$ 和 $x_i^* - \lceil x_i^* \rceil$ 代表该整型变量在该节点 LP 问题解中的小数部分，score 函数的计算方法与 SB 分数相同，利用该计算方法可以有效降低运算时间，但是产生的搜索树的规模大于采用 SB 规则产生的搜索树规模。由于 PC 方法在分支定界过程的初始阶段没有足够的历史数据积累，因此 PC 分数计算的不准确，导致在分支的开始阶段算法效果很差，鉴于此，Reliability branching 方法在开始阶段采用 SB 规则进行分支，直到积累足够多的历史数据使得 PC 分数能够可靠地进行分支，再采用 PC 规则进行分支，在 Hybrid branching[145] 方法中，PC 分数和其他分数结合进行分支决策。

根据以上内容可以发现，传统变量选择策略在分支效果和分支规则运算时间上存在矛盾，随着人工智能技术的发展，为了解决该问题，一种思路是通过机器学习方法对 SB 分数进行估计，从而在保持分支效果的同时降低运算时间，另一种思路是通过机器学习方法探索新的分支规则，从而缩短算法求解时间。不论哪种思路都需要对分支策略采用人工智能的方法进行建模，模型需要感知当前求解器的状态，然后做出变量选择的决策。

5.2 基于图指针神经网络的分支策略建模

本章提出了一种图指针神经网络模型对分支策略进行建模，并结合求解器的图特征，设计了全局特征和历史特征对求解器的状态进行更加有效的描述，在此基础上，图指针网络模型以图结构特征、全局特征和历史特征作为输入，输出各个分支变量选择的概率值，从而指导分支变量的选取。

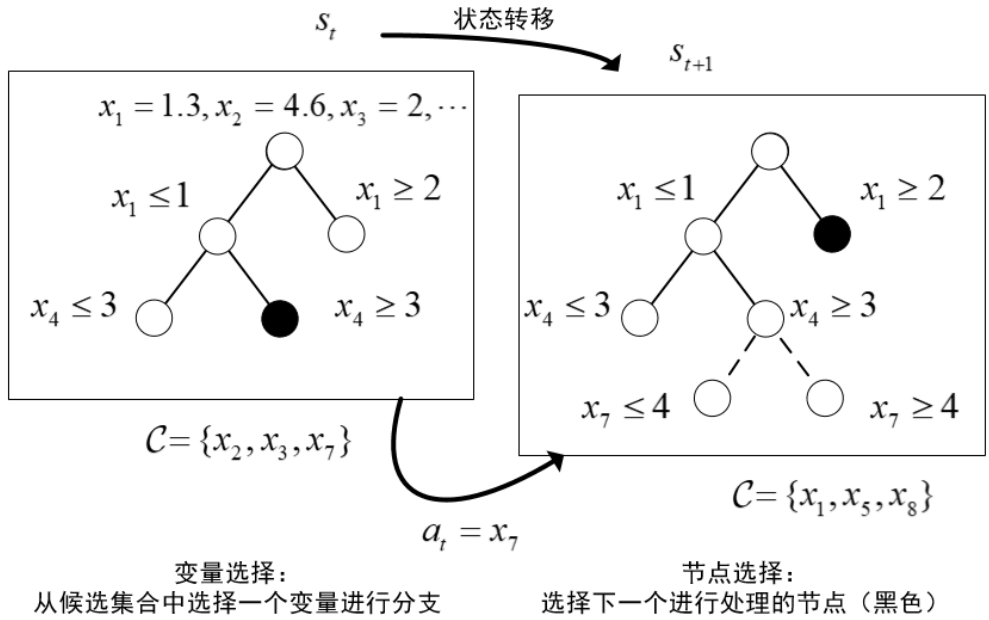


图 5.1　分支定界法示意图

5.2.1　马尔可夫决策过程建模

首先，分支定界法中分支变量的选择过程可以建模为马尔可夫决策过程[146]，其示意图如图 5.1 所示。

在第 t 步决策中，求解器处于 $\mathbf{s}_t$ 状态，该状态包含当前搜索树的状态、所有历史的分支决策、迄今为止找到的最优整数解、每个节点的 LP 解、当前的叶子节点、以及求解器其他的统计信息（例如每个启发式规则的调用情况等)。基于求解器当前的状态 $\mathbf{s}_t$ ，智能体根据策略 $\pi(\mathbf{a}_t \,|\, \mathbf{s}_t)$ 从候选集合 $\mathcal{C} = \{i \mid x_i^\star \notin \mathbb{Z}, i \in \mathcal{I}\}$ 中选择一个不满足整数约束的整型变量 i 进行分支，即当前智能体的动作 $a_t = i$。

求解器对分支之后的两个 LP 子问题进行求解，更新上下界并对搜索树进行剪枝，然后选择下一个要分支的叶子节点，此时当前环境转换至一个新的状态 $\mathbf{s}_{t+1}$，并且再次调用分支策略 $\pi(\mathbf{a}_{t+1} \,|\, \mathbf{s}_{t+1})$ 以进行下一个分支决策，这个过程如图 5.1所示。

作为马尔可夫决策过程，分支定界法的分支决策过程是回合制的，每一个回合即求解一个 MILP 问题实例，初始状态对应该 MILP 问题搜索树的根节点，最终状态为优化过程的结束，即所有叶子节点都无法被继续分支。因此采用分支策略 π，该马尔科夫决策过程可以建模为:

$$p_\pi(\tau) = p(\mathbf{s}_0) \prod_{t=0}^{T-1} \sum_{\mathbf{a} \in \mathcal{A}(\mathbf{s}_t)} \pi(\mathbf{a} \,|\, \mathbf{s}_t) p(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t, \mathbf{a}).$$

对分支策略 π 进行学习主要通过以下步骤进行：1）对问题状态 s_t 进行定义，在分支决策的每一步中，都需要根据当前问题状态进行分支决策，由于分支定界过程中没有标准化的环境状态定义，因此首先需要对问题状态进行定义，尽可能多地提取问题在当前状态下的特征，同时兼顾运算时间，从而进行更好的分支决策；2）对分支策略 π 进行建模，模型需要能够实现问题状态 s_t 到分支“动作”a_t 的准确映射，例如神经网络、随机森林、支持向量机等模型，需要根据分支定界问题的特征对模型进行设计，从而提高模型性能；3）最后采用高效的训练算法对策略模型进行学习，可以通过多种机器学习方法对策略 π 进行学习，以最小化搜索树的规模、减少算法求解时间，常用的策略学习方法是强化学习，它以该分支定界问题的目标作为奖励函数，通过大量实例的求解过程对策略进行反馈调整，从而达到提高分支定界法效率的目的。

因此，后文按照上述三个步骤对本章所提出的基于深度强化学习的分支定界组合优化方法进行介绍。

5.2.2 状态定义

首先对分支定界算法在第 t 步决策中的环境状态 s_t 进行定义，s_t 由变量特征、约束特征、边特征、全局特征、历史特征构成，即 $s_t = (V, C, E, G, H)$，s_t 由当前求解器状态的二部图特征定义，如图 5.2 所示，求解器的当前状态即为当前的 LP 线性规划形式，即由 m 条约束构成的图，图的左侧是每条约束中的变量，图的右侧是每条约束的右边值，图的边 $(i, j) \in E$ 是变量 i 和约束 j 之间的连接状态，即第 j 条约束中是否包含变量 i，边的权重即为连接约束 j 和变量 i 的约束系数，

根据上述代表求解器当前状态的二部图结构，本章对构成当前状态的变量特征、约束特征、边特征、全局特征、历史特征设计如下：

（1）变量特征代表在第 t 步分支过程中，候选变量的当前特征属性，例如变量类型、变量系数、变量的当前值、变量的当前值是否在边界上，变量当前值的小数部分等等，特征类型包括 0-1 值、实数值等，用于表征候选变量的当前状态，共有 n 个候选变量，特征维度为 d，因此变量特征的维度为 $n \times d$，变量特征的详细介绍见表 5.1 。

（2）约束特征代表在第 t 步分支过程中，当前 LP 问题约束的特征属性，例如该约束的右边值、约束左边值是否恰好达到边界、约束系数与目标系数相似性等，属性类型包括 0-1 值、实数值等，当前 LP 问题共有 m 个约束，特征维度为 c，因此约束特征的维度为 $m \times c$，约束特征的详细介绍见表 5.2 。

（3）边特征是在每条约束中，每个变量在该条约束中的系数，因此共有 $m \times n$ 条边，特征维度为 1，即系数值，如果约束中不包含某个变量，则系数值为 0。

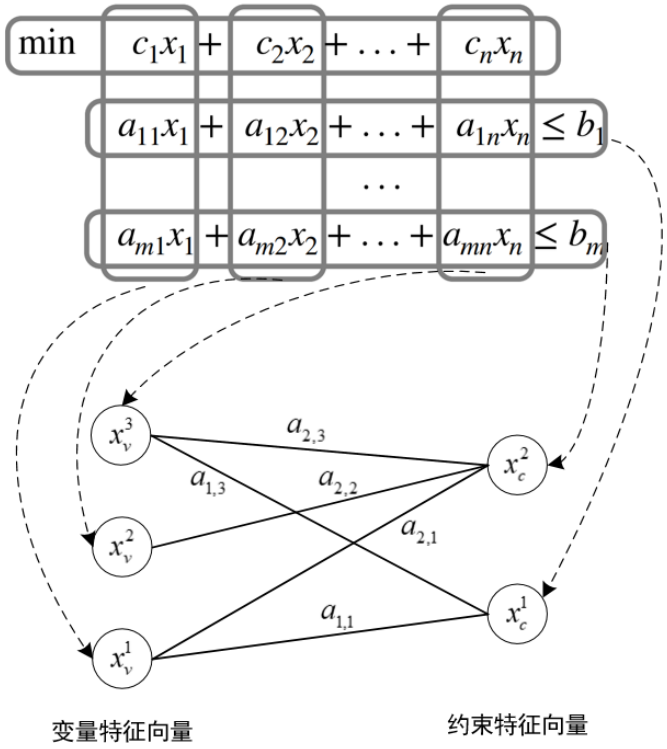


图 5.2　分支定界求解器状态的图结构

（4）全局特征是求解器当前的全局状态，例如问题当前的最优性差距、当前节点的目标值与上下界之间的差距、当前搜索树的深度、当前节点的深度等，特征列表见表 5.3 ，全局特征是基于 PySCIPOpt 的 API 接口进行提取，代表当前阶段 MILP 整体特征的是当前阶段 MILP 上下界的差距、可行解 / 不可行解的数量等信息，代表当前 LP 子问题节点特征的是当前节点的深度、当前节点的 LP 目标值信息等。

当前节点的深度和当前 MILP 上下界的差距直接通过调用 PySCIPOpt 接口得到，可行解节点数和不可行解数量由产生可行解以及不可行解的叶子节点的比例计算，按照如下公式计算：

$$P_{feasible} = \frac{N_{feasible}}{\max(N_{leaves}, 0.1)} \tag{5.5}$$

当前节点的 LP 目标函数值与全局上界和全局下界之间差距 gap 由以下公式计算，其中当前节点的 LP 目标值和全局上下界均由 PySCIPOpt 接口得到：

$$\text{gap}(x,y) = \begin{cases} 0 & \text{, if } xy < 0 \\ \frac{|x-y|}{\max\{|x|,|y|,1\times10^{-10}\}} & \text{, else} \end{cases} \tag{5.6}$$

当前节点的 LP 目标函数值与全局上界和全局下界之间的相对位置关系 pos 由以下公式计算：

$$\mathrm{relPos}(z,x,y)=\frac{|x-z|}{|x-y|}. \tag{5.7}$$

(5) 历史特征由两部分构成，第一部分为：从初始状态开始到上一步分支，之前所有步 $1\cdots t-1$ 选择的分支变量的集合 $\mathcal{C}_1=a_1\cdots a_{t-1}$；第二部分为：生成当前节点的过程中，变量值发生改变的变量集合 $\mathcal{C}_1=a_1\cdots a_{t-1}$，即通过增加一个整型约束后，新问题的解中变量值发生改变的所有变量。

传统方法只考虑了变量特征、约束特征和边特征，本章进一步对全局特征、历史特征进行了提取，从而能够获得更加丰富的环境状态，为了在当前节点选择下一步进行分支的变量，考虑当前节点的状态和当前搜索树的全局状态是十分有意义的，同时，观察当前节点生成的程中值发生改变的变量、观察历史分支过程中选择的变量，也可以为分支变量的选择提供有效信息，因此本章对全局特征和历史特征进行了设计，从而使得提取的特征能够更加充分地描述当前问题的状态。

表 5.1　分支定界过程中的变量特征

特征类型	特征描述
二进制	描述变量的类型，分别代表该变量是否为 01 变量、整数变量、连续变量
实数	该变量在 MILP 目标函数上的系数，并根据其约束系数的欧式范数进行归一化
二进制	该变量是否有上界 / 下界
二进制	该变量是否位于其上界 / 下界
实数	该变量值的小数部分
二进制	该变量的基状态，分别代表该变量是否在其上界、下界、上下界中间
实数	该变量的 reduce cost 值，表示该变量的值由 0 变为非 0 而要求目标系数进行改变的量
实数	在单纯形过程中，该变量自上次为 basic 到现在经历的迭代次数，根据总迭代次数进行归一化
实数	该变量在当前 LP 解中的变量值
实数	该变量在当前最好的可行解中的变量值
实数	该变量在所有可行解中的变量值的平均值

表 5.2　分支定界过程中的约束特征

特征类型	特征描述
实数	该约束的左边系数值与目标函数的系数值的相似度，用 cos 相似度计算
实数	该约束的右边值，并根据其左边系数的欧式范数进行归一化
二进制	该约束的对偶变量的值，并根据约束系数和目标系数的欧式范数的乘积进行归一化
二进制	该约束是否位于其边界

表 5.3　分支定界过程中的全局特征

特征类型	特征描述
实数	当前节点深度
实数	当前可行解数量，并根据总节点数进行归一化
实数	当前不可行解数量，并根据总节点数进行归一化
实数	当前上下界差距
实数	当前节点 LP 目标值与全局上界的差距
实数	当前节点 LP 目标值与全局下界的差距
实数	当前节点 LP 目标值与全局上下界的相对位置
实数	当前节点 LP 目标值与根节点上界的差距
实数	当前上界与根节点上界的差距

5.2.3　分支策略建模

由上节提取的问题特征可以看出，问题的状态具有二部图结构，即左侧节点(变量）与右侧节点（约束）通过边相连，因此可以通过图神经网络模型对该状态进行处理，首先对状态的图结构进行建模，过程如图 5.2 所示，即提取当前状态下变量的特征、约束的特征，变量节点与约束节点以约束系数为权重进行相连。

下一步，将变量选择的分支策略 $\pi_\theta(\mathbf{a}|\mathbf{s}_t)$ 参数化为图神经网络，图神经网络技术能够有效处理图结构的信息，已成功应用于各种具有图形结构输入的机器学习任务，例如分子特性预测、社交网络和引文网络中的各类任务等。

由于分支定界法的图结构状态信息具有稀疏性，即变量和约束之间的连接关系具有稀疏性，而图神经网络模型的计算复杂度只与图的密度相关，而与图的规模无关，因此图神经网络模型适用于处理该具有稀疏性质的分支定界状态图结构，并且节点的顺序不会影响图神经网络的输出，因此本章采用图神经网络对分支策略进行建模。

同时，对于分支定界法的变量选择过程，候选变量集合的规模会很大，单纯采用图神经网络进行节点概率的输出随着候选变量集合规模的变大准确率会明显降低，同时图神经网络无法对全局特征、历史特征进行处理，因此为了提高模型的表达能力，使模型能够更加充分地理解在分支定界法过程中当前的 MILP 状态，本章提出了一种图指针神经网络（Graph Pointer Neural Network, GPN）对分支策略进行建模，将全局特征和历史特征作为“查询 *query*”，与每个变量通过图神经网络计算得到的节点特征进行注意力计算，从而得到变量选择的概率，因此该模型包括两部分：1）图神经网络计算节点的特征向量；2）注意力机制输出节点选择概率，其中图神经网络处理的是 MILP 的节点特征、边特征和约束特征，注意力机制处理的是 MILP 的全局特征和历史特征。

