DeepACO：用于组合优化的神经增强蚂蚁系统

摘要

蚁群算法是一种常用的元启发式搜索算法，且在求解组合优化问题上有不错的效果。但是传统的蚁群算法必须针对某一类问题特定设计一些专家规则，因此其效果很大程度上依赖于制定者水平的高低以及对问题理解程度。针对这个情况，本文提出了深度蚁群算法（DeepACO），利用深度强化学习来自主学习启发式规则。因此 DeepACO 能够有效的辅助已有的蚁群算法，免去手动设计复杂的启发式规则。DeepACO 使用单一神经架构和一组超参数在 8 个组合优化问题上始终优于其他传统的蚁群算法。

关键词：蚁群算法；组合优化问题；深度强化学习

1 引言

本部分主要是根据这篇论文的选题背景以及论文的贡献。

1.1 选题背景

蚂蚁系统本质上是自学习者。它们利用化学信号和环境线索来定位食物并将其返回群体。蚂蚁沉积的信息素痕迹表明了食物源的质量和距离。信息素痕迹的强度随着更多蚂蚁的到来而增加，并由于蒸发而减少，从而创建了一个自学习的觅食系统。受自然界蚂蚁系统的启发，研究人员提出并开发了蚁群优化（ACO）元启发式算法，用于（但不限于）组合优化问题（COP）[1] 。ACO 部署一群人工蚂蚁，通过重复的解构建和信息素更新来探索解空间。通过特定于实例的信息素轨迹和特定于问题的启发式措施，探索偏向于更有前景的领域。信息素踪迹和启发式测量都表明解决方案组件的前景如何。通常，信息素轨迹会针对所有解决方案组件进行统一初始化，并在解决实例时进行学习。相反，启发式措施是根据问题的先验知识预先定义的，并且为复杂的 COP 设计适当的启发式措施非常具有挑战性。在过去的几十年里，研究和实践工作一直致力于精心设计启发式措施，以追求知识驱动的绩效提升。然而，这种算法定制例程存在一定的缺陷：1）需要额外的努力，并且使得 ACO 灵活性较差；2）启发式措施的有效性很大程度上依赖于专家知识和人工调优；3）考虑到可用的专业知识的缺乏，为研究较少的问题设计启发式措施可能特别具有挑战性。

1.2 选题意义

由于目前的蚁群算法大都只注意争对某种特定问题设计启发式规则，因此当问题类型发生改变那么就需要重新设计启发式规则，但是某些问题的启发式不容易设计，这需要设计人

员有较强的经验。为了解决这个缺点，论文提出了 DeepACO，一种通用的神经增强 ACO 元启发式算法，以及针对上述限制的解决方案。DeepACO 旨在加强现有 ACO 算法的启发式措施，并在未来的 ACO 应用中免除繁琐的手动设计。主要涉及两个学习阶段。第一阶段通过跨 COP 实例训练神经模型来学习从实例到其启发式测量的特定问题映射。在学习到的测量的指导下，第二阶段在使用 ACO 求解实例的同时学习实例特定的信息素轨迹。通过偏置解决方案构造并引导局部搜索（LS）逃避局部最优，将第一阶段学习的启发式措施纳入 ACO（第二学习阶段）。

1.3 贡献

总结，这篇论文的主要贡献如下：

- 提出 DeepACO，一种神经增强的 ACO 元启发式算法。它增强了现有的 ACO 算法，并在未来的 ACO 应用中免除了繁琐的手动设计。
- 提出了 DeepACO 的三种扩展实现来平衡探索和利用，这通常可以应用于基于热图的 NCO 方法。
- 验证了 DeepACO 在八个 COP 中始终优于其 ACO 同行，同时表现优于或与特定问题的 NCO 方法相当。

