A Unified Neural Divide-and-Conquer Framework for Large-Scale CO Problems

摘要

单级神经组合优化求解器在不需要专家知识的情况下，对各种小规模组合优化（CO）问题取得了接近最优的结果。然而，当应用于大规模 CO 问题时，这些求解器表现出显著的性能下降 [1] 。近年来，由分而治之策略驱动的两阶段神经方法在解决大规模 CO 问题方面显示出高效率。然而，这些方法的性能在划分或征服过程中高度依赖于问题特定的启发式，这限制了它们对一般 CO 问题的适用性 [2] 。此外，这些方法采用单独的训练方案，忽略了划分和征服策略之间的相互依赖性，常常导致次优解。为了解决这些缺点，本文开发了一个统一的神经分治框架（即 UDC），用于解决一般的大规模 CO 问题。UDC 提供了一种划分-征服-重聚（DCR）训练方法来消除次优划分策略的负面影响。所提出的 UDC 框架采用高效的图神经网络（GNN）进行全局实例划分，采用定长子路径求解器征服划分的子问题，具有广泛的适用性，在 10 个具有代表性的大规模 CO 问题中取得了优异的性能。

关键词：大规模组合优化（Large-scale CO）；神经网络（Neural Networks）；分而治之（Divide-and-Conquer）；图神经网络（Graph Neural Networks）；划分-征服-重聚（DCR）

1 引言

主要研究（超）大规模（高达 500、1000、2000 节点）的 TSP 问题、CVRP、OP 问题和 MIS 以及相关变种等 10 个问题，即大规模或超大规模组合优化问题 [3] 。组合优化 (CO) 有许多实际应用，包括路线规划、电路设计、生物学等。近年来，神经组合优化 (neural combinatorial optimization, NCO) 方法得到了发展，以有效解决旅行商问题 (traveling Salesman Problem, TSP) 和有能力车辆路径问题 (Capacitated Vehicle Routing Problem,CVRP) 等典型 CO 问题 [4] 。

2 相关工作

2.1 利用 SL 的单阶段求解器

BQ-NCO 和 LEHD 开发了子路径构建过程，并采用带重解码器的模型来学习该过程。然而，训练这样的重量级模型需要监督学习 (SL)，这限制了它们在高质量解决方案无法作为标签提供的问题中的适用性 [5] 。

2.2 利用 RL 的单阶段求解器

ELG、ICAM 和 DAR 整合了辅助信息来指导基于 rl 的单阶段构造求解器的学习。然而，辅助信息需要针对特定问题的设计，并且自注意机制导致的 O(N2) 复杂性也对大规模 (N 节点) 问题提出了挑战 [6] 。

2.3 神经分治法

这些方法执行两个阶段的求解过程，包括用于总体实例划分的划分阶段和用于子问题求解的征服阶段。通过在一个或两个阶段使用神经网络，TAM、H-TSP 和 GLOP 在大规模 TSP 和 CVRP 中表现出优越的效率。尽管这些神经分治法的应用前景广阔，但在适用性和求解质量方面仍存在不足 [7] 。一些神经分治方法依赖于问题特定的启发式划分 (例如，GLOP 和 SO) 或征服阶段 (例如，L2D 和 RBG)，这限制了它们在各种 CO 问题中的泛化性。此外，所有现有的神经分治方法都采用单独的训练过程，在预训练的征服策略之后训练分割策略。然而，单独的训练方案忽略了分治策略和征服策略之间的相互依赖性，可能导致无法解决的对抗 [8] 。一旦预训练的征服策略对于某些子问题是次优的，这样的训练方案可能会引导分割策略适应预训练的次优征服策略，从而收敛到局部最优。