基于图指针神经网络的分支策略建模方法论述如下。

（1）初始嵌入计算

由于变量特征、约束特征、边特征、全局特征具有不同的维度，比如变量特征为 13 维，全局特征为 9 维，因此，首先对变量特征 $\mathbf{x}_v$、约束特征 $\mathbf{x}_c$、边特征 $\mathbf{x}_e$、全局特征 $\mathbf{x}_g$ 进行同维度映射：

$$\begin{aligned}
\mathbf{x}_v &\leftarrow \mathrm{EMBEDDING}\left(\mathbf{x}_v\right) \\
\mathbf{x}_c &\leftarrow \mathrm{EMBEDDING}\left(\mathbf{x}_c\right) \\
\mathbf{x}_e &\leftarrow \mathrm{EMBEDDING}\left(\mathbf{x}_e\right) \\
\mathbf{x}_g &\leftarrow \mathrm{EMBEDDING}\left(\mathbf{x}_g\right)
\end{aligned} \tag{5.8}$$

其中 EMBEDDING$(\cdot)$ 由一个双层全连接神经网络构成，其隐含层和输出层的维度均为 d_h，区别只是输入层的维度不同，层间的激活函数为 LeakyRELU 函数：

$$\mathrm{LeakyRELU}\left(x\right)=\begin{cases} x, & \text{if } x \geq 0 \\ 10^{-2} \times x, & \text{otherwise} \end{cases} \tag{5.9}$$

（2）图神经网络计算

对于同维度映射之后的变量特征、约束特征、边特征，通过如下图神经网络方法进行特征向量的计算：

$$\begin{aligned}
\mathbf{x}_c^i &\leftarrow \mathbf{f}_{\mathcal{C}}\left(\mathbf{x}_c^i, \sum_j^{(i,j)\in E} \mathbf{g}_{\mathcal{C}}\left(\mathbf{x}_c^i, \mathbf{x}_v^j, \mathbf{x}_e^{i,j}\right)\right) \\
\mathbf{x}_v^j &\leftarrow \mathbf{f}_{\mathcal{V}}\left(\mathbf{x}_v^j, \sum_j^{(i,j)\in E} \mathbf{g}_{\mathcal{V}}\left(\mathbf{x}_v^j, \mathbf{x}_c^i, \mathbf{x}_e^{i,j}\right)\right)
\end{aligned} \tag{5.10}$$

其中函数 $\mathbf{g}(\cdot)$ 的计算方式为：

$$g\left(\mathbf{x}_c^i, \mathbf{x}_v^j, \mathbf{x}_e^{i,j}\right)=\mathrm{MLP}\left(\mathbf{x}_c^i+\mathbf{x}_v^j+\mathbf{x}_e^{i,j}\right) \tag{5.11}$$

其中 MLP 是以 LeakyRELU 为激活函数的双层全连接神经网络，函数 $\mathbf{f}(\cdot)$ 也

是以 LeakyRELU 为激活函数的双层全连接神经网络。因此该图神经网络计算方法与文献 [147] 的计算方法相似，由两部分图卷积过程构成，第一步卷积过程计算第 i 个约束的特征向量，它由与第 i 个约束存在连接关系的所有变量 j 的特征 $\mathbf{x}_v^j$、边的特征 $\mathbf{x}_e^{i,j}$ 以及其自身的特征进行聚合，从而得到第一步更新后的约束特征向量。第二步卷积过程计算第 j 个变量的特征向量，它由与第 j 个约束存在连接关系的所有约束 i 的特征 $\mathbf{x}_c^i$、边的特征 $\mathbf{x}_e^{i,j}$ 以及其自身的特征进行聚合，从而得到第二步卷积后的变量特征向量，该特征向量即为图神经网络输出的变量特征向量，该向量聚合了 MILP 的图结构信息，聚合了约束和变量之间的信息和关系，从而能够有效反映变量的图特征。

（3）历史特征向量计算

在第 t 步分支定界的分支决策过程，可以对当前步的历史特征进行计算，由两部分构成，第一部分为：从初始状态开始到上一步分支，之前所有步 $1\cdots t-1$ 选择的分支变量的集合 $\mathcal{C}_1 = a_1 \cdots a_{t-1}$，由于图神经网络每一步都可以输出所有变量的特征向量，因此可以采用循环神经网络的方式计算整个决策序列的历史分支情况，但是由于该模型的最终目标是降低计算压力，因此采用直接求平均值的方式对历史分支情况进行估计，则该部分历史特征向量计算为：

$$\mathbf{x}_{h1}^t = \mathrm{MLP}\left(\frac{1}{t-1}\sum_{i=1}^{t-1}\mathbf{x}_v^{a_i}\right) \tag{5.12}$$

其中 MLP 为一个单层的全连接神经网络层，a_i 为第 i 步分支决策选择的变量。

历史特征的第二部分为：生成当前节点的过程中，变量值发生改变的变量集合 $\mathcal{C}_2$，同样的对该集合内所有的变量特征向量进行取平均值操作，得到代表该部分特征的向量 $\mathbf{x}_{h2}^t$，并同样采用一个单层的全连接神经网络层对其进行线性映射。如果 $t == 0$，那么 $\mathbf{x}_{h2}^t$ 和 $\mathbf{x}_{h1}^t$ 是零向量。

（4）注意力机制

因此可以定义一个查询 *query* 向量，该向量由全局特征、两部分历史特征组成，代表求解器当前的状态，*query* 向量计算为这三部分特征向量的加权平均：

$$\mathbf{q_t} = w_1 * \mathbf{x}_g^t + w_2 * \mathbf{x}_{h1}^t + w_3 * \mathbf{x}_{h2}^t \tag{5.13}$$

其中 w_1, w_2, w_3 是可训练的权重值。

定义“键”*key* 向量为所有候选变量的特征向量的线性映射 $k_i = W_k\mathbf{x}_v^i, i \in \mathcal{C}$，则注意力机制定义为，在第 t 步决策过程中，代表该步求解器状态的 $\mathbf{q_t}$ 对所有候选变量 $\mathbf{k_i}, i \in \mathcal{C}$ 的查询，计算得到的注意力值即输出的选择每个变量的概率，具

体地，注意力计算为：

$$\begin{aligned} u_i^t &= W_3\left(W_1 k_i + W_2 q_t\right) \quad i \in (1,\ldots,n) \\ a_i^t &= \operatorname{softmax}\left(u_i^t\right) \qquad\quad i \in (1,\ldots,n) \end{aligned} \tag{5.14}$$

输出的 a_i^t 即为第 t 步分支决策中，选择第 i 个变量的概率值，此时可以选择概率最大的变量作为分支变量。

（5）数据归一化层

由于约束系数、各个特征的值的数值范围不统一，所以有必要对输入的变量特征、约束特征、边特征、全局特征进行归一化操作，但是传统神经网络的归一化方式，如 LayerNorm，是针对神经网络各层的数据统计特性进行归一化，而无法扩展到不同规模的问题上，比如当组合优化问题规模变大，约束的数量增多，在训练过程中采用的归一化算子则无法应用到该新问题上，因此本章参考 [147] 的思路，在神经网络模型上添加归一化层，并且添加了全局特征的归一化层，使得全局特征能够泛化到不同规模的问题上。

具体地，归一化层的计算方法介绍如下，首先设计一个简单的线性变换 $\mathbf{x} \leftarrow (\mathbf{x} - \beta)/\sigma$，$\mathbf{x}$ 即为数据集的变量特征、约束特征、边特征和全局特征，β 和 σ 参数是归一化的系数，分别通过对数据集 $\mathbf{x}$ 的均值和标准差进行计算得到，初始化完成之后对该值进行固定，在训练过程中不再变化。这种归一化系数的计算遍历了整个训练数据集，从而能够使得该归一化算子能够应对不同类型的数据集，因此给定一个新问题，该归一化系数仍然能够实现可靠的归一化，从而提高算法的泛化性能。

5.2.4 基于图指针神经网络的分支定界流程

利用上述的图指针神经网络 GANN 对变量选择的策略进行建模，采用该模型定义的策略进行变量选择，基于该分支策略的分支定界算法流程如算法5.1所示。

具体地，首先将 MILP 原问题的线性松弛作为根节点，并维护一个队列数据结构，用于存储待求解的子问题节点，每个节点定义一个初始下界，代表其父节点的最优解的目标值，这种方法提供了“全局上界更新后，如果存在节点的下界大于全局上界，则对该节点进行剪枝”的功能，当从队列中取出该节点时，对其父节点的下界和全局上界进行了比较，如果该下界大于全局上界，那么对该节点进行剪枝。全局上界初始化为 ∞，每次得到更优的可行解时更新上界。

每次从队列中取出一个子问题进行求解，如果不满足整数约束，计算候选变量的变量特征、约束特征、边特征、全局特征、历史特征，输入到图指针神经网络模型中，模型输出选择各个变量的概率，以贪婪的方法选择概率最大的变量作为分支变量，并根据该变量添加两个约束生成两个子问题节点，添加到队列中。直

算法 5.1 基于图指针神经网络的分支定界法

算法输入： 根节点 R，代表原始 MILP 的线性松弛

算法输出： 最优解

$R.lowerBound \leftarrow -\infty$ // 初始化根节点的下界
$Queue \leftarrow \{R\}$ // 探索的节点存储在队列中
$UpperBound \leftarrow \infty$ // 初始化全局上界
$S^* \leftarrow null$ // 最优解
while $Queue$ 不为空 **do**
    $N \leftarrow Queue.get()$ // 取出队列里第一个节点
    if $N.lowerBound \geq UpperBound$ **then**
        // 节点 N 的父节点的下界比当前全局上界大，不继续求解，剪枝
        continue
    end if
    $S_r \leftarrow \text{solve}(N)$
    if S_r 无解 **then**
        // 该节点无解，剪枝
        continue
    end if
    $O_r \leftarrow S_r.objectiveValue$
    if $O_r > UpperBound$ **then**
        // 该节点下界比当前全局上界大，不继续求解，剪枝
        continue
    end if
    if S_r 为满足所有整数约束的可行解 **then**
        $UpperBound \leftarrow O_r$
        $S^* \leftarrow S_r$
        // 找到比当前最优解更好的解，更新全局上界
        continue
    end if
    提取当前状态特征 $state = (V, C, E, G, H)$
    $V \leftarrow \text{GPN}(S_r, state)$ // 调用神经网络模型从 S_r 选择变量 V
    $a \leftarrow \text{floor}(V.value)$
    $L \leftarrow \text{addConstraint}(N, V \leq a)$
    $R \leftarrow \text{addConstraint}(N, V \geq a)$
    $L.lowerBound \leftarrow O_r$，$R.lowerBound \leftarrow O_r$
    $Queue.add(L)$，$Queue.add(R)$
end while
return S^*

到队列里所有问题都被求解完毕，返回最优解。

5.3 模型训练方法

分支定界的变量选择决策过程可以建模为马尔科夫过程，并采用强化学习方法对策略进行优化。传统强化学习方法，例如策略梯度方法或者值函数法，通过计算累积奖励值来对策略进行学习，这在奖励信号频繁的情况下有较好的表现，然而分支定界法的目标是最小化搜索步数或者求解时间，需要大量的决策才能获得最终奖励，是典型的“多步决策”问题，如果采用传统的基于累积奖励值的强化学习方法进行策略优化，会产生收敛慢、奖励信号稀疏、训练效率低等问题。

模仿学习（Imitation Learning）是强化学习的一个分支，因其能很好的解决强化学习中的多步决策问题，近年来得到了广泛关注[148]，在自然语言处理、机器人控制等领域存在很多应用。模仿学习通过行为克隆的方式对智能体的策略进行学习，使智能体的策略模仿专家动作，通过专家经验来指导智能体的行为，从而克服单独采用强化学习方法奖励信号稀疏、训练效率低、模型收敛慢甚至无法收敛的问题。

模仿学习的基本思想是从专家经验提供的范例动作中进行学习，需要人为提供带标签的决策数据 $\{\tau_1, \tau_2, \ldots, \tau_m\}$，每个决策实例 τ_i 包含求解该实例过程中的“状态 - 动作”序列 $\tau_i =< s_1^i, a_1^i, s_2^i, a_2^i, \ldots >$，因此，可以将每个决策实例中的“状态 - 动作”对提取出来，组成带标签的监督学习训练集 $\mathcal{D} = \{(s_1, a_1), (s_2, a_2), (s_3, a_3), \ldots\}$。

通过上述过程，可以把决策问题转换为分类问题，把每一步的状态作为特征，把相应的动作作为标签，从而通过梯度下降方法以最小化标签和预测值之间的差异来对模型进行训练，即模仿专家经验，因此称为模仿学习。

本节采用模仿学习以 strong branch 分支规则作为模仿对象对模型进行训练，strong branch 规则通过计算每个候选变量的 SB 分数来进行分支决策，但是计算所有候选变量的 SB 分数需要大量耗时，因此通过对 strong branch 规则进行模仿从而得到耗时少但是表现相近的分支模型。

首先需要构建训练数据集，过程如下：针对某一类组合优化问题，每次随机生成一个问题实例，采用 SB 规则对该实例进行求解，求解过程中，记录“状态 - 动作”对，状态即上节所述的求解器状态特征，动作即选择的分支变量，通过求解大量问题实例，训练数据集 $\mathcal{D} = \{(\mathbf{s}_i, \mathbf{a}_i^\star)\}_{i=1}^N$ 由求解过程中记录的“状态 - 动作”对构成。

根据训练数据集，可通过最小化专家样本动作 $\mathbf{a}^\star$ 和策略模型输出动作 $\pi(s)$ 之间的差异来对模型的参数进行更新：

$$\mathcal{L}(\theta) = \frac{1}{N} \sum_{(\mathbf{s},\mathbf{a}^*)\in\mathcal{D}} \text{loss}\left(\pi_\theta(\mathbf{s}), \mathbf{a}^\star\right). \tag{5.15}$$

$\text{loss}(*)$ 是定义真实值和预测值之间差异的函数，文献中存在多种 $\text{loss}(*)$ 函数，最常用的是交叉熵函数：

$$\text{loss}(p,\ \textbf{class}\ ) = -\log\left(\frac{\exp(p[\ \textbf{class}\ ])}{\sum_j \exp(p[j])}\right) = -p[\ \textbf{class}\ ] + \log\left(\sum_j \exp(p[j])\right) \tag{5.16}$$

其中 class 是训练集中通过 SB 规则选择的变量序号，p 是神经网络模型输出的变量选择的概率。

但在分支定界问题中，考虑到不同变量可能具有相同的 SB 分数，或者多个变量的 SB 分数相差极小，此时选择该不同变量可能具有相同的效果，采用传统的模仿学习方法无法应对这种情况，无法使模型对 SB 分数的真实分布进行有效地学习。

鉴于此，构造训练集时，不对变量选择的动作进行收集，而是收集 SB 分数，从而构成训练集 $\mathcal{D} = \{(\mathbf{s}_i, score_i^\star)\}_{i=1}^N$，进一步地，采用 KL 散度（Kullback–Leibler divergence）作为衡量预测值和真实值之间差异的 $\text{loss}(*)$，KL 散度用来衡量一个分布与另一个分布的相似性，因此针对于分支定界的变量选择问题，可以采用 KL 散度衡量神经网络模型输出的概率分布与 SB 分数的分布之间的相似性，以最小化 KL 散度为目标可以使模型输出与 SB 分数相似的概率分布，从而克服多个变量之间 SB 分数相同或相似的问题。

具体地，考虑两个概率分布 P 和 Q，P 是数据的真实分布，Q 是模型输出用来拟合 P 的近似分布，KL 散度定义为两个分布之间的相似性，当值为 0 时两个分布完全相同，计算如下：

$$D_{\text{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right) \tag{5.17}$$

针对本章所述的分支定界变量选择问题，$\pi_\theta(\mathbf{s})$ 是模型输出的近似分布，目标是使得该分布尽可能与训练样本中 SB 分数的分布 $score^\star$ 相同，因此该问题的损失函数定义为：

$$\mathcal{L}(\theta) = D_{\text{KL}}(score^\star\|\pi_\theta(\mathbf{s})) = \sum_{(\mathbf{s},score^\star)\in\mathcal{D}} score^\star \log\left(\frac{score^\star}{\pi_\theta(\mathbf{s})}\right) \tag{5.18}$$