2 相关工作

本部分主要介绍目前神经组合优化的最新研究方法以及简单介绍一下传统蚁群算法的算法思想和框架。

2.1 神经组合优化

神经组合优化 (NCO) 是一个跨学科领域，利用深度学习技术解决 COP。一般来说，现有的 NCO 方法可以分为端到端的方法和混合方法。端到端的方法通过学习自回归解决方案构造或热图生成，以进行后续基于采样的解码。基于学习自回归的方式的代表有 Kool 等 [4] 提出基于自注意力层的模型。基于构造热图的方式有 Dai [3] 提出了强化学习和图嵌入的组合方式来求解组合优化问题。Xiao [7] 提出了一种通用的引导非自回归知识蒸馏（GNARKD）方法来获得具有低推理延迟的高性能非自回归模型，从而解决非自回归模型的推理能力差的问题。虽然端到端方法非常高效，但它们在处理大规模数据上得到不到很好的泛化能力，所以目前也有许多学者研究如何在大规模和超大规模上是泛化问题。因此端到端构建的解决方案可以通过迭代细化和算法混合进一步改进。

混合方法结合了神经网络来训练出一个启发式规则，如本篇论文 Ye [8] 提出的一种神经增强的蚁群算法，通过神经网络设计启发式规则，或者通过辅助启发式规则来做出决策如 Ma [5] 提出一种用于解决路由问题的新型学习搜索 (L2S) 求解器。它学习基于定制的动作分解方法和定制的循环双流解码器来执行灵活的 k-opt 交换。这些方法都是通过训练出一个神经学习者来自主学习相应的启发式规则，本质上式基于数据设计，设计出的启发式规则需要通过与传统的元启发式框架结合来对问题求解。

由于两者的主体方法不同，因此两者的求解效果以及求解速度也有不同，端到端的方式因为利用预训练的方式，因此当模型使用在同一类型的问题求解上，其求解速度很快，但是因为该方法的模型需要进行大量的预训练，因此其运用于大规模数据上的收效甚微。与之相对的，利用深度神经网络辅助的元启发式算法的求解速度会慢很多，但是却能够有不错的求解效果。很明显，DeepACO 属于混合方法，DeepACO 利用深度强化学习对启发式规则进行学习，然后在 ACO 这算法框架下进行迭代搜索。

2.2 传统蚁群算法

我们首先定义 COP 模型和信息素模型。它们是实现 ACO 算法的先决条件。

COP 模型 一般来说，COP 模型由以下部分组成：1) 在一组离散决策变量 $X_i, i = 1, ..., n$ 上定义的搜索空间 $\mathbf{S}$，其中每个决策变量 X_i 从有限集 $D_i = v_i^1, ..., v_i^{|D_i|}$ 中取值; 2）决策变量必须满足的一组约束 Ω; 3) 最小化目标函数 $f : \mathbf{S} \to \mathbb{R}_0^+$。COP 的可行解决方案 s 是满足 Ω 中所有约束的所有决策变量的完整分配。

信息素矩阵 COP 模型定义了 ACO 背景下的信息素模型。不失一般性，信息素模型是一个构造图，其中包括作为节点的决策变量和作为边的解决方案组件 [26]。每个解决方案分量 c_{ij} 表示将值 v_{ji} 分配给决策变量 X_i，与其信息素试验 τ_{ij} 和启发式测量 η_{ij} 相关联。τ_{ij} 和 ηij 都表明将 c_{ij} 包含在解决方案中的可能性有多大。通常，ACO 统一初始化并迭代更新信息素轨迹，但预定义并修复启发式措施。作为一个激励示例，对于 TSP，实例可以直接转换为全连接构造图，其中城市 i 成为节点 i，解决方案组件 c_{ij} 表示在城市 i 之后立即访问城市 j。然后，$_{ij}$ 通常设置为城市 i 和 j 之间距离的倒数。定义 COP 模型和信息素模型后，我们引入 ACO 迭代，通常包括解决方案构造、可选的局部搜索（Local search，LS）进行细化和信息素更新。