3 本文方法

3.1 本文方法概述

如图所示：

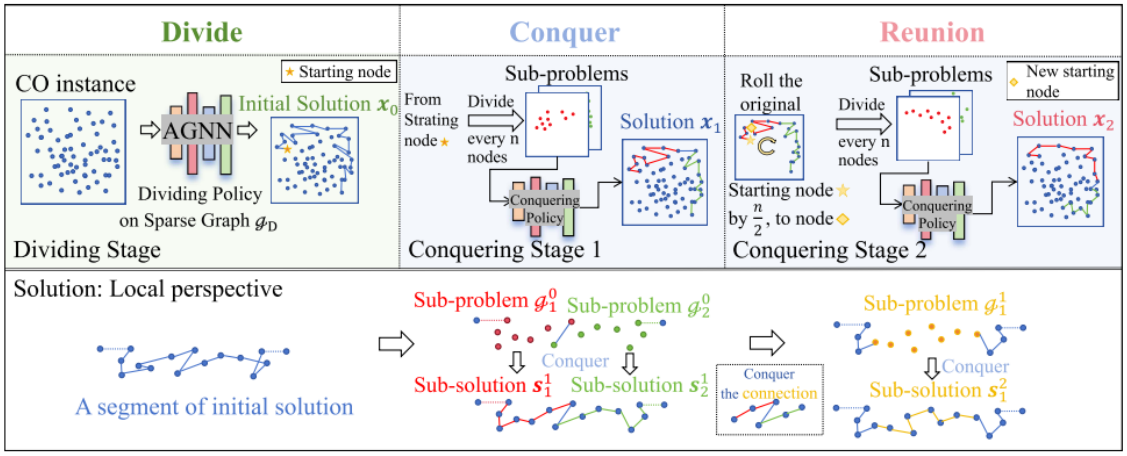

图 1. 相比传统的神经分治方法，增加了 reunion 的过程，这个过程考虑了 divide 阶段划分导致的次优解情况，从而优化了 divide 过程。文中也将 reunion 视为第二阶段的 conquer（错开步幅）。

3.2 马尔可夫过程相关函数

$$\pi_c(s_k \mid \mathcal{G}_k, \Omega_k, \theta) = \begin{cases} \prod_{t=1}^{n} p(s_{k,t} \mid s_{k,1:t-1}\mathcal{G}_k, \Omega_k, \theta), & \text{if } s_k \in \Omega_k \\ 0, & \text{otherwise} \end{cases}$$

$$\nabla \mathcal{L}_{c1}(\mathcal{G}) = \frac{1}{\alpha\beta\left\lfloor \frac{N}{n} \right\rfloor} \sum_{c=1}^{\alpha\left\lfloor \frac{N}{n} \right\rfloor} \sum_{i=1}^{\beta} \left[\left(f'(s_c^{1,i}, \mathcal{G}_c^0) - \frac{1}{\beta} \sum_{j=1}^{\beta} f'(s_c^{1,j}, \mathcal{G}_c^0) \right) \nabla \log \pi_c(s_c^{1,j} \mid \mathcal{G}_c^0, \Omega_c^0, \theta) \right]$$

$$\nabla \mathcal{L}_{c2}(\mathcal{G}) = \frac{1}{\alpha\beta\left\lfloor \frac{N}{n} \right\rfloor} \sum_{c=1}^{\alpha\left\lfloor \frac{N}{n} \right\rfloor} \sum_{i=1}^{\beta} \left[\left(f'(s_c^{2,i}, \mathcal{G}_c^1) - \frac{1}{\beta} \sum_{j=1}^{\beta} f'(s_c^{2,j}, \mathcal{G}_c^1) \right) \nabla \log \pi_c(s_c^{2,j} \mid \mathcal{G}_c^1, \Omega_c^1, \theta) \right]$$

图 2. 策略函数与梯度

4 复现细节

4.1 与已有开源代码对比

在本项目中，我在原有的统一神经分治框架（UDC）上进行了一些创新性改进，尤其是在训练效率方面。我利用了多显卡共同训练的技术，使得模型能够在多台 GPU 设备上并行处理，从而显著提高了训练速度和效率。这一改进有效解决了大规模 CO 问题训练过程中计算资源不足的问题，能够在更短的时间内训练出更加准确和高效的模型。我的贡献主要体现在以下几个方面：多显卡训练的实现：我采用了基于数据并行的多显卡训练策略，通过合理划分数据批次并将其分配到不同的 GPU 上进行并行计算，从而加速了模型训练过程，尤其是在处理大规模数据时，能够大幅减少训练时间。训练效率的显著提升：通过引入分布式训练框架，我显著提高了训练过程的速度，使得在大规模 CO 问题的应用中，模型能够在更短的时间内完成训练，适应更复杂的任务需求。高效利用计算资源：我在保证训练精度的前提下，优化了多显卡训练中的数据和模型参数分配，确保了 GPU 资源的最大化利用，提升了整体计算效率。这些改进与技术贡献，不仅有效提升了训练效率，还使得模型在面对大规模组合优化问题时，能够在合理的时间内完成高质量的训练，从而更好地应对实际应用中的挑战。