另外，虽然候选变量集合中存在大量候选变量，但是决策过程中只关心 SB 分

数最高的几个变量，其他变量的概率分布对于变量选择的决策没有影响，因此采用 KL 散度对两个分布的相似性进行衡量时，对概率值最高的几个变量的分布应该着重考虑。如果对模型输出的概率值按从大到小进行排序，那么前几个概率值的分布与真实 SB 分数的分布之间的差异性更加重要，排序在后面的大量变量的概率值分布与 SB 分数分布之间的差异性不会影响变量的选择过程，因此更应该关注前几个变量的分布特性。鉴于此，本节提出 top-k KL 散度，即关注 top-k 个变量的概率分布相似性。

具体地，首先对模型输出的概率值 $\pi_\theta(\mathbf{s})$ 进行排序，选择前 k 个概率值最大的变量，其集合为 $\mathcal{I}_k$，计算这 k 个变量的 KL 散度值 $D_{\mathrm{KL}}(score^\star_{\mathcal{I}_k} \| \pi_\theta(\mathbf{s})_{\mathcal{I}_k})$，计算方法与式 (5.17) 相同。

最终的训练损失函数定义为：

$$\mathcal{L}(\theta) = D_{\mathrm{KL}}(score^\star \| \pi_\theta(\mathbf{s})) + D_{\mathrm{KL}}(score^\star_{\mathcal{I}_k} \| \pi_\theta(\mathbf{s})_{\mathcal{I}_k}) \tag{5.19}$$

损失函数的第一项能够使得模型输出概率的总体分布与 SB 分数的分布相似，第二项使得模型更加关注概率最大的几个变量的分布特性，从而弱化无关变量的分布与真实分布存在差异对模型的影响，防止出现大量训练资源用在拟合大部分无关变量概率分布的情况。通过基于 top-k KL 散度的模仿学习训练方法，实现训练效率的提高。

5.4 实验结果与讨论

5.4.1 实验设置

由于本章所提方法（GPN）是对传统分支定界法的改进，因此首先与经典分支定界法进行比较，并进一步与当前基于机器学习改进的分支定界方法进行比较。采用业界广泛使用的 SCIP 开源优化求解器作为基准测试环境，所有方法均基于 SCIP 进行实现。

5.4.1.1 测试问题

本章在三个经典的组合优化问题上进行方法测试。

（1）集合覆盖问题（Set Cover Problem）

采用集合覆盖问题 [149] 作为第一个测试问题，测试问题包含 1000 个变量。首先在具有 500 个约束的的训练集上进行模型训练，随后在 500 和 1000 规模的测试问题上进行模型评估。

（2）带容量约束的设备选址问题（Capacitated Facility Location Problem）

采用带容量约束的设备选址问题 [150] 作为第二个测试问题，问题包含 100 个设备，首先在 100 客户的测试问题上进行模型训练，随后在 100 和 200 客户规模

的测试问题上进行模型评估。

（3）最大独立集问题（Max Independent Set Problem）

最后在最大独立集问题 [151] 上进行测试，按照文献 [151] 中介绍的过程进行 Erdos-Rényi 随机图的生成，首先在 500 个节点的问题上进行模型训练，随后在 500 和 1000 规模的测试问题上进行模型评估。

5.4.1.2 对比算法

用于对比的方法包括：

（1）首先与当前使用最广泛的 Reliability pseudocost（RPB）分支规则进行实验比较，这是 SCIP 求解器中默认的分支方法。其次对经典的 Strong Branching（SB）规则以及 Pseudocost（PB）规则进行了实验对比。上述方法基于 SCIP 求解器实现，并且只允许在根节点采用割平面，在分支过程中禁用其他启发式方法，其余 SCIP 参数均设置为默认值。

（2）其次，与四个经典的基于机器学习的分支规则进行实验比较：基于 Extra-Trees [152] 模型的分支打分方法 [153] (TREES)；基于 SVMrank [154] 和 LambdaMART [155] 模型的 [156] 分支排名方法 (SVMRANK) 和 [157] (LMART)；基于图神经网络的分支方法[147](GNN)。本章所提方法与文献 [147] 采用了相似的图神经网络建模方式，但是设计了更加有效的全局特征和历史特征，并且设计了基于图指针神经网络的建模方法。

5.4.1.3 实验参数设置

所有对比算法均采用完全相同的 SCIP 求解器参数，采用不同的分支规则进行实验。。本章所提模型的隐层向量维度均设置为 $d_h = 64$，均采用 Adam 优化器以 0.001 的学习率进行训练，模仿学习 top-k 损失函数的 k 值设置为 10，训练时如果连续 10 个 epoch 损失函数不下降，则降低学习率至 20%，连续 20 个 epoch 损失函数不下降，则终止训练。

5.4.1.4 数据生成和处理

（1）训练数据生成

用于模仿学习的训练数据通过离线方式生成。每次随机生成一个问题实例，采用 SCIP 求解器对问题进行求解，求解过程中按照如下方式进行样本的收集：

首先，在求解过程中，以 95% 的概率采用 SCIP 默认的 RPB 规则进行分支，以 5% 的概率采用 SB 规则进行分支。

在收集过程中，只收集按照 SB 分数进行分支的样本，包括当前的图结构状态（变量特征、约束特征、边特征、全局特征、历史特征），候选变量集合，候选变量中每个变量的 SB 分数。

最后，不断随机生成问题并进行求解，直到收集到 140,000 个样本，100,000 个样本作为训练集，2,000 个样本作为验证集，2,000 个样本作为测试集。

（2）测试方法

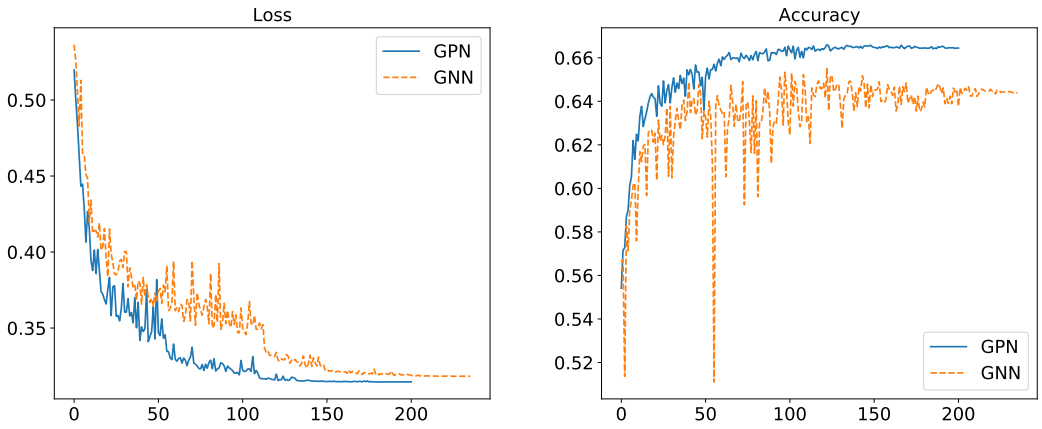
首先对模仿的准确率进行测试，由于不同候选变量可能具有相同的 SB 分数或者多个变量的 SB 分数差距很小，因此采用如下准确率评价指标[147]：1）模型输出变量是 SB 分数最高的变量的比例（acc@1）；2）模型输出变量在 SB 分数最高的前 5 个变量当中的比例（acc@5）；3）模型输出变量在 SB 分数最高的前 10 个变量当中的比例（acc@10）。acc@1 即为传统的准确率，acc@5 和 acc@10 评估模型输出的变量概率分布是否与 SB 分数的分布具有相似性[147]，通过上述方法可以有效地评估模型效果。

其次对算法的求解时间进行对比测试，所有测试方法均基于 SCIP 求解器实现，使用相同的默认参数，唯一的区别是分支时调用的分支规则。

5.4.2 模型性能评估实验结果

主要从两个方面对本章所提方法进行实验分析，首先与当前基于机器学习的分支方法进行比较，以验证本章所提模型的性能，其次与传统方法和机器学习方法进行分支定界求解时间的比较，以验证所提方法在提高精确算法求解速度上的优势。

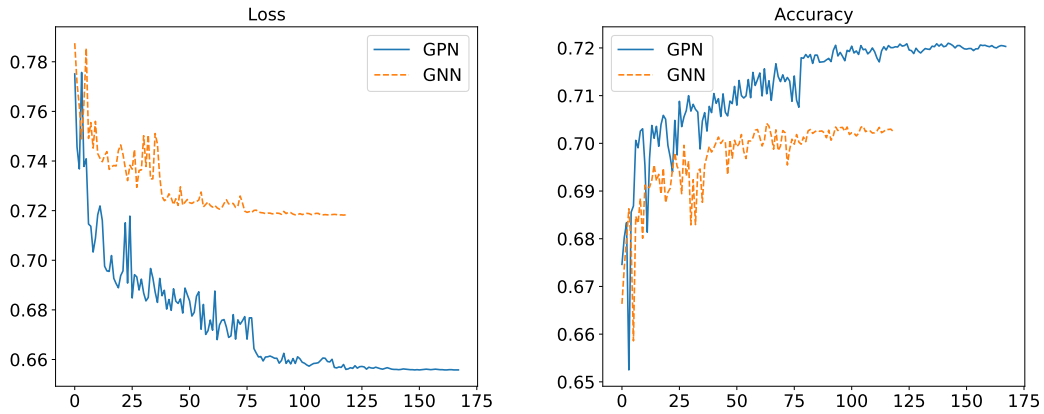
图 5.3 、图 5.4 、图 5.5 展示了本章所提的图指针神经网络模型（GPN）与传统图神经网络模型（GNN）在三个测试问题上的训练表现，分别展示了训练过程中 Loss 的收敛情况以及在验证集上的模型准确率。



(a) Loss 收敛曲线　　(b) 模型准确率收敛曲线

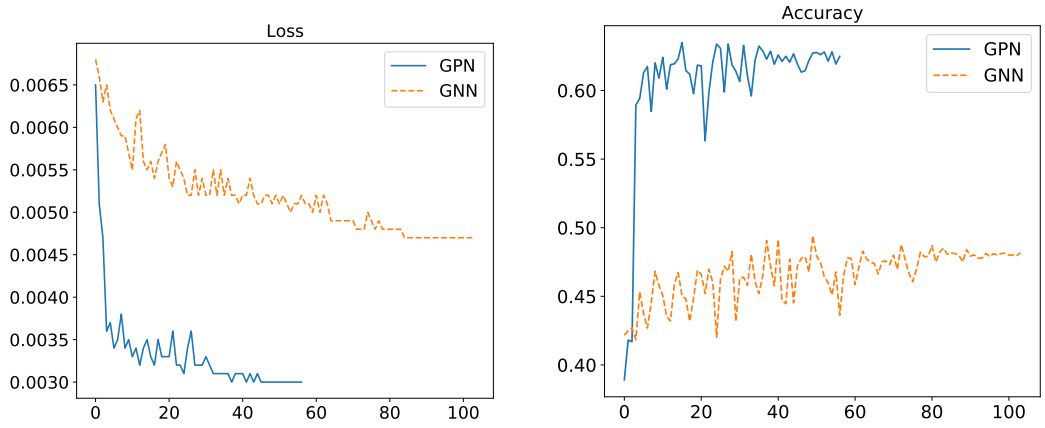
图 5.3　集合覆盖问题对比模型 Loss 和模型准确率收敛曲线

从该实验结果可见，本章所提方法在收敛速度上和最终收敛性能上均优于



(a) Loss 收敛曲线　　(b) 模型准确率收敛曲线

图 5.4　设备选址问题对比模型 Loss 和模型准确率收敛曲线



(a) Loss 收敛曲线　　(b) 模型准确率收敛曲线

图 5.5　最大独立集问题对比模型 Loss 和模型准确率收敛曲线

GNN 方法，GNN 方法是当前采用图神经网络对分支规则进行建模的基准模型，本章提出的的图指针神经网络在模型性能上能够有效超越该方法，并且可以发现，GPN 方法在训练过程中模型的 Loss 下降过程更加稳定、收敛速度更快、最终的收敛性能最优，而 GNN 模型的性能很早便无法继续提升，达到了模型的最大能力，这主要是本章提出的基于注意力的指针机制能够综合考虑问题的图特征和全局特征，有效地对变量进行选择。

在带容量约束的设备选址问题和最大独立集问题上，GPN 模型的优势更加明显，在模型收敛速度和最终的收敛性能上取得了更大优势，这也体现在后续在求解时间的模型表现上。

表 5.4 、表 5.5 、表 5.6 展示了在三个测试问题上，本章所提 GPN 方法与 TREES、SVMRANK、LMART、GNN 方法在模型准确率上的比较结果，准确率计

算为模型输出变量与基于 SB 规则选择得到变量的准确率，并分别列出了 acc@1、acc@5 和 acc@10 准确率的结果。

表 5.4　集合覆盖问题上对比方法的模型准确率

	acc@1	acc@5	acc@10
TREES	51.8	80.5	91.4
SVMRANK	57.6	84.7	94
LMART	57.4	84.5	93.8
GNN	65.5	92.4	**98.2**
GPN	**66.5**	**92.7**	**98.2**

表 5.5　设备选址问题上对比方法的模型准确率

	acc@1	acc@5	acc@10
TREES	63	97.3	**99.9**
SVMRANK	67.8	98.1	**99.9**
LMART	68	98	**99.9**
GNN	71.2	98.6	**99.9**
GPN	**72.2**	**98.7**	**99.9**

表 5.6　最大独立集问题上对比方法的模型准确率

	acc@1	acc@5	acc@10
TREES	30.9	47.4	54.6
SVMRANK	48	69.3	78.1
LMART	48.9	68.9	77
GNN	56.5	80.8	89
GPN	**63.2**	**86.9**	**92.6**

可见 GPN 方法在所有测试问题上均得到了最高的准确率，在最大独立集问题上取得了明显优于传统机器学习方法的结果，验证了该模型的有效性。

5.4.3　求解时间对比实验结果

分支定界法的求解速度与搜索树的规模（节点数目）有关，同时与每个分支策略的计算成本、每个节点的求解时间也相关，因此采用算法总求解时间和搜索树的规模作为评估指标。

表 5.7 、表 5.9 、表 5.9 列出了 GPN 与对比方法在三个测试问题上的实验结果，数值结果为各个方法在 100 个随机生成问题上的平均表现。每个测试问题均在与训练集相同规模以及更大规模的问题上进行测试，由于对比的 PB、RPB、SB 规则可以在 CPU 上多线程运行，因此表中的时间结果为并行运行的平均结果，其余方法均基于 GPU 和 CPU 进行单线程运行。

由表 5.7 可见，在求解 500 和 1000 规模的集合覆盖问题上，相对于 SCIP 优化求解器中使用的 PB 分支规则和 RPB 分支规则，GPN 方法至少达到了 40% 的求解速度的提升，在节点数量上，求解 1000 规模的集合覆盖问题时，GPN 方法得到的节点数量仅多于 SB 方法，显示了其模型良好的性能。SB 规则总是可以得到最小规模的搜索树，但是由于分支策略计算的长耗时，总求解时间并不存在优势。相对于图神经网络方法，GPN 方法在求解速度和节点数量上均得到了提升，并且在更大规模的问题上也得到了稳定的优化效果。

由表 5.8 可见，在求解 100 规模和 200 规模带容量约束的设备选址问题上，GPN 方法显示出了更大的优势，相对于 SCIP 优化求解器中使用的 PB 分支规则和 RPB 分支规则，GPN 方法达到了两倍左右的求解速度的提升，相对于图神经网络方法，在求解速度和节点数量上获得了更大的提升，SB 方法需要分支的节点数远小于其他算法，但求解时间也最长。

由表 5.9 可见，在求解 500 和 1000 规模的最大独立集问题上，相对于图神经网络方法，GPN 方法在求解速度和节点数量上获得了将近 10% 和 20% 的提升，相对于专家设计的 PB、RPB、SB 规则，求解速度达到了两倍左右的提升，需要注意的是，由于 PB、RPB、SB 规则是在 CPU 上多线程实现的，因此线程间可能存在时间损耗，因此求解时间会略长于单线程运行的算法，但是单线程运行需要大量时间才能对每个问题的 100 个实例进行测试，因此采用多线程进行算法实现，GPN 方法相对于传统的 PB、RPB、SB 方法获得了大量的求解速度提升，因此多线程导致的求解时间误差可以安全地忽略。