解决方案构建和局部搜索（可选） 在 τ_{ij} 和 η_{ij} 的两个偏置下，人工蚂蚁通过遍历构造图构造解 $s = \{s_t\}_{t=1}^n$。如果一只蚂蚁在第 t 个构造步骤 $(s_{t-1} = i)$ 位于节点 i 并构造了部分解 $s_{<t} = \{s_t\}_{t=1}^{t-1}$，则选择节点 j 作为其下一个目的地 $(s_t = j)$ 通常由下式给出

$$P(s_t|s_{<t}, \rho) = \begin{cases} \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{c_{il} \in N(s_{<t})} \tau_{il}^{\alpha} \cdot \eta_{il}^{\beta}} & \text{if } c_i j \in N(s_{s<t}), \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

这里，ρ 表示 COP 实例，$N(s_{<t})$ 是给定部分解的可行解分量的集合，α 和 β 是控制参数，除非另有说明，在本工作中始终设置为 1。为了简化符号，我们省略了 τ η c 和 N 对 ρ 的依赖。基于等式 (1)，构建完整的解决方案需要 n 步图遍历。生成 s 的概率可以分解为

$$P(s|\rho) = \prod_{t=1}^{n} P(s_t|s_{<t}, \rho). \tag{2}$$

在构建解决方案后，可选择应用局部搜索 (LS) 来细化解决方案。

信息素更新 在解决方案构建后，信息素更新过程会评估解决方案并相应地调整信息素轨迹，即增加优等解决方案中组件的信息素轨迹，同时减少劣等解决方案中的组件信息素轨迹。详细的更新规则可能会有所不同，具体取决于所使用的 ACO 变体。ACO 通过迭代上述过程智能地探索解空间，最终收敛到（次）最优解。我们建议读者参考 [1] 了解更多细节。

3 本文方法

3.1 本文方法概述

此部分主要是对论文的方法进行总体的介绍以及对于每个创新点做出详细的分析，DeepACO 的算法框架如图1所示：

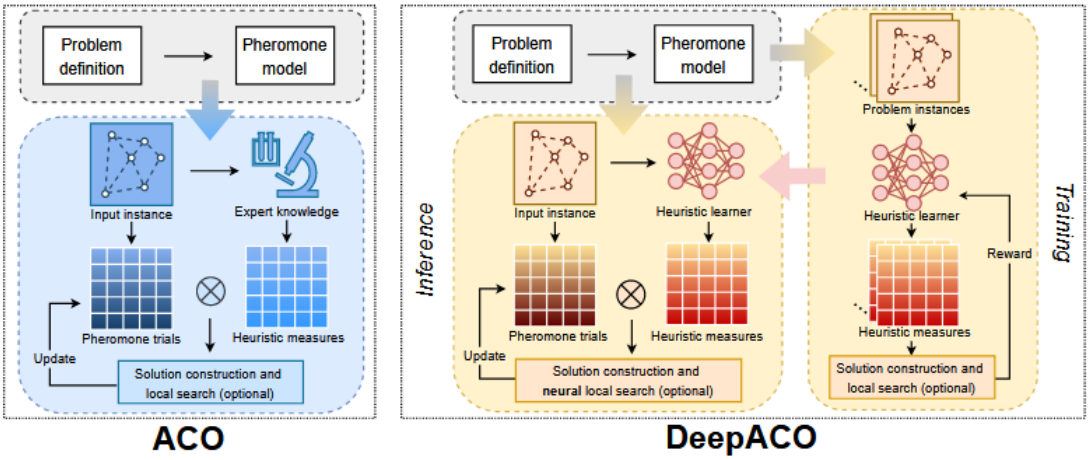

图 1. DeepACO 算法框架

左边为传统的 ACO 算法框架，右边为神经增强后的 ACO 方法。从图中可以看到左边为传统的 ACO 算法框架，而图中右边则是 DeepACO 算法框架，两者的差别在于传统的 ACO 使用的式专家知识来设计启发式，而 DeepACO 采用的是神经学习者来学习启发式规则。

3.2 参数化启发式空间

为了使神经学习者学习到启发式策略，论文引入了图神经网络以及可学习的参数去参数化启发式规则，这里启发式策略简写为 η_θ，它包含与每个解分量 c_{ij}，$i \in 1, ..., n$，$j \in 1, ..., |D_i|$。DeepACO 按照方程 (2) 构建解决方案，但有 η_θ 的偏差：

$$P_{\eta_\theta}(s|\rho) = \prod_{t=1}^{n} P_{\eta_\theta}(s_t|s_{<t}, \rho). \tag{3}$$