同时可以看出，本章提出的 GPN 方法能够很好地泛化到更大规模的问题上，在所有类型和规模的测试问题上，其运行时间都优于专家设计的启发式分支规则和图神经网络方法，取得了比专业的优化求解器 SCIP 更快的求解速度。

分支定界法的目标是以尽可能快的速度求解组合优化问题，因此分支策略必须在决策质量和每个决策所花费的时间之间进行权衡，一个极端例子就是 SB 分支规则，通过计算 SB 分数进行变量选择，可以以很少的搜索次数得到最终解，但每个决策步骤都非常耗时，以至于整体运行时间很长。因此本章所提方法能够更好的实现决策质量和决策时间之间的权衡，方法可得到比 SB 分数略差的分支效果，但是需要更少的计算时间，从而提高了整体求解速度。

表 5.7　集合覆盖问题上对比方法的求解速度对比

	500 规模		1000 规模	
方法	求解时间	节点数量	求解时间	节点数量
SB	7.02	**12.5**	173.9	**227.5**
PB	2.88	98.6	19.8	2211.2
RPB	3.73	18.8	22.1	1192.5
GNN	2.07	43.9	14.2	900.6
GPN	**2.04**	42.3	**13.9**	891.2

表 5.8　设备选址问题上对比方法的求解速度对比

	100 规模		200 规模	
方法	求解时间	节点数量	求解时间	节点数量
SB	157.4	**116.5**	1163.3	**158.7**
PB	82.8	541.9	510.7	614.2
RPB	96.7	264.7	598.9	303.5
GNN	37.4	467.4	145.6	529.6
GPN	**35.2**	428.9	**140.3**	516.2

5.5　本章小结

本章对求解组合优化问题的分支定界法进行了研究，提出了一种基于深度强化学习的分支定界组合优化方法。区别于通过专家经验对分支定界法分支规则进行设计的传统方式，本章设计了一种图指针神经网络模型对变量选择的规则进行自动学习，将图神经网络模型和注意力机制相结合，从而有效提高模型效果。将分支定界的变量选择过程建模为马尔可夫决策过程，将每步求解器的状态建模为

表 5.9　最大独立集上对比方法的求解速度对比

	500 规模		1000 规模	
方法	求解时间	节点数量	求解时间	节点数量
SB	87.1	**35.41**	2844.4	**164.5**
PB	14.6	1937.8	2002.9	17213
RPB	11.4	92.7	210.2	6717
GNN	5.01	61.7	222.5	14862
GPN	**4.63**	45.3	**198.6**	12587

图结构，在对变量特征、边特征、约束特征进行提取的基础上，额外提取了问题的全局特征和历史特征，从而实现对当前状态更加丰富的表征，以提高模型表现。设计了一种基于 top-k KL 散度的模仿学习方法对模型进行训练，克服了传统训练方法训练效率低的缺陷。通过实验表明，本章所提出的方法在求解速度上明显超越了传统人工设计的分支规则，比当前主流优化求解器 SCIP 求解速度提升了 40% 左右，并取得了比传统基于机器学习的分支方法更好的效果，能够实现对分支定界法的有效加速。

第六章　基于深度强化学习的无人机路径规划方法

本章将论文所研究的深度强化学习组合优化方法应用到基于无线能量传输的无人机路径规划问题上，无人机由于其灵活性和多功能性，在运输、军事任务、救灾、通信等各种应用中发挥着至关重要的作用，然而，无人机的续航时间是制约其应用的瓶颈问题，近年来随着无线能量传输技术的发展，通过无线传能的可以实现无人机的无线充电，从而有效扩展无人机的工作时间和工作范围。本章采用深度强化学习方法对无线传能场景下的无人机在线路径规划问题进行研究，对无人机功耗模型和无线能量传输模型进行构建，通过深度强化学习方法对路径优化模型进行离线训练和在线应用，相对于 Google OR-tools 求解器，本章所提方法在不同规模的实际问题上实现了 4 倍到 500 倍求解速度的提升。

6.1　考虑无线传能的无人机路径规划问题

近年来，无人机在物流交通 [158]、灾害救援 [159]、军事侦察打击 [160]、工业巡检 [161]、无线通信 [162] 等领域取得了广泛的应用，但是由于无人机电池容量的限制 [163]，其续航时间通常只有数十分钟，电量耗尽时需要进行降落、回收、再起飞等一系列操作，但是在灾害救援、军事应用等场景下，由于复杂的地理工况和对抗环境，通常不方便甚至不安全进行无人机降落起飞，从而极大制约了无人机效能的发挥，在 2021 年 12 月，美国国防高级研究计划局 (DARPA) 对 Electric Sky 公司的无人机无线能量传输技术研究进行资助，从而扩展无人机在极端环境和战场环境下的搜索、救援、侦察和打击等能力。

鉴于此，本章对无线能量传输场景下的无人机路径规划问题进行研究，在灾害、军事等应用场景下，系统的响应速度尤为重要，但是随着问题规模和复杂度的增加，传统优化方法很难实现问题的实时、在线优化，因此本章将基于深度强化学习的组合优化方法应用到该问题上，从而实现无人机路径的实时规划。

文章考虑了一个基于无线传能的无人机路径规划的典型场景，给定一系列目标任务点，旨在找到一条遍历所有目标任务点且飞行时间最短的路径，无人机以满电量状态从基站出发，最后返回基站，由于无人机电池电量受限，一旦其荷电状态 (SoC) 降低到一定水平（文章设定为 20%），必须返回基站进行无线充电，当无人机与基站距离达到一定阈值 x_{start} 时，基站即可为无人机进行充电，如果路途中未充满，则无人机在基站顶部悬停直到电量充满。通过该无线传能充电方式，可以实现无人机任务全流程的无人化，极大降低极端环境下的人工成本和安全风险。

6.2 系统建模

本章假设目标任务点和基站分布在二维平面中，不考虑地形因素，无人机以固定高度 H 飞行，选择四旋翼无人机作为研究对象，首先对无人机功耗模型、无线能量传输模型、以及系统优化模型进行构建。

6.2.1 无人机功耗模型

假设无人机在目标任务点和基站之间以恒定速度 V 飞行，四旋翼无人机在速度 V 条件下的功耗为 [164]:

$$P(V) = P_0\left(1+\frac{3V^2}{U_{tip}^2}\right) + P_i\left(\sqrt{1+\frac{V^4}{4v_0^4}}-\frac{V^2}{2v_0^2}\right)^{\frac{1}{2}} + \frac{d_0\rho sAV^3}{2} \tag{6.1}$$

其中 P_0 和 P_i 是叶片功率和感性负载功率:

$$P_0 = \frac{\delta}{8}\rho sA\Omega^3R^3 \tag{6.2}$$

$$P_i = (1+k)\frac{W^{\frac{3}{2}}}{\sqrt{2\rho A}} \tag{6.3}$$

其中 ρ、s、A、Ω、R、W 分别是空气密度、旋翼实度、转子盘面积、叶片角速度、转子半径和无人机重量，假设无人机的物理属性和飞行环境保持不变，则 P_0 和 P_i 是常数，v_0 表示转子平均速度，U_{tip} 是转子叶片的叶尖速度，分别计算为:

$$v_0 = \sqrt{\frac{W}{2\rho A}} \tag{6.4}$$

$$U_{tip} \triangleq \Omega R \tag{6.5}$$

其中 $d_0 \triangleq \frac{S_{FP}}{sA}$ 为机身阻力比，S_{FP} 是机身等效平板面积。

无人机以 V 速度飞行时消耗的功率可以通过 式 (6.1) 计算为 $P(V)$；当无人机悬停，即 $V=0$ 时，其功耗为 $P(0)=P_0+P_i$。因此，四旋翼无人机飞行和悬停所需的能量可以计算为 $E_V = P(V)\cdot T_{\text{flying}}$ 和 $E_{hover} = P(0)\cdot T_{\text{hover}}$。

6.2.2 无线能量传输模型

假设基站与无人机之间无阻挡、为视距链路，因此采用自由空间路径损耗（FSPL）模型 [165] 对无线能量传输损耗进行计算：

$$PL(d) = 20\lg\{f\} + 20\lg\{d\} - 147.55dB \tag{6.6}$$

其中 f 是载波频率，d 是无人机与基站之间的距离。

进一步考虑射频 - 直流（RF-DC）转换损耗，针对 RF-DC 转换损耗，存在多种线性和非线性模型，本章采用转换效率 η 固定的线性损耗模型 [164] 进行计算：

$$P_{DC} = \eta \cdot P_{RF} \tag{6.7}$$

其中 P_{RF} 是能量收集器的输入射频功率，P_{DC} 是输入到电池中的直流功率。

将无人机实际接收到的功率表示为 $P_{charge}(d)$，则：

$$P_{charge}(d) = P_t + G_t + G_{uav} - PL(d) \tag{6.8}$$

其中 P_t 是基站的发射功率，单位是 dBW，G_t 是基站发射天线增益，单位是 dBi，G_{uav} 是无人机接收天线增益，单位是 dBi，$PL(d)$ 为基站与无人机之间无线信道的传输损耗，单位为 dB，即：

$$P_{charge}(d) = P_t + G_t + G_{uav} - 20\lg\{f\} - 20\lg\{d\} + 147.55dB \tag{6.9}$$

可见充电功率随距离 d 变化，因此，充电过程可以分为两个阶段：无人机在到达基站顶部之前进行移动充电；无人机在基站顶部进行悬停充电，分别建模如下。

6.2.2.1 悬停充电模型

无人机在 H 高度进行悬停充电时，充电功率恒定：

$$P_{\text{charge}}^{\text{hover}} = P_t + G_t + G_{uav} - PL(H) \tag{6.10}$$

悬停充电模式无人机实际接收到的直流能量计算为：

$$E_{\text{charge}}^{\text{hover}} = \eta 10^{\frac{P_{\text{charge}}^{\text{hover}}}{10}} T^{\text{hover}} \tag{6.11}$$

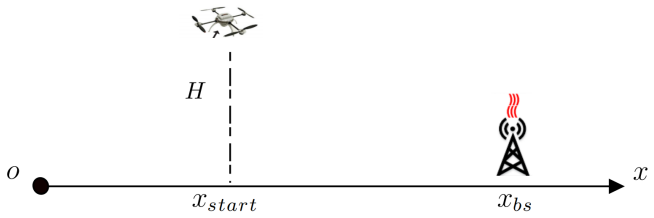


图 6.1　无人机无线充电过程示意图

由于无人机悬停时仍会消耗能量，消耗能量计算如下：

$$E_{\text{use}}^{\text{hover}} = P(0) \cdot T^{\text{hover}} \tag{6.12}$$

将无人机到达基站顶部时仍然需要进行补充的电量记为 E_{hover}，则悬停充电时间可计算为：

$$T^{\text{hover}} = \frac{E_{\text{hover}}}{\eta 10^{\frac{P_{\text{uav-r}}}{10}} - P(0)} \tag{6.13}$$

6.2.2.2　移动充电模型

当无人机靠近基站时，基站开始为无人机充电，该过程可见图 6.1 ，开始充电的位置由无人机能量收集器的激活阈值 P_ϵ 决定，将开始充电位置的水平坐标表示为 x_{start}，则满足：

$$P_t + G_t + G_{uav} - 20\lg\{f\} - 20\lg\{d_{\text{start}}\} + 147.55dB \geqslant P_\epsilon dB \tag{6.14}$$

其中 $d_{\text{start}} = \sqrt{(x_{bs} - x_{\text{start}})^2 + \Delta H^2}$ 为基站与无人机开始充电位置之间的距离，因此无人机开始充电的位置可以计算为：

$$x_{\text{start}} = x_1 - \sqrt{10^{\frac{P_t + G_t + G_{uav} - 20\lg\{f\} + 147.55 - P_\epsilon}{10}} - \Delta H^2}. \tag{6.15}$$

假设从时间 $t = 0$ 开始进行充电，无人机的速度为 V，则在时间 t，无人机与基站之间的距离为：

$$d_t = \sqrt{(x_{start} + Vt - x_{bs})^2 + \Delta H^2} \tag{6.16}$$

则 t 时刻的充电功率为：

$$P_{charge}^t = \Omega - 20\lg(d_t) \tag{6.17}$$

其中 $\Omega = P_t + G_t + G_{uav} - 20\lg(f) + 147.55$，将无人机从 x_{start} 位置飞到基站顶部所花费的时间表示为 $T_{\text{charge}}^{fly} = (x_{bs} - x_{start})/V$，则无人机在此充电阶段接收的总能量计算为 [164]:

$$\begin{aligned} E_{fly} =& \eta \int_0^{T_{\text{charge}}^{fly}} 10^{\frac{P_{charge}^t}{10}} dt \\ =& \frac{2\eta 10^{\frac{\Omega}{10}}}{\sqrt{4A_2C_2 - B_2^2}} \arctan \frac{B_2 + 2C_2 T_{\text{charge}}^{fly}}{\sqrt{4A_2C_2 - B_2^2}} \\ & - \frac{2\eta 10^{\frac{\Omega}{10}}}{\sqrt{4A_2C_2 - B_2^2}} \arctan \frac{B_2}{\sqrt{4A_2C_2 - B_2^2}} \end{aligned} \tag{6.18}$$

其中 $A_2 = (x_{\text{start}} - x_{bs})^2 + \Delta H^2$、$B_2 = 2V(x_{\text{start}} - x_{bs})$，$C_2 = V^2$。

将无人机到达 x_{start} 位置时的剩余能量记为 E_r，根据式 (6.13)，则无人机的悬停时间计算为:

$$T^{\text{hover}} = \frac{E_{full} - E_r - E_{fly}}{\eta 10^{\frac{P_{\text{uav-r}}}{10}} - P(0)} \tag{6.19}$$

6.2.3 系统优化模型

问题的目标是最小化无人机的总任务时间，包括飞行时间和充电时间。令 d_{ij} 代表节点 i 到节点 j 之间的距离，索引 0 的节点为基站，假设无人机返回基站的次数为 K，即无人机路径由 K 个哈密顿回路组成，令 x_{ijk} 为决策变量，$x_{ijk} = 1$ 代表节点 i 和 j 在第 k 个回路中相连，该优化问题建模为:

$$\min \sum_{\substack{i,j=0 \\ i \neq j}}^{N} \sum_{k=1}^{K} d_{ij} x_{ijk}/V + \sum_{i=0}^{K} T_i^{\text{hover}} \tag{6.20}$$

受制于

$$\sum_{k=1}^{K} \sum_{j=1}^{N} x_{0jk} = \sum_{k=1}^{K} \sum_{j=1}^{N} x_{j0k} = K \tag{6.21}$$

$$\sum_{i=1}^{N} x_{ijk} = \sum_{i=1}^{N} x_{jik}, (k \in K, \forall j = 1, 2, \ldots, N) \tag{6.22}$$

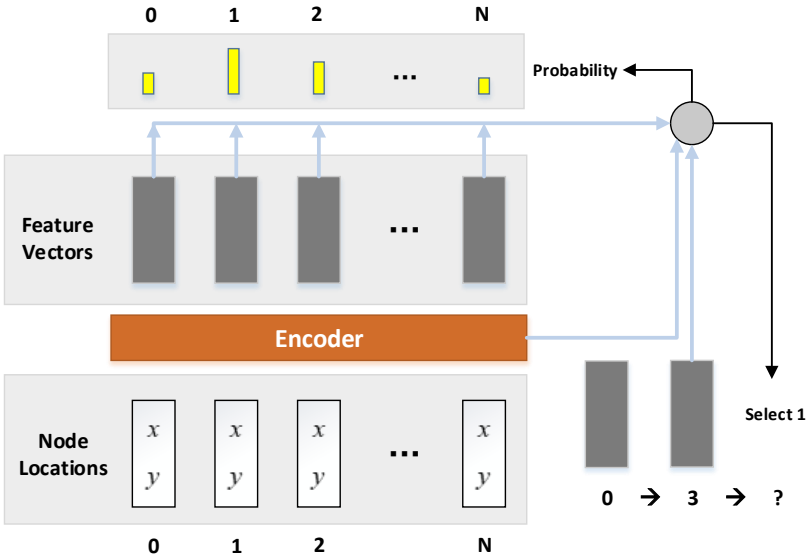


图 6.2　无人机路径规划深度神经网络模型结构

$$\sum_{i=1}^{K}\sum_{i=0}^{N} x_{ijk} = 1, (j = 1, 2, \ldots, N) \tag{6.23}$$

$$P(V)\sum_{i=0}^{N}\sum_{j=0}^{N} \mathrm{x}_{ijk} d_{ij}/V \leq (1 - SOC_{min})E_{full}, (k \in K) \tag{6.24}$$

约束条件式 (6.21) 表示进入基站的无人机数量与离开基站的无人机数量相同；约束式 (6.22) 表示进出每个目标任务节点的无人机数量相同；约束式 (6.23) 表示每个目标任务节点只被访问一次；约束式 (6.24) 为能量约束，表示无人机荷电状态需要一直保持在 SOC_{min} 水平之上。

6.3　深度神经网络模型

本章所提模型由两部分组成：编码器和解码器。编码器根据节点的二维坐标计算节点的 d_h 维特征向量 $\mathbf{h}_i$。解码器以自回归方式构建无人机路径，在每一步，解码器根据由节点的特征向量 $\mathbf{h}_i$ 和“上下文向量”计算得到的节点选择概率选择下一步前往的目标节点，模型架构如图 6.2 所示，黄色代表计算得到的节点概率。

6.3.1　编码器

类似于第三章，编码器采用多头注意力（MHA）来计算每个节点 i 的特征向量 $\mathbf{h}_i$[106]。

首先，给定所有节点的二维坐标 $\mathbf{x}$，节点的初始 d_h 维特征向量计算如下：

$$\mathbf{h}_i^{(0)} = W^{\mathbf{x}}\mathbf{h}_i^{(0)} + \mathbf{b}^{\mathbf{x}}. \tag{6.25}$$

其中 $W^{\mathbf{x}}$ 和 $\mathbf{b}^{\mathbf{x}}$ 是模型参数，给定节点的初始特征向量 $\mathbf{h}^{(0)}$，MHA 进一步将 $\mathbf{h}^{(0)}$ 转换为节点最终的特征向量。

自注意力是 MHA 的基本算子，它计算每个节点对输入序列中所有其他节点的注意力值，因此，得到的每个节点的注意力值（特征向量）不仅存储了节点自身的信息，还存储了其与其他节点的关系。节点 i 的注意力值由其 *query* 和所有节点的 *key-value* 对计算得出，每个节点 i 的 *query*、*key* 和 *value* 都是其初始特征向量 $\mathbf{h}_i^{(0)}$ 的线性投影：

$$\mathbf{q}_i = W^Q\mathbf{h}_i^{(0)}, \quad \mathbf{k}_i = W^K\mathbf{h}_i^{(0)}, \quad \mathbf{v}_i = W^V\mathbf{h}_i^{(0)}. \tag{6.26}$$

矩阵化形式表示为：

$$\mathbf{Q} = W^Q\mathbf{X}, \quad \mathbf{K} = W^K\mathbf{X}, \quad \mathbf{V} = W^V\mathbf{X}. \tag{6.27}$$

假设输入序列中有 N 个节点，节点 i 的注意力值是所有 N 个节点的 *values* 的加权和，权重是通过节点 i 的 *query* 和所有 N 个节点的 *keys* 计算得到，即计算节点 i 的“查询”和所有“键”的匹配度，该权重计算为 $\mathbf{q}_i\mathbf{K}^T$。因此，节点 i 的注意力值为 $\mathbf{q}_i\mathbf{K}^T\mathbf{V}$，softmax 函数用于对权重进行归一化。因此，所有节点的注意力值计算为：

$$\text{Attention}\,(Q,K,V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_h}}\right)V \tag{6.28}$$

其中 $1/d_h$ 是缩放因子，进一步地，MHA 将 $\mathbf{h}^{(0)}$ 分割成 M 个子向量，并进行 M 次上述自注意力计算，从而得到 M 个 d_h/M 维度的向量，最后将它们进行拼接，并进行线性映射，从而得到最终的 d_h 维度的特征向量 $\mathbf{h}^{(N)}$，因此 MHA 可以提取更多维度、更丰富的节点特征信息。

编码器的模型架构与第三章相同，由 N 个顺序连接的注意力层组成，每层由一个 MHA 层和一个全连接层组成，将节点的初始特征向量 $\mathbf{h}^{(0)}$ 进行 N 层处理，得到所有节点的最终特征向量 $\mathbf{h}^{(N)}$。

6.3.2 解码器

解码器中，选择各个节点的概率根据其特征向量 $\mathbf{h}^{(N)}$ 和当前的“上下文向量(Context)”计算。上下文向量表示，在第 t 步，环境的当前状态，包括在 $t' < t$ 生

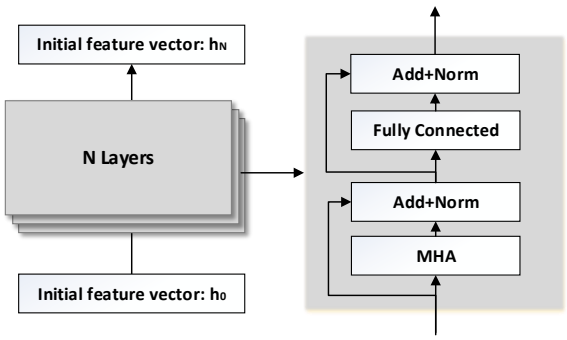


图 6.3　编码器结构

成的部分路径和无人机的当前的荷电状态，按照以下方式进行计算。

当无人机离开基站时，无人机的 SOC 初始化为 $SOC_t = 1$，后续更新如下：

$$SOC_{t+1} = \begin{cases} SOC_t - P(V)d(\pi_t, \pi_{t-1})/V/E_{full} & \pi_t \neq 0 \\ 1 & \pi_t = 0 \end{cases} \tag{6.29}$$

其中索引 $\pi_t = 0$ 的节点代表基站，$d(*,*)$ 代表两个节点之间的距离。

上下文向量按照如下方式设计：

$$\mathbf{d}_t = \begin{cases} \left[\overline{\mathbf{h}}^{(N)}, \mathbf{h}_{\pi_{t-1}}^{(N)}, SOC_t\right] & t > 1 \\ \left[\overline{\mathbf{h}}^{(N)}, \mathbf{h}_0^{(N)}, SOC_t\right] & t = 1 \end{cases} \tag{6.30}$$

其中 $\mathbf{h}_{\pi_{t-1}}^{(N)}$ 是在步骤 $t-1$ 中选择的节点的特征向量。$\overline{\mathbf{h}}^{(N)}$ 是所有节点特征向量 $\mathbf{h}^{(N)}$ 的平均值，表征问题的整体状态。

上下文向量可以看作在第 t 步解码过程的 *query* 值，同时，每个节点都有一个由其特征向量 $\mathbf{h}^{(N)}$ 表示的 key 值，因此，可以在所有 $key_i, i = 1 \ldots N$ 中，选择与当前 *query* 最匹配的一个节点作为下一个访问节点，该匹配度采用注意力机制进行计算。首先对 *query* 和 *key* 进行计算：

$$\mathbf{q}_t = W^{Q'}\mathbf{d}_t, \quad \mathbf{k}_i = W^{K'}\mathbf{h}_i^{(N)} \tag{6.31}$$

节点的选择概率计算为：

$$u_i = \begin{cases} C \cdot \tanh\left(\frac{\mathbf{q}_t^T \mathbf{k}_i}{\sqrt{d_h}}\right) & \text{if } i \text{ is not masked} \\ -\infty & \text{otherwise.} \end{cases} \tag{6.32}$$

其中 C 对概率值进行裁剪，softmax 函数对 u_i 进行归一化。

进一步设计屏蔽机制（Mask）以处理约束，即一个节点如果不可选则被屏蔽。考虑以下几种情况：1）如果节点 $i>0$ 在时间 $t'<t$ 已经被访问过，则它被屏蔽；2）如果在上一步 $t-1$ 节点 $i=0$ 被访问，即基站被访问，则在第 t 步基站被屏蔽，代表无人机充满电则应该离开基站；3）以下情况，节点 $i>0$ 将被屏蔽：

$$P(V)\left(d(\pi_{t-1},i)+d(i,0)\right)/V>(SOC_{t-1}-SOC_{min})E_{full} \tag{6.33}$$

即剩余电量应该足够无人机前往节点 i 并返回基站。

解码过程在图 6.2 中进行了可视化。在图 6.2 中，节点 0 和 3 已经被访问，任务是确定下一个目标节点，通过当前的上下文向量和来自编码器的特征向量计算各个节点的概率值，可通过贪婪策略选择概率最大的节点 1，该过程不断循环，直到所有节点都被访问并且无人机返回基站。

6.4 模型训练方法

采用深度强化学习方法训练该深度神经网络模型。给定一个问题 s，上节建立的深度神经网络模型输出概率分布 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$，通过对该概率分布进行采样即可生成解：

$$p_\theta(\boldsymbol{\pi}|s)=\prod_{t=1}^{k}p_{\boldsymbol{\theta}}\left(\pi_t|s,\boldsymbol{\pi}_{1\sim t-1}\right),k\leq N. \tag{6.34}$$

模型根据问题 s 的特征、以及问题当前的状态 $\boldsymbol{\pi}_{1\sim t-1}$ 生成选择各节点的的概率 π_t，模型由神经网络参数 $\boldsymbol{\theta}$ 进行参数化，通过对参数进行寻优，可生成无人机的近似最优路径。

参数优化的目标是最小化无人机消耗的总时间 $T(\pi)$，由于总时间 $T(\pi)$ 在无人机任务结束时才可计算得到，因此采用基于蒙特卡洛的 REINFORCE 算法进行训练，给定一个问题实例 s 和模型输出的概率分布 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$，按照以下公式对模型参数进行更新：

$$\begin{aligned}\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})&=\mathbf{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)}\left[\nabla\log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)T(\boldsymbol{\pi})\right]\\ \boldsymbol{\theta}&\leftarrow\boldsymbol{\theta}+\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta}).\end{aligned} \tag{6.35}$$

进一步，引入基线值 $b(s)$ 来改进 式 (6.35)：

$$\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})=\mathbf{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)}\left[\nabla\log p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)(T(\boldsymbol{\pi})-b(s))\right]. \tag{6.36}$$

其中 $b(s)$ 表示平均性能，如果 $T(\boldsymbol{\pi})-b(s)<0$，则对该策略 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$ 进行正向激励，因为它的性能优于平均性能 $b(s)$，反之亦然。本章采用文献[106] 中提出的 greedy rollout baseline 方法来计算 $b(s)$，将训练期间表现最好的模型存储为基准模型 $\boldsymbol{\theta}^*$，通过将 s 输入到基准模型 $\boldsymbol{\theta}^*$ 中可计算得到 $p_{\boldsymbol{\theta}^*}(\boldsymbol{\pi}|s)$，基于贪婪策略对 $\boldsymbol{\pi}^*$ 进行采样，可以得到无人机路径，该路径的总时间 $T(\boldsymbol{\pi}^*)$ 作为 $b(s)$ 值，因此，$b(s)$ 可以代表基线性能，从而指导优化的方向。

训练流程如算法 6.1 所示，首先随机生成训练样本 s，即一组目标任务节点的坐标，然后采用模型 $\boldsymbol{\theta}$ 产生无人机路径 $\boldsymbol{\pi}$，从而计算得到对应的任务时间 $T(\boldsymbol{\pi})$，根据式 (6.36) ，可以通过最小化 $T(\boldsymbol{\pi})$ 实现对 $\boldsymbol{\theta}$ 的更新。

算法 6.1 REINFORCE 强化学习训练方法

算法输入： 训练轮数 n_e，每次训练样本数 B，每轮训练步数 n_s
算法输出： 模型最优参数 $\boldsymbol{\theta}$

初始化策略网络参数 $\boldsymbol{\theta}$ 和基准策略的神经网络参数 $\boldsymbol{\theta}^*$
for $epoch \leftarrow 1:n_e$ **do**
    for $step \leftarrow 1:n_s$ **do**
        生成 B 个样本 s_i。
        $\boldsymbol{\pi}_i \leftarrow p_{\boldsymbol{\theta}}(s_i)$. 对策略网络的模型输出概率分布进行采样，得到样本中问题的无人机路径。
        $\boldsymbol{\pi}_i^* \leftarrow p_{\boldsymbol{\theta}^*}(s_i)$. 对基准策略模型的输出概率进行贪婪动作选择，构造得到样本中问题的无人机路径。
        $\nabla\mathcal{L} \leftarrow \sum_{i=1}^{B}\left(T\left(\boldsymbol{\pi}_i\right)-T\left(\boldsymbol{\pi}_i^*\right)\right)\nabla_{\boldsymbol{\theta}}\log p_{\boldsymbol{\theta}}\left(\boldsymbol{\pi}_i\right)$
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}+\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$. 更新参数 $\boldsymbol{\theta}$
    end for
    if $p_{\boldsymbol{\theta}}$ 表现优于 $p_{\boldsymbol{\theta}^*}$ **then**
        $\boldsymbol{\theta}^* \leftarrow \boldsymbol{\theta}$
    end if
end for

6.5 实验结果与讨论

6.5.1 实验设置

无人机的物理参数见表 6.1 ，参数取自文献 [164]，本文中无人机的速度设置为 $10m/s$。根据式 (6.1) ，无人机的飞行功率为 $P(V)=62.49W$，悬停功率为 $P(0)=82.46W$，SOC_{min} 设置为 20%，无人机电池容量为 $3830mAh$（$43.8Wh$），电压为 $11.4V$。对于无线能量传输设备的参数，文中设置 $P_t=58$dBW、$G_t=58$dBi [164]，$f=915MHz$。此外，RF-DC 转换效率为 $\eta=0.6$ [164]，无人机的充电阈

表 6.1　无人机物理参数

符号	参数	值
m	机身质量	1.0 kg
ρ	空气密度	1.225 kg/m^3
b	叶片数量	4
R	旋翼半径	0.25 m
A	旋翼盘面积	0.19634 m^2
c	叶片长度	0.0196 m
s	旋翼实度	0.0998
δ	剖面阻力系数	0.012
ω	叶片角速度	400 rad/s
k	电感应功率增量修正系数	0.05
U_{tip}	转子叶片的叶尖速度	100.0 m/s
v_0	旋翼平均诱导速度	4.5135 m/s
S_{FP}	机身等效平板面积	0.0079 m^2
d_0	机身阻力比	0.4030
P_0	叶片功率	36.01 W
P_i	感性负载功率	46.44 W

值功率设置为 $P_\epsilon = 17$dBW。

表 6.2 列出了模型训练的超参数，批量大小为 256、节点向量维度为 128、编码器 - 解码器的隐层向量维度为 128。均使用 Adam 优化器 [109] 以 10^{-4} 的固定学习率进行模型训练。

给定模型输出的概率分布 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$，采用以下三种策略来构造解：

（1）贪婪策略（Greedy）。在解码过程，每一步均从 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$ 中选择概率最大的节点，构造得到一个解。

（2）采样策略（Sample）。在每一步节点选择，均从概率分布 $p_{\boldsymbol{\theta}}(\boldsymbol{\pi}|s)$ 中进行随机采样，采样 $N_{sample} = 1280$ 次得到 N_{sample} 个解，选择其中最好的一个作为输出解。

（3）波束搜索（Beam Search, BS）。本章采用宽为度 $N_{beam} = 1000$ 的波束搜索构造解。该方法类似于广度优先搜索，但是搜索过程中只保留前 N_{beam} 个最好的部分解，只对该 N_{beam} 个节点进行探索。当 N_{beam} 为无穷大时，波束搜索退化为广度优先搜索。

本章提出的深度强化学习方法与以下传统方法进行实验比较：

（1）Google OR-tools [166]。谷歌开发的高性能优化求解器，具有多种元启发式的组合优化算法，本章采用算法库中提供的路径优化算法模板对问题进行建模，