特别是，我们利用了 Joshi [2] 和 Qiu [6] 等人推荐的神经模型。它由依赖于各向异性消息传递和边缘门控机制的 GNN 主干网络以及将提取的边缘特征映射为启发式度量的多层感知器 (MLP) 解码器组成。

3.3 局部搜索与神经引导扰动交织

在 ACO 中，可以选择应用局部搜索（LS）来细化构建的解决方案。然而，LS 是一种局限性的方法，因为它贪婪地接受任何具有较低目标值的改变解决方案，并且很容易陷入局部最优。在 DeepACO 中，使用了经过充分学习的启发式措施来指示解决方案组件的全局最优性。利用这些指标并包含具有更大全局最优性的解决方案组件，最终（即使不是立即）可以

带来更好的解决方案。然而，由于 COP 固有的复杂性，仅仅依靠学到的启发式方法是不现实的。

基于这些考虑，论文在算法 1 中提出了 LS 与神经引导扰动的交织（简称 NLS）。NLS 交织了旨在降低目标值的 LS 和使学习到的最优值产生偏差的神经引导扰动。在每次迭代中，第一阶段利用 LS 反复细化解决方案，直到（可能）达到局部最优。第二阶段利用 LS 稍微扰动局部最优解，以获得更高的累积启发式测量。

3.4 训练启发式学习器

我们在 COP 实例训练启发式学习器。启发式学习器 θ 将每个实例 ρ 映射到其启发式测量 η_ρ。然后，我们最小化构造解和 NLS 细化构造解的预期目标值：

$$\text{minimize} \quad \mathcal{L}(\theta|\rho) = \mathbb{E}_{s\sim P_{\eta_\theta}(\cdot|\rho)}[f(s) + Wf(NLS(s, f, +\infty))] \tag{4}$$

其中 W 是平衡两项的系数。直观上，第一个损失项鼓励直接构建最优解决方案，然而，这往往因 COP 的复杂性而被推迟。第二个则鼓励构建最适合 NLS 的解决方案，如果将端到端构建与 NLS 结合起来，可以更容易地学习高质量的解决方案。即便如此，由于 NLS 细化解决方案的质量差异较小，仅依赖第二项会导致训练效率低下。因此，我们发现实施这两个损失项进行训练很有用。请注意，NLS 过程本身不涉及梯度下降。

3.5 三种探索方式

蚁群算法的采用类似于自回归的方式，这种方式在每一步都通过采样来选择下一个节点，因此会使得概率较大的节点会被经常选中，这使得到这个点的信息素增加，从而收敛到局部最优。因此，蚁群算法有一个需要平衡的点就是如何利用已有的信息素来求解以及如何探索新的解。为此 DeepACO 这篇论文提出了三种方式来平衡探索与利用。

多头注意力: DeepACO 在 GNN 中使用 m 个多层感知机解码器，用来产生不同的启发式，以鼓励探索解决方案空间中的不同最优值。

Top-k 熵损失： 保持其行为的多样性。它经常被用作正则化器来鼓励探索并防止过早收敛到次优策略。

模仿损失： 设计相关的专家启发式，然后通过模仿学习计算损失函数，从而不断逼近专家启发式。

4 复现细节

该论文已经在 Github 上开源了，因此该选题就直接运行相关对比实验。因为我目前研究在于旅行商问题（Travelling Salesman Problem，TSP）和车辆路径规划（Vehicle Routing Problem，VRP），因此仅对这两个人问题进行实验复现。

4.1 实验环境搭建

论文给定的配置环境如下环境如下：

- **Python:**3.8.0

- **Pytorch:**1.7.0
- **numpy:**1.23.3

本实验结果采用的平台是：CPU 使用 Intel 12700k，GPU 使用 RTX 3090。

5 实验结果分析

本部分对实验所得结果进行分析，详细对实验内容进行说明，实验结果进行描述并分析。

方法	TSP100	
	Obj.	Time(s)
AM	7.945	0.36
GNN	7.907	6.13
DeepACO(NLS,T=4)	7.767	0.50
DeepACO(NLS,T=10)	7.763	1.23
DeepACO*(NLS,T=4)	7.763	0.48
DeepACO*(NLS,T=10)	7.778	1.25