表 6.2　实验参数设置

参数	值	参数	值
批量大小	256	隐层维度	128
训练轮次	100	多头注意力特征个数	8
每轮次的训练集规模	320000	编码器层数	3
优化器	Adam	学习率	1e-4

采用推荐参数进行求解。

（2）Clarke-Wright savings（CW）启发式算法 [167]。该方法是解决路径规划问题的经典启发式算法之一，采用该开源库[1]的参数和代码对问题进行求解。

（3）粒子群优化算法（PSO）[168]。

（4）蚁群优化算法 (ACO) [169]。

（5）自适应大邻域搜索（ALNS）[170]。3)、4)、5) 的算法参数和代码都取自该开源库[2]。

Google OR-tools 可以超越目前的大多数方法，是多个优化竞赛（例如 MiniZinc Challenge 2020、2021）中的冠军算法，因此，虽然仍有大量其他方法可以求解该路径规划问题，本章不再穷举各类启发式算法进行对比，而主要采用 Google OR-tools 和上述代表性的传统算法作为对比算法。

所有算法均在 Python 中实现，在参数为 GTX 2080Ti GPU、Intel 64GB 16-Core i7-9800X CPU 的平台上进行实验，采用 100 个随机生成的问题实例作为测试集，采用平均目标函数和平均运行时间作为性能指标。

6.5.2　实验结果

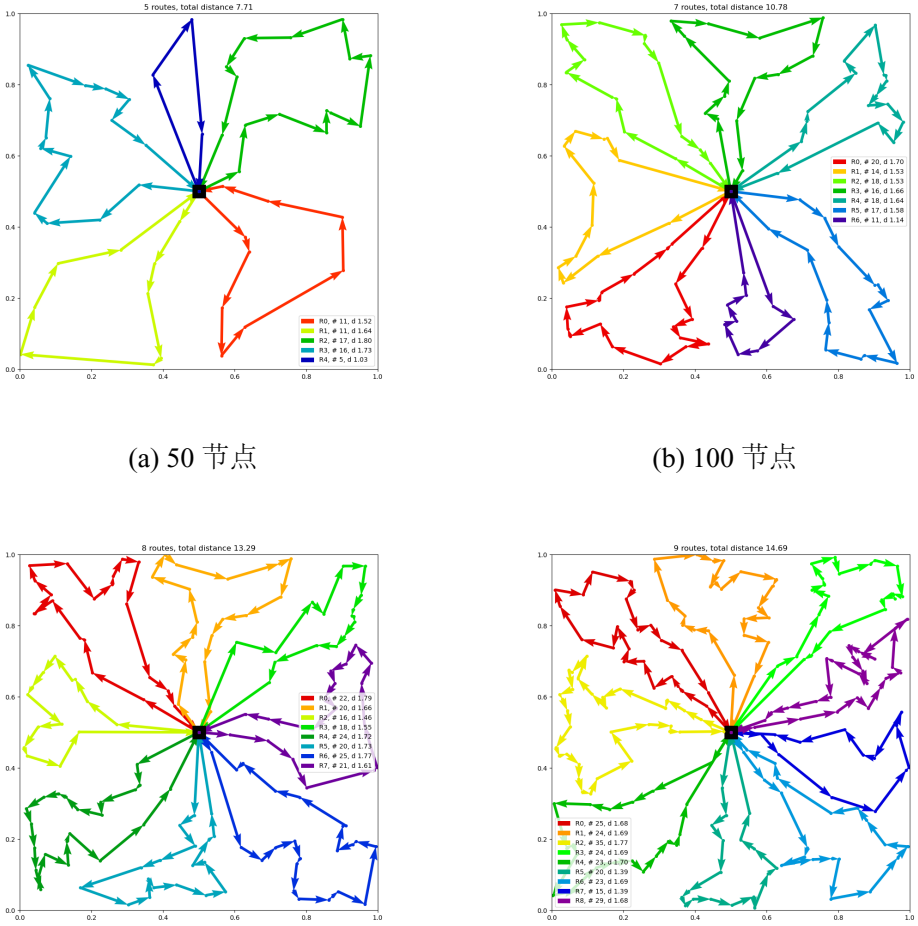
图 6.4 可视化了基于采样策略的深度强化学习（DRL）模型在 50、100、150、200 节点问题上的解，可以看出 DRL 方法可以有效地求解该问题，得到的无人机路径无交叉。

表 6.3 列出了 DRL 方法和对比方法在 20、50 和 100 个节点的小规模问题上的数值实验结果。Cost 是无人机执行任务的平均总时间（以小时为单位），Time 是算法的平均求解时间。并分别列出了贪婪、采样和波束搜索策略的实验结果，给出了 DRL 方法与性能最佳的方法之间的优化性能差距（Gap），并且给出了 DRL 方法相对于 Google OR-Tools 的求解速度提升（Speed*）。

从实验结果可见，与贪婪策略相比，采样策略和波束搜索策略可以有效提高解的质量，同时增加了算法运行时间，但运行时间总在一秒钟之内，可以看出本

[1]https://github.com/ishelo/VRP-CW

[2]https://github.com/PariseC/Algorithms_for_solving_VRP



(a) 50 节点　　(b) 100 节点

(c) 150 节点　　(d) 200 节点

图 6.4　DRL 方法在不同规模问题上构造得到的解

章所提算法具有较好的在线优化能力。

由表 6.3 可见，DRL 方法在 20 和 50 节点问题上的优化性能和求解速度方面都优于 Google OR-Tools，在 100 个节点问题上，DRL 方法的性能比 Google OR-Tools 稍差（0.7%），但运行速度比它快五倍以上。另外，DRL 方法和 Google OR-Tools 在所有问题上都优于对比算法 CW、PSO、ACO 和 ALNS。

表 6.4 给出了 DRL 算法和对比方法在 150、200 节点问题上的实验结果。针对该大规模问题，所有求解器的运行时间都限制在 1000 秒内，并且给出了各个算法在 1000 秒内可以求解得到的问题个数。可以看到 Google OR-Tools 无法在 1000 秒内解决所有的 150 节点、200 节点的测试问题。在 150 节点问题上，DRL 方法的性能比 OR-Tools 差 1.1%，但它的运行速度比 OR-Tools 快 592 倍，并且，DRL 方法构造得到解的平均成本优于其他四种比较方法。Clarke-Wright 是针对路径

表 6.3　小规模问题数值实验结果

	20 Nodes		50 Nodes		100 Nodes	
	Cost/h	Time/s	Cost/h	Time/s	Cost/h	Time/s
OR-Tools	2.378	0.054	3.072	0.374	**3.900**	**1.71**
Clarke-Wright	2.428	0.001	3.169	0.009	4.069	0.055
PSO	2.386	16.3	3.286	40.66	4.506	82.4
ACO	2.456	2.86	3.397	11.41	4.589	35.49
ALNS	2.392	12.1	3.331	24.81	4.458	142
Our(Greedy)	2.392	0.002	3.142	0.002	4.000	0.006
Our(Sample)	2.375	0.039	3.058	0.116	3.928	0.304
Our(BS)	**2.367**	**0.013**	**3.050**	**0.056**	3.936	0.209
Gap/speed*	best	4.2	best	6.7	0.7%	5.6

表 6.4　大规模问题数值实验结果

	150 Nodes			200 Nodes		
	Cost/h	Time/s	Solved	Cost/h	Time/s	Solved
OR-Tools	**4.492**	**341.6**	69	-	1000	0
Clarke-Wright	4.733	0.154	100	5.300	0.343	100
PSO	5.464	126.9	100	6.250	172.8	100
ACO	5.494	77.7	100	6.331	148.7	100
ALNS	5.506	295.8	100	5.728	1038	100
Our(Greedy)	4.631	0.004	100	5.217	0.006	100
Our(Sample)	4.542	0.577	100	**5.097**	**0.94**	100
Our(BS)	4.564	0.479	100	5.142	0.253	100
Gap/speed*	1.1%	592.0		best	-	

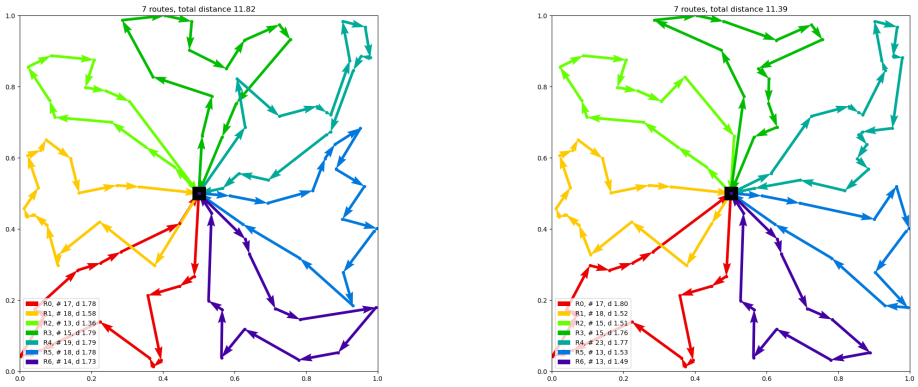
优化问题专门设计的启发式方法，始终可以在合理的运行时间内得到较好的解，然而基于贪婪策略的 DRL 方法在小规模归问题上，可以在几乎相同的求解时间上超越 Clarke-Wright 方法，在大规模问题上，在求解速度和优化性能上均优于 Clarke-Wright。此外，在 200 节点的测试问题上，基于贪婪策略的 DRL 方法比所有对比算法均快 50 倍以上，同时可以得到最好的优化性能。

从数值实验结果中可以得到如下结论：

（1）在优化性能方面，Google OR-Tools 和 DRL 方法在所有测试问题上都优于 CW、PSO、ACO 和 ALNS；

（2）Google OR-Tools 在五个测试问题中的两个问题上（100 和 150 节点问题）上优于 DRL 方法，但差距均在 1% 左右，而 DRL 方法在求解速度上远超 Google OR-Tools；

（3）在 20 节点问题上，Clarke-Wright 方法与 DRL 方法具有相似的优化性能和求解速度，DRL 方法在其他规模的测试问题上优于 Clarke-Wright 方法。DRL 方法在小规模问题上的运行速度比其他基准算法快 4 倍以上，在大规模问题上的运行速度比传统方法快 50 倍以上，同时取得了最好的优化性能。



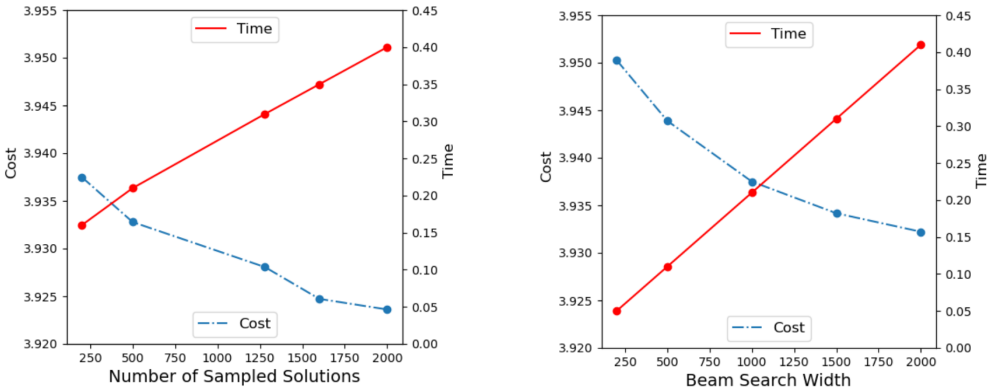
(a) 贪婪策略　　(b) 波束搜索

图 6.5　基于贪婪策略和波束搜索构造得到的无人机路径

进一步对构造无人机路径的不同策略进行分析。图 6.5 给出了 DRL 方法在 100 节点问题上基于贪婪策略和波束搜索构造得到的无人机路径。可见，通过波束搜索使无人机路径得到了明显的改善，但是会增加求解时间。图 6.6 进一步分析了具有不同参数的采样策略和波束搜索的性能。可以观察到，采样策略和波束搜索的运行时间分别随着采样解的数量和波束搜索宽度的增加而线性增加，同时解的质量会相应的提高，但是解质量提高的幅度逐渐变缓，因此，可以根据计算资源和求解实时性要求选择不同的参数，从而在不同求解时间下得到不同质量的无人机路径。

6.6　本章小结

本章将深度强化学习方法应用于考虑无线能量传输的无人机路径优化问题上。通过使用深度神经网络对该问题进行建模，通过深度强化学习方法对模型进行离线训练，从而利用该模型实现无人机路径的在线优化，克服了基于迭代搜索的传



(a) 采样策略　　(b) 波束搜索策略

图 6.6　具有不同参数的采样策略和波束搜索的性能

统启发式方法的局限性，显著减少了计算时间。通过于 Google OR-Tools、粒子群、自适应大邻域搜索等方法的对比，验证了本章所提深度强化学习方法能够实现该问题的精准快速求解。

第七章　总结与展望

随着现代网络、通信、智能化技术的飞速发展，在各个领域中需要解决的组合优化问题规模不断扩大，并且对于实时求解、在线优化的需求越来越高，例如复杂军事系统中的作战调度问题、通信网络等领域的资源分配问题以及外卖、网约车等领域的订单派送问题等，随着问题规模的扩大和优化实时性要求的提高，对优化引擎的快速求解能力提出了更高的要求，但是传统的组合优化方法很难胜任。近年来，随着人工智能技术的迅猛发展，人工智能技术展示了其强大的学习能力与优化决策能力，为实现组合优化问题更加高效的求解提供了新的可能性。本文瞄准实际应用中对于算法实时性的要求，对基于深度强化学习的组合优化方法进行研究，对组合优化的单目标方法、多目标方法、精确方法等方面进行了全方位的研究，探索人工智能技术解决传统组合优化问题的新方法、新概念、新技术，提出采用深度强化学习求解组合优化问题的系统全面的解决方案。

7.1　总结

本文重点研究了基于深度强化学习的组合优化框架、单目标组合优化、多目标组合优化以及组合优化精确方法。

（1）论文首先对近些年利用深度强化学习方法解决组合优化问题的相关理论方法与应用研究进行了综述研究，对其基本原理、相关方法、应用研究进行系统地总结和分析，对当前方法进行了分类总结，对不同种类方法的优势和缺陷进行了分析，为该领域的研究提供了系统全面的综述介绍。相关工作已发表于自动化学报期刊。

（2）作为新兴起的新方法，基于深度强化学习的组合优化方法涉及到大量新技术和新概念，与传统优化方法存在较大差距，因此对基于深度强化学习求解组合优化问题的科学原理和求解框架进行研究。本文将深度强化学习方法求解组合优化问题的流程分为三步：根据问题类型进行模型选择；设计深度神经网络对组合优化问题进行建模；采用强化学习方法对深度神经网络参数进行学习。并针对各个步骤，对涉及到的组合优化问题、深度神经网络模型、深度强化学习方法的相关理论进行了阐述，详细研究了组合优化问题建模需要采用的深度神经网络模型，理清了基于深度强化学习求解组合优化问题的科学原理，提出了基于深度强化学习的组合优化方法的流程框架和求解范式。

（3）进一步地，针对复杂单目标组合优化问题，本文提出了一种基于深度强化学习的覆盖旅行商组合优化方法，对具有动态特性、解序列长度不定的单目标覆盖旅行商问题进行研究，克服了当前人工智能模型在处理大规模、动态复杂特

性问题上的局限，采用多头注意力机制处理问题的静态特征，设计了一种动态嵌入方法处理问题的动态特性，所提方法相对于传统启发式方法，在达到相同优化水平的前提下，能够获得 10 倍以上求解速度的提升，并且模型一旦训练好，能够泛化到不同类型的问题上，而不需要对模型进行重新训练。

（4）针对多目标组合优化问题，本文提出了一种基于深度强化学习的多目标旅行商组合优化方法，当前采用深度学习技术求解组合优化问题的方法均针对单目标组合优化问题，在多目标优化上仍然存在空白，并且存在很大挑战，本文首次提出了一种采用深度学习方法求解传统多目标组合优化问题的方法，采用端到端方式直接输出问题的帕累托前沿，具体地，通过分解策略和基于邻域的参数迁移策略实现对多目标问题的建模，所提方法在多达 500 规模、多达 5 个目标的旅行商问题上取得了超越传统方法的性能，并且具有快速的求解能力。

（5）针对能够得到组合优化问题理论最优解的分支定界法。本文提出了一种基于深度强化学习的分支定界组合优化方法，提高了传统组合优化方法的求解速度，具体地，设计了一种图指针神经网络模型对分支定界法的变量选择策略进行建模，设计了全局特征和历史特征以提高模型的性能，并设计了一种基于 top-k KL 散度的模仿学习方法对模型进行训练，从而实现了比专业优化求解器求解速度快 40% 的优化性能，同时超越了当前基于机器学习改进的精确算法，在求解速度和模型准确率上均获得了提升。

（6）最后，针对实际应用问题，本文利用深度强化学习方法解决了考虑无线能量传输的无人机路径优化问题，构建了基于多头注意力机制的深度神经网络模型，通过强化学习对模型进行离线训练，实现了无人机路径的实时在线优化，该方法相对于 Google OR-tools、粒子群、自适应大邻域搜索等传统方法，在不同规模问题上实现了 4 倍到 500 倍求解速度的提升。

通过以上内容，对深度强化学习求解传统组合优化问题这个领域进行了全面的研究，实现对组合优化单目标、多目标、近似算法、精确算法的全方位覆盖，提出了在不同应用场景下求解组合优化问题的系统的解决方案。

7.2 展望

本文重点探索和研究了基于深度强化学习的组合优化方法，对单目标组合优化、多目标组合优化、组合优化精确方法等方面进行了研究，但是在许多优化问题中，涉及到多种复杂巨系统，面临的环境具有复杂特点，问题规模和约束更大更多，仍存在很多亟待解决的关键技术。接下来，给出几个后续的研究展望：

（1）在模型方面。该类方法的显著缺陷就是无法保证优化解的质量，与传统方法仍然存在差距，直接采用深度神经网络模型输出的解可能会出现较差的情况，

一些情况下需要进一步通过波束搜索、局部搜索、采样策略等方式进一步提升解的质量，这说明当前的模型仍然有很大的提升空间，未来需要进一步对求解组合优化问题的深度神经网络模型进行研究，如何有效结合图神经网络和指针网络模型是一个较好的研究方向。

（2）在研究对象方面。本文虽然对具有复杂特性的单目标组合优化问题进行了研究，但实际中的组合优化问题通常具有更加复杂的特性，如超多目标、超多约束、非静态等复杂特性，当前方法对该类问题进行求解时仍然存在建模困难的问题。未来基于深度强化学习方法对多目标、约束优化、动态优化问题进行研究是一个重要的研究方向。

（3）在深度强化学习训练算法方面。当前模型的训练大多采用 REINFORCE、DQN 等传统训练算法，具有采样效率低、收敛慢等缺陷，如何根据组合优化问题的特性设计更加高效的强化学习训练算法也是一个未来需要着重研究的内容。

（4）最后，由于基于人工智能技术求解传统优化问题是一个新兴的研究领域，当前的应用仍然较少，如何利用基于深度强化学习的组合优化方法来解决工程实际中的在线调度优化问题将会成为未来重要的研究方向。

参考文献

[1] Papadimitriou C, Steiglitz K. Combinatorial optimization: Algorithms and complexity [M]. Courier Corporation, 1998.

[2] Lawler E, Wood D. Branch-and-bound methods: A survey [J]. Operations Research. 1966, 14 (4): 699–719.

[3] Bertsekas D. Dynamic programming and optimal control. Athena Scientific Belmont, MA, 1995.

[4] Sniedovich M. Dynamic programming: foundations and principles [M]. CRC Press, 2010.

[5] Festa P. A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems [C]. In 2014 16th International Conference on Transparent Optical Networks (ICTON. 2014: 1–20.

[6] Williamson D, Shmoys D. The design of approximation algorithms [M]. Cambridge University Press, 2011.

[7] Vazirani V V. Approximation algorithms [M]. Springer, 2001.

[8] Teoh E, Tang H, Tan K. A columnar competitive model with simulated annealing for solving combinatorial optimization problems [C]. In The 2006 IEEE International Joint Conference on Neural Network Proceedings. 2006: 3254–3259.

[9] PJM V L, EHL A, JK L. Job shop scheduling by simulated annealing [J]. Operations research. 1992, 40 (1): 113–125.

[10] J W, M L. Solving the multiple-machine weighted flow time problem using tabu search [J]. IIE transactions. 1993, 25 (2): 121–128.

[11] Basu S. Tabu search implementation on traveling salesman problem and its variations: a literature survey [J]. American Journal of Operations Research. 2012, 2 (2): 163–173.

[12] Halim A, Ismail I. Combinatorial optimization: comparison of heuristic algorithms in travelling salesman problem [J]. Archives of Computational Methods in Engineering. 2019, 26 (2): 367–380.

[13] Rezoug A, Bader-El-Den M, Boughaci D. Guided genetic algorithm for the multidimensional knapsack problem [J]. Memetic Computing. 2018, 10 (1): 29–42.

[14] Lin B, Sun X, Salous S. Solving travelling salesman problem with an improved hybrid genetic algorithm [J]. Journal of Computer and Communications. 2016, 4 (15):

98−106.

[15] Prado R, Silva R, Guimarães F, et al. Using differential evolution for combinatorial optimization: A general approach [C]. In 2010 IEEE International Conference on Systems, Man and Cybernetics. 2010: 11−18.

[16] Onwubolu G, Davendra D. Differential evolution: A handbook for global permutation-based combinatorial optimization [M]. Springer Science & Business Media, 2009.

[17] Deng W, Xu J, Zhao H. An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem [J]. IEEE Access. 2019, 7: 20281−20292.

[18] Ramadhani T, Hertono G, Handari B. An Ant Colony Optimization algorithm for solving the fixed destination multi-depot multiple traveling salesman problem with non-random parameters [C]. In AIP Conference Proceedings, AIP Publishing LLC. 2017: 30123.

[19] Zhong Y, Lin J, Wang L, et al. Discrete comprehensive learning particle swarm optimization algorithm with Metropolis acceptance criterion for traveling salesman problem [J]. Swarm and Evolutionary Computation. 2018, 42: 77−88.

[20] Nouiri M, Bekrar A, Jemai A, et al. An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem [J]. Journal of Intelligent Manufacturing. 2018, 29 (3): 603−615.

[21] Lourenço H, Martin O, Stützle T. Iterated local search: Framework and applications [C]. In Handbook of Metaheuristics. 2019: 129−168.

[22] Grasas A, Juan A, Lourenço H. SimILS: a simulation-based extension of the iterated local search metaheuristic for stochastic combinatorial optimization [J]. Journal of Simulation. 2016, 10 (1): 69−77.

[23] Zhang G, Zhang L, Song X, et al. A variable neighborhood search based genetic algorithm for flexible job shop scheduling problem [J]. Cluster Computing. 2019, 22 (5): 11561−11572.

[24] Hore S, Chatterjee A, Dewanji A. Improving variable neighborhood search to solve the traveling salesman problem [J]. Applied Soft Computing. 2018, 68: 83−91.

[25] Silver D, Schrittwieser J, Simonyan K. Mastering the game of go without human knowledge [J]. Nature. 2017, 550 (7676): 354−359.

[26] Mnih V, Kavukcuoglu K, Silver D. Human-level control through deep reinforcement learning [J]. Nature. 2015, 518 (7540): 529−533.

[27] Hopfield J，Tank D．Neural computation of decisions in optimization problems [J]．Biological Cybernetics．1985，52 (3)：141−152．

[28] Smith K. Neural networks for combinatorial optimization: a review of more than a decade of research [J]. INFORMS Journal on Computing. 1999, 11 (1): 15−34.

[29] Vinyals O, Fortunato M, Jaitly N. Pointer networks [C]. In Advances in Neural Information Processing Systems. 2015.

[30] Bello I, Pham H, V L Q, et al. Neural combinatorial optimization with reinforcement learning [C]. In 5th International Conference on Learning Representations, ICLR 2017 - Workshop Track Proceedings. 2017.

[31] Nazari M, Oroojlooy A, Takac M, et al. Reinforcement learning for solving the vehicle routing problem [C]. In Advances in Neural Information Processing Systems. 2018.

[32] Deudon M, Cournut P, Lacoste A, et al. Learning heuristics for the tsp by policy gradient [C]. In International Conference on the Integration of Constraint Programming, Artificial Intelligence, and Operations Research. 2018: 170−181.

[33] Kool W, Van Hoof H, Welling M. Attention, learn to solve routing problems! [C]. In 7th International Conference on Learning Representations, ICLR. 2019.

[34] Ma Q, Ge S, He D, et al. Combinatorial optimization by graph pointer networks and hierarchical reinforcement learning [C]. In AAAI Workshop on Deep Learning on Graphs: Methodologies and Applications. 2020.

[35] Li K, Zhang T, Wang R. Deep Reinforcement Learning for Multiobjective Optimization [J]. IEEE Transactions on Cybernetics. 2020.

[36] Dai H, Khalil E, Zhang Y, et al. Learning combinatorial optimization algorithms over graphs [C]. In Advances in Neural Information Processing Systems. 2017: 6348−6358.

[37] Mittal A, Dhawan A, Manchanda S, et al. Learning heuristics over large graphs via deep reinforcement learning. 2019. arXiv preprint arXiv:190303332,.

[38] Li Z, Chen Q, Koltun V. Combinatorial optimization with graph convolutional networks and guided tree search [C]. In Advances in Neural Information Processing Systems. 2018: 539−548.

[39] Nowak A, Villar S, Bandeira A, et al. A note on learning algorithms for quadratic assignment with graph neural networks [C]. In Proceeding of the 34th International Conference on Machine Learning. 2017: 22.

[40] Joshi C，Laurent T，Bresson X．An efficient graph convolutional network technique for the travelling salesman problem．2019．arXiv preprint arXiv:190601227,．

[41] Helsgaun K. An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems [M]. Roskilde: Roskilde University, 2017.

[42] Perron L, OR-Tools F V. URL https://developers google com/optimization. 2019.

[43] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual. 2021. https://www.gurobi.com.

[44] Bengio Y, Lodi A, Prouvost A. Machine learning for combinatorial optimization: a methodological tour d’ Horizon. 2018. arXiv preprint arXiv:181106128,.

[45] Chen X, Tian Y. Learning to perform local rewriting for combinatorial optimization [C]. In Advances in Neural Information Processing Systems. 2019: 6281−6292.

[46] Yolcu E, Póczos B. Learning local search heuristics for boolean satisfiability [C]. In Advances in Neural Information Processing Systems. 2019: 7992−8003.

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

[48] Lu H, Zhang X, Yang S. A Learning-based Iterative Method for Solving Vehicle Routing Problems [C]. In International Conference on Learning Representations. 2019.

[49] Vaswani A，Shazeer N，Parmar N．Attention is all you need [C]．In Advances in Neural Information Processing Systems．2017：5998−6008．

[50] Scarselli F, Gori M, Tsoi A, et al. The graph neural network model [J]. IEEE Transactions on Neural Networks. 2008, 20 (1): 61−80.

[51] Joshi C, Laurent T, Bresson X. On Learning paradigms for the travelling salesman problem [C]. In NeurIPS Workshop on Graph Representation Learning. 2019.

[52] Joshi C, Cappart Q, Rousseau L-M, et al. Learning TSP requires rethinking generalization. 2020. arXiv preprint arXiv:200607054,.

[53] Barrett T, Clements W, Foerster J, et al. Exploratory combinatorial optimization with reinforcement learning [C]. In Proceedings of the AAAI Conference on Artificial Intelligence. 2020: 3243−3250.

[54] Beloborodov D, Ulanov A, Foerster J, et al. Reinforcement learning enhanced quantum-inspired algorithm for combinatorial optimization. 2020. arXiv preprint arXiv:200204676,.

[55] Cappart Q, Goutierre E, Bergman D, et al. Improving optimization bounds us-

ing machine learning: Decision diagrams meet deep reinforcement learning [C]. In Proceedings of the AAAI Conference on Artificial Intelligence. 2019: 1443–1451.

[56] Abe K, Sato I, Sugiyama M. Solving NP-Hard Problems on graphs by reinforcement learning without domain knowledge [J]. Simulation. 2019, 1: 1.

[57] Hu H, Zhang X, Yan X, et al. Solving a new 3d bin packing problem with deep reinforcement learning method. 2017. arXiv preprint arXiv:170805930,.

[58] Laterre A, Fu Y, Jabri M. Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization. 2018. arXiv preprint arXiv:180701672,.

[59] Huang J, Patwary M, Diamos G. Coloring big graphs with alphagozero. 2019. arXiv preprint arXiv:190210162,.

[60] Li J, Shi W, Zhang N, et al. Delay-aware VNF scheduling: A reinforcement learning approach with variable action set [J]. IEEE Transactions on Cognitive Communications and Networking. 2020.

[61] Mijumbi R，Hasija S，Davy S，等. Topology-aware prediction of virtual network function resource requirements [J]. IEEE Transactions on Network and Service Management. 2017，14 (1)：106–120.

[62] Mijumbi R, Hasija S, Davy S, et al. A connectionist approach to dynamic resource management for virtualised network functions [C]. In 2016 12th International Conference on Network and Service Management. 2016: 1–9.

[63] Quang P, Hadjadj-Aoul Y, Outtagarts A. A deep reinforcement learning approach for VNF Forwarding Graph Embedding [J]. IEEE Transactions on Network and Service Management. 2019, 16 (4): 1318–1331.

[64] Solozabal R, Ceberio J, Sanchoyerto A, et al. Virtual network function placement optimization with deep reinforcement learning [J]. IEEE Journal on Selected Areas in Communications. 2020, 38 (2).

[65] Liu Q, Han T, Moges E. EdgeSlice: Slicing wireless edge computing network with decentralized deep reinforcement learning. 2020. arXiv preprint arXiv:200312911,.

[66] N V H, D T H, DN N, et al. Optimal and fast real-time resource slicing with deep dueling neural networks [J]. IEEE Journal on Selected Areas in Communications. 2019, 37 (6).

[67] Mseddi A, Jaafar W, Elbiaze H, et al. Intelligent resource allocation in dynamic fog computing environments [C]. In 2019 IEEE 8th International Conference on Cloud Networking. 2019: 1–7.

[68] Almasan P, Suárez-Varela J, Badia-Sampera A, et al. Deep reinforcement learn-

ing meets graph neural networks: exploring a routing optimization use case. arXiv preprint arXiv: 191007421, 2020.

[69] Meng X, Inaltekin H, Krongold B. Deep reinforcement learning-based topology optimization for self-organized wireless sensor networks [C]. In 2019 IEEE Global Communications Conference. 2019: 1–6.

[70] Lu J, Feng L, Yang J, et al. Artificial agent: The fusion of artificial intelligence and a mobile agent for energy-efficient traffic control in wireless sensor networks [J]. Future Generation Computer Systems. 2019, 95: 45–51.

[71] Zhang S, Shen W, Zhangt M, et al. Experience-driven wireless D2D network link scheduling: A deep learning approach [C]. In IEEE International Conference on Communications. 2019: 1–6.

[72] Huang L, Bi S, Zhang Y. Deep reinforcement learning for online computation offloading in wireless powered mobile-edge computing networks [J]. IEEE Transactions on Mobile Computing. 2019.

[73] Wang J，Hu J，Min G，等．Computation offloading in multi-access edge computing using a deep sequential model based on reinforcement learning [J]．IEEE Communications Magazine．2019，57 (5)：64–69．

[74] Jiang Q, Zhang Y, Yan J. Neural combinatorial optimization for energy-efficient offloading in mobile edge computing [J]. IEEE Access. 2020, 8.

[75] Yu J, Yu W, Gu J. Online vehicle routing with neural combinatorial optimization and deep reinforcement learning [J]. IEEE Transactions on Intelligent Transportation Systems. 2019, 20 (10).

[76] Holler J, Vuorio R, Qin Z. Deep reinforcement learning for multi-driver vehicle dispatching and repositioning problem [C]. In 2019 IEEE International Conference on Data Mining. 2019: 1090–1095.