表 1. TPS100 的实验结果

从图中可以看到实际 TSP 的复现的实验结果与原论文相差无二，原论文在最后生成的最终解中加入了局部搜索，然后分别采用的 T=4 和 T=10 步的扰动。

方法	CVRP100		CVRP400		CVRP1000		CVRP2000	
	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)	Obj.	Time(s)
AM	16.42	0.06	29.33	0.20	61.42	0.59	114.36	1.87
TAM-LKH3	16.08	0.86	25.93	1.35	46.34	1.82	64.78	5.63
DeepACO(NLS,T=4)	16.07	2.97	25.31	3.65	45.00	10.21	61.89	14.53
DeepACO(NLS,T=10)	15.77	3.87	25.27	5.89	44.82	15.87	61.66	35.94
DeepACO*(NLS,T=4)	16.08	2.96	25.27	3.70	45.11	10.30	61.93	14.44
DeepACO*(NLS,T=10)	15.74	3.82	25.15	5.95	44.52	16.27	61.72	36.14

表 2. CVRP 问题的求解效果

对于 CVRP 问题，复现结果也与原论基本一致，同样是在最后生成的解中加入局部搜索算子，采用了 T=4 和 T=10 步的扰动，进一步收敛最后的结果。

6 总结与展望

DeepACO 与传统的 ACO 算法相比，能够取得比较好的效果，同时利用深度强化学习进行算法增强和设计自动化，很好地辅助已有的蚁群算法，对于机器学习领域，DeepACO 提供

了一个多功能且适应性强的 NCO 框架，可以与 SOTA ACO 和 EA 算法无缝集成，包括改进的解决方案构建和信息素更新规则、算法融合以及复杂的本地搜索运算符的整合。但是由于蚁群算法的求解依赖于 $n \times n$ 的信息素矩阵，因此 DeepACO 可能会受到所有学习的启发式信息压缩为 n × n 启发式度量矩阵的限制。由于 COP 的复杂性和这种表达问题模式的受限方式，DeepACO 在未合并 LS 组件时可能无法产生接近最优的解决方案。因此可以尝试通过利用深度强化学习辅助的混合元启发算法对问题进行求解，避免通过深度强化学习学到的启发式信息受限从而导致求解能力下降

参考文献

[1] Marco Dorigo and Thomas Stützle. *Ant colony optimization: overview and recent advances*. Springer, 2019.

[2] Chaitanya K Joshi, Quentin Cappart, Louis-Martin Rousseau, and Thomas Laurent. Learning the travelling salesperson problem requires rethinking generalization. *arXiv preprint arXiv:2006.07054*, 2020.

[3] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

[4] Wouter Kool, Herke Van Hoof, and Max Welling. Attention, learn to solve routing problems! *arXiv preprint arXiv:1803.08475*, 2018.

[5] Yining Ma, Zhiguang Cao, and Yeow Meng Chee. Learning to search feasible and infeasible regions of routing problems with flexible neural k-opt. *Advances in Neural Information Processing Systems*, 36, 2024.

[6] Ruizhong Qiu, Zhiqing Sun, and Yiming Yang. Dimes: A differentiable meta solver for combinatorial optimization problems. *Advances in Neural Information Processing Systems*, 35:25531–25546, 2022.

[7] Yubin Xiao, Di Wang, Boyang Li, Mingzhao Wang, Xuan Wu, Changliang Zhou, and You Zhou. Distilling autoregressive models to obtain high-performance non-autoregressive solvers for vehicle routing problems with faster inference speed. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20274–20283, 2024.

[8] Haoran Ye, Jiarui Wang, Zhiguang Cao, Helan Liang, and Yong Li. Deepaco: neural-enhanced ant systems for combinatorial optimization. *Proc. Conf. on Neural Information Processing Systems*, 36, 2024.