[77] Liang X, Du X, Wang G, et al. A deep reinforcement learning network for traffic light cycle control [J]. IEEE Transactions on Vehicular Technology. 2019, 68 (2): 1243–1253.

[78] Chen X, Tian Y. Learning to progressively plan. 2018. arXiv preprint arXiv: 1810.00337,.

[79] Zheng P, Zuo L, Wang J, et al. Pointer networks for solving the permutation flow shop scheduling problem [C]. In Proceedings of International Conference on Computers and Industrial Engineering. 2018: 2–5.

[80] Pan R, Dong X, Han S. Solving permutation flowshop problem with deep rein-

forcement learning [C]. In 2020 Prognostics and Health Management Conference. 2020: 349–353.

[81] Mirhoseini A, Pham H, V L Q. Device placement optimization with reinforcement learning. 2017. arXiv preprint arXiv:170604972,.

[82] Mirhoseini A, Goldie A, Pham H, et al. A hierarchical model for device placement [C]. In International Conference on Learning Representations. 2018.

[83] François-Lavet V, Taralla D, Ernst D, et al. Deep reinforcement learning solutions for energy microgrids management [C]. In European Workshop on Reinforcement Learning. 2016.

[84] Zi-dong Z, Cai-Ming Q, Dong-Xia Z, et al. A coordinated control method for hybrid energy storage system in microgrid based on deep reinforcement learning [J]. Power System Technology. 2019, 43 (6): 1914–1921.

[85] Valladares W, Galindo M, Gutiérrez J. Energy optimization associated with thermal comfort and indoor air control via a deep reinforcement learning algorithm [J]. Building and Environment. 2019, 155: 105–117.

[86] Mocanu E, Mocanu D, Nguyen P. On-line building energy optimization using deep reinforcement learning [J]. IEEE Transactions on Smart Grid. 2018, 10 (4): 3698–3708.

[87] Sutskever I, Vinyals O, Le Q V. Sequence to sequence learning with neural networks [J]. Advances in neural information processing systems. 2014, 27: 3104–3112.

[88] Cho K, Van Merriënboer B, Gulcehre C, et al. Learning phrase representations using RNN encoder-decoder for statistical machine translation [J]. arXiv preprint arXiv:1406.1078. 2014.

[89] Bahdanau D, Cho K, Bengio Y. Neural machine translation by jointly learning to align and translate [J]. arXiv preprint arXiv:1409.0473. 2014.

[90] Rummery G A, Niranjan M. On-line Q-learning using connectionist systems [M]. Citeseer, 1994.

[91] Mossalam H, Assael Y M, Roijers D M, et al. Multi-objective deep reinforcement learning [J]. arXiv preprint arXiv:1610.02707. 2016.

[92] Vinyals O, Fortunato M, Jaitly N. Pointer networks [C]. In Advances in neural information processing systems. 2015: 2692–2700.

[93] Current J R, Schilling D A. The covering salesman problem [J]. Transportation science. 1989, 23 (3): 208–213.

[94] Shariff S R, Moin N H, Omar M. Location allocation modeling for healthcare

facility planning in Malaysia [J]. Computers & Industrial Engineering. 2012, 62 (4): 1000–1010.

[95] Reina D, Marin S T, Bessis N, et al. An evolutionary computation approach for optimizing connectivity in disaster response scenarios [J]. Applied Soft Computing. 2013, 13 (2): 833–845.

[96] Golden B, Naji-Azimi Z, Raghavan S, et al. The generalized covering salesman problem [J]. INFORMS Journal on Computing. 2012, 24 (4): 534–553.

[97] Salari M, Naji-Azimi Z. An integer programming-based local search for the covering salesman problem [J]. Computers & Operations Research. 2012, 39 (11): 2594–2602.

[98] Shaelaie M H, Salari M, Naji-Azimi Z. The generalized covering traveling salesman problem [J]. Applied Soft Computing. 2014, 24: 867–878.

[99] Venkatesh P, Srivastava G, Singh A. A multi-start iterated local search algorithm with variable degree of perturbation for the covering salesman problem [C]. In Harmony Search and Nature Inspired Optimization Algorithms. 2019: 279–292.

[100] Salari M, Reihaneh M, Sabbagh M S. Combining ant colony optimization algorithm and dynamic programming technique for solving the covering salesman problem [J]. Computers & Industrial Engineering. 2015, 83: 244–251.

[101] Tripathy S P, Tulshyan A, Kar S, et al. A metameric genetic algorithm with new operator for covering salesman problem with full coverage [J]. Int J Control Theory Appl. 2017, 10 (7): 245–252.

[102] Pandiri V, Singh A, Rossi A. Two hybrid metaheuristic approaches for the covering salesman problem [J]. Neural Computing and Applications. 2020: 1–21.

[103] Pandiri V, Singh A. An artificial bee colony algorithm with variable degree of perturbation for the generalized covering traveling salesman problem [J]. Applied Soft Computing. 2019, 78: 481–495.

[104] Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need [C]. In Advances in neural information processing systems. 2017: 5998–6008.

[105] Deudon M, Cournut P, Lacoste A, et al. Learning heuristics for the tsp by policy gradient [C]. In International conference on the integration of constraint programming, artificial intelligence, and operations research. 2018: 170–181.

[106] Kool W, van Hoof H, Welling M. Attention, Learn to Solve Routing Problems! [J]. arXiv preprint arXiv:1803.08475. 2018.

[107] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition [C].

In Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770–778.

[108] Williams R J. Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning [J]. Machine Learning. 1998, 8 (3-4): 229–256.

[109] Kingma D P, Ba J. Adam: A method for stochastic optimization [J]. arXiv preprint arXiv:1412.6980. 2014.

[110] Nazari M, Oroojlooy A, Snyder L V, et al. Deep reinforcement learning for solving the vehicle routing problem [J]. arXiv preprint arXiv:1802.04240. 2018.

[111] Reinelt G. TSPLIB—A traveling salesman problem library [J]. ORSA journal on computing. 1991, 3 (4): 376–384.

[112] Deb K, Pratap A, Agarwal S, et al. A fast and elitist multiobjective genetic algorithm: NSGA-II [J]. IEEE transactions on evolutionary computation. 2002, 6 (2): 182–197.

[113] Zhang Q, Li H. MOEA/D: A multiobjective evolutionary algorithm based on decomposition [J]. IEEE Transactions on evolutionary computation. 2007, 11 (6): 712–731.

[114] Ke L, Zhang Q, Battiti R. MOEA/D-ACO: A multiobjective evolutionary algorithm using decomposition and antcolony [J]. IEEE transactions on cybernetics. 2013, 43 (6): 1845–1859.

[115] Beirigo B A, dos Santos A G. Application of nsga-ii framework to the travel planning problem using real-world travel data [C]. In 2016 IEEE Congress on Evolutionary Computation (CEC). 2016: 746–753.

[116] Peng W, Zhang Q, Li H. Comparison between MOEA/D and NSGA-II on the multi-objective travelling salesman problem [C]. In Multi-objective memetic algorithms. 2009: 309–324.

[117] Lin S, Kernighan B W. An effective heuristic algorithm for the traveling-salesman problem [J]. Operations research. 1973, 21 (2): 498–516.

[118] Johnson D. Local search and the traveling salesman problem [C]. In Proceedings of 17th International Colloquium on Automata Languages and Programming, Lecture Notes in Computer Science. 1990: 443–460.

[119] Angel E, Bampis E, Gourvès L. A dynasearch neighborhood for the bicriteria traveling salesman problem [C]. In Metaheuristics for Multiobjective Optimisation. 2004: 153–176.

[120] Jaszkiewicz A. On the performance of multiple-objective genetic local search

on the 0/1 knapsack problem-a comparative experiment [J]. IEEE Transactions on Evolutionary Computation. 2002, 6 (4): 402–412.

[121] Ke L, Zhang Q, Battiti R. A simple yet efficient multiobjective combinatorial optimization method using decompostion and Pareto local search [J]. IEEE Trans. Cybern. 2014, 44: 1808–1820.

[122] Cai X, Li Y, Fan Z, et al. An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization [J]. IEEE Transactions on Evolutionary Computation. 2014, 19 (4): 508–523.

[123] Cai X, Sun H, Zhang Q, et al. A grid weighted sum pareto local search for combinatorial multi and many-objective optimization [J]. IEEE transactions on cybernetics. 2018, 49 (9): 3586–3598.

[124] Lust T, Teghem J. The multiobjective traveling salesman problem: a survey and a new approach [C]. In Advances in Multi-Objective Nature Inspired Computing. 2010: 119–141.

[125] Zhang X, Tian Y, Cheng R, et al. A Decision Variable Clustering-Based Evolutionary Algorithm for Large-scale Many-objective Optimization [J]. IEEE Transactions on Evolutionary Computation. 2016, In Press.

[126] Ming M, Wang R, Zhang T. Evolutionary Many-Constraint Optimization: An Exploratory Analysis [C] // Deb K, Goodman E, Coello Coello C A, et al. In Evolutionary Multi-Criterion Optimization. Cham, 2019: 165–176.

[127] Wolpert D H, Macready W G, et al. No free lunch theorems for optimization [J]. IEEE transactions on evolutionary computation. 1997, 1 (1): 67–82.

[128] Hsu C-H, Chang S-H, Liang J-H, et al. Monas: Multi-objective neural architecture search using reinforcement learning [J]. arXiv preprint arXiv:1806.10332. 2018.

[129] Murata T, Ishibuchi H, Gen M. Specification of genetic search directions in cellular multi-objective genetic algorithms [C] // Zitzler E, Deb K, Thiele L, et al. In Evolutionary Multi-Criterion Optimization. 2001: 82–95.

[130] Li K, Deb K, Zhang Q, et al. An evolutionary many-objective optimization algorithm based on dominance and decomposition [J]. IEEE Transactions on Evolutionary Computation. 2015, 19 (5): 694–716.

[131] Deb K, Jain H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints [J]. IEEE Transactions on Evolutionary Computation. 2014, 18 (4): 577–601.

[132] Miettinen K. Nonlinear multiobjective optimization [M]. Springer Science & Business Media, 2012.

[133] Wang R, Zhou Z, Ishibuchi H, et al. Localized weighted sum method for many-objective optimization [J]. IEEE Transactions on Evolutionary Computation. 2018, 22 (1): 3–18.

[134] Wang R, Zhang Q, Zhang T. Decomposition-based algorithms using Pareto adaptive scalarizing methods [J]. IEEE Transactions on Evolutionary Computation. 2016, 20 (6): 821–837.

[135] Nazari M, Oroojlooy A, Snyder L, et al. Reinforcement learning for solving the vehicle routing problem [C]. In Advances in Neural Information Processing Systems. 2018: 9839–9849.

[136] Bello I, Pham H, Le Q V, et al. Neural combinatorial optimization with reinforcement learning [J]. arXiv preprint arXiv:1611.09940. 2016.

[137] Tian Y, Cheng R, Zhang X, et al. PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization [J]. IEEE Computational Intelligence Magazine. 2017, 12 (4): 73–87.

[138] Glorot X, Bengio Y. Understanding the difficulty of training deep feedforward neural networks [C]. In Proceedings of the thirteenth international conference on artificial intelligence and statistics. 2010: 249–256.

[139] Ishibuchi H, Murata T. A multi-objective genetic local search algorithm and its application to flowshop scheduling [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). 1998, 28 (3): 392–403.

[140] Jaszkiewicz A. Genetic local search for multi-objective combinatorial optimization [J]. European journal of operational research. 2002, 137 (1): 50–71.

[141] Land A H, Doig A G. An automatic method for solving discrete programming problems [C]. In 50 Years of Integer Programming 1958-2008. 2010: 105–132.

[142] Lodi A, Zarpellon G. On learning and branching: a survey [J]. Top. 2017, 25 (2): 207–236.

[143] Gleixner A, Hendel G, Gamrath G, et al. MIPLIB 2017: data-driven compilation of the 6th mixed-integer programming library [J]. Mathematical Programming Computation. 2021: 1–48.

[144] Applegate D, Bixby R, Chvátal V, et al. Finding cuts in the TSP (A preliminary report) [R]. 1995.

[145] Achterberg T, Berthold T. Hybrid branching [C]. In International Conference

on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. 2009: 309–311.

[146] Howard R A. Dynamic programming and markov processes. [J]. 1960.

[147] Gasse M, Chételat D, Ferroni N, et al. Exact combinatorial optimization with graph convolutional neural networks [J]. arXiv preprint arXiv:1906.01629. 2019.

[148] Hussein A, Gaber M M, Elyan E, et al. Imitation learning: A survey of learning methods [J]. ACM Computing Surveys (CSUR). 2017, 50 (2): 1–35.

[149] Balas E, Ho A. Set covering algorithms using cutting planes, heuristics, and subgradient optimization: a computational study [C]. In Combinatorial Optimization. 1980: 37–60.

[150] Cornuéjols G, Sridharan R, Thizy J-M. A comparison of heuristics and relaxations for the capacitated plant location problem [J]. European journal of operational research. 1991, 50 (3): 280–297.

[151] Bergman D, Cire A A, Van Hoeve W-J, et al. Decision diagrams for optimization [M]. Springer, 2016.

[152] Geurts P, Ernst D, Wehenkel L. Extremely randomized trees [J]. Machine learning. 2006, 63 (1): 3–42.

[153] Alvarez A M, Louveaux Q, Wehenkel L. A machine learning-based approximation of strong branching [J]. INFORMS Journal on Computing. 2017, 29 (1): 185–195.

[154] Joachims T. Optimizing search engines using clickthrough data [C]. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. 2002: 133–142.

[155] Burges C J. From ranknet to lambdarank to lambdamart: An overview [J]. Learning. 2010, 11 (23-581): 81.

[156] Khalil E, Le Bodic P, Song L, et al. Learning to branch in mixed integer programming [C]. In Proceedings of the AAAI Conference on Artificial Intelligence. 2016.

[157] Hansknecht C, Joormann I, Stiller S. Cuts, primal heuristics, and learning to branch for the time-dependent traveling salesman problem [J]. arXiv preprint arXiv:1805.01415. 2018.

[158] Wang Z, Sheu J-B. Vehicle routing problem with drones [J]. Transportation research part B: methodological. 2019, 122: 350–364.

[159] De Cubber G, Balta H, Doroftei D, et al. UAS deployment and data processing during the Balkans flooding [C]. In 2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014). 2014: 1–4.

[160] Bry A, Bachrach A, Roy N. State estimation for aggressive flight in GPS-denied environments using onboard sensing [C]. In 2012 IEEE International Conference on Robotics and Automation. 2012: 1–8.

[161] Nikolic J, Burri M, Rehder J, et al. A UAV system for inspection of industrial facilities [C]. In 2013 IEEE Aerospace Conference. 2013: 1–8.

[162] Chen Y, Feng W, Zheng G. Optimum placement of UAV as relays [J]. IEEE Communications Letters. 2017, 22 (2): 248–251.

[163] Simic M, Bil C, Vojisavljevic V. Investigation in wireless power transmission for UAV charging [J]. Procedia Computer Science. 2015, 60: 1846–1855.

[164] Yan H, Chen Y, Yang S-H. UAV-Enabled Wireless Power Transfer With Base Station Charging and UAV Power Consumption [J]. IEEE Transactions on Vehicular Technology. 2020, 69 (11): 12883–12896.

[165] Zeng Y, Xu J, Zhang R. Energy minimization for wireless communication with rotary-wing UAV [J]. IEEE Transactions on Wireless Communications. 2019, 18 (4): 2329–2345.

[166] Perron L, Furnon V. OR-Tools. https://developers.google.com/optimization/.

[167] Clarke G, Wright J W. Scheduling of vehicles from a central depot to a number of delivery points [J]. Operations research. 1964, 12 (4): 568–581.

[168] Marinakis Y, Marinaki M, Migdalas A. A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows [J]. Information Sciences. 2019, 481: 311–329.

[169] Zhang H, Zhang Q, Ma L, et al. A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows [J]. Information Sciences. 2019, 490: 166–190.

[170] Azi N, Gendreau M, Potvin J-Y. An adaptive large neighborhood search for a vehicle routing problem with multiple routes [J]. Computers & Operations Research. 2014, 41: 167–173.