4.2 原文实验结果

越大规模效果越好，conquer 越多次（x 的下标）越好，而且基本都是学习类的最优方案。

	Methods	N =500 Obj.↓	Gap	Time	N =1,000 Obj.↓	Gap	Time	N =2,000 Obj.↓	Gap	Time
TSP	LKH3	**16.52**	-	5.5m	**23.12**	-	24m	**32.45**	-	1h
	Attn-MCTS*	16.97	2.69%	6m	23.86	3.20%	12m	33.42	2.99%	-
	DIMES*	16.84	1.93%	2.2h	23.69	2.47%	4.6h	-	-	-
	DIFUSCO+Search*	17.48	5.80%	19m	25.11	8.61%	59.2m	-	-	-
	BQ	16.72	1.18%	46s	23.65	2.27%	1.9m	34.03	4.86%	3m
	LEHD	16.78	1.56%	16s	23.85	3.17%	1.6m	34.71	6.98%	12m
	POMO	20.19	22.19%	1m	32.50	40.57%	8m	50.89	56.85%	10m
	ELG	17.18	4.00%	2.2m	24.78	7.18%	13.7m	36.14	11.37%	14.2m
	ICAM	16.65	0.77%	38s	23.49	1.58%	1m	34.37	5.93%	3.8m
	GLOP(more revisions)	16.88	2.19%	1.5m	23.84	3.12%	3.5m	33.71	3.89%	9m
	SO-mixed*	16.94	2.55%	32m	23.77	2.79%	32m	33.35	2.76%	14m
	H-TSP	17.55	6.22%	20s	24.72	6.91%	40s	34.88	7.49%	48s
	UDC-$\boldsymbol{x}_2$(α=50, greedy)	16.94	2.53%	20s	23.79	2.92%	32s	34.14	5.19%	23s
	UDC-$\boldsymbol{x}_{50}$(α=50)	16.78	1.58%	4m	23.53	1.78%	8m	33.26	2.49%	15m
CVRP	LKH3	**37.23**	-	7.1h	46.40	7.90%	14m	64.90	8.14%	40m
	BQ	38.44	3.25%	47s	44.17	2.72%	55s	62.59	4.29%	3m
	LEHD	38.41	3.18%	17s	43.96	2.24%	1.3m	61.58	2.62%	9.5m
	ELG	38.34	2.99%	2.6m	43.58	1.33%	15.6m	-	-	-
	ICAM	37.49	0.69%	42s	43.07	0.16%	26s	61.34	2.21%	3.7m
	L2D*	-	-	-	46.3	7.67%	3m	65.2	8.64%	1.4h
	TAM(LKH3)*	-	-	-	46.3	7.67%	4m	64.8	7.97%	9.6m
	GLOP-G (LKH-3)	42.45	11.73%	2.4m	45.90	6.74%	2m	63.02	5.01%	2.5m
	UDC-$\boldsymbol{x}_2$(α=50, greedy)	40.04	7.57%	1m	46.09	7.18%	2.1m	65.26	8.75%	5m
	UDC-$\boldsymbol{x}_{50}$(α=50)	38.34	3.01%	7.7m	43.48	1.11%	14m	60.94	1.55%	23m
	UDC-$\boldsymbol{x}_{250}$(α=50)	37.99	2.06%	35m	**43.00**	-	1.2h	**60.01**	-	2.15h

图 3. TSP 与 CVRP

	Methods	N =500 Obj.↑[2]	Gap	Time	N =1,000 Obj.↓[2]	Gap	Time	N =2,000 Obj.↓[2]	Gap	Time
OP (dist)	EA4OP	**81.70**	-	7.4m	**117.70**	-	63.4m	**167.74**	-	38.4m
	BQ-bs16*	-	4.10%	10m	-	10.68%	17m	-	-	-
	AM	64.45	21.11%	2s	79.28	32.64%	5s	99.37	40.76%	2s
	AM-bs1024	68.59	16.04%	1.2m	79.65	32.32%	4m	95.48	43.08%	1.6m
	MDAM-bs50	65.78	19.49%	2.6m	81.35	30.88%	12m	105.43	37.15%	7m
	Sym-NCO	73.22	10.38%	27s	95.36	18.98%	1.5m	121.37	27.64%	20s
	UDC-$\boldsymbol{x}_2$(α=1)	75.28	7.86%	17s	109.42	7.03%	25s	135.34	19.32%	5s
	UDC-$\boldsymbol{x}_{50}$(α=1)	78.02	4.51%	23s	111.72	5.08%	33s	139.25	16.99%	8s
	UDC-$\boldsymbol{x}_{250}$(α=1)	79.27	2.98%	48s	113.27	3.76%	1.1m	144.10	14.09%	21s
PCTSP	OR-Tools*	14.40	9.38%	16h	20.60	12.66%	16h	-	-	-
	AM-bs1024	19.37	47.13%	14m	34.82	90.42%	23m	87.53	249.60%	10m
	MDAM-bs50	14.80	13.30%	6.5m	22.21	21.43%	53m	36.63	46.30%	43m
	Sym-NCO	19.46	47.81%	1.3m	29.44	61.01%	7m	36.95	47.58%	2m
	GLOP	14.30	9.03%	1.5m	19.80	8.08%	2.5m	27.25	8.84%	1m
	UDC-$\boldsymbol{x}_2$(α=1)	13.95	5.99%	34s	19.50	6.65%	1m	27.22	8.73%	18s
	UDC-$\boldsymbol{x}_{50}$(α=1)	13.25	0.64%	42s	18.46	0.97%	1.3m	25.46	1.68%	23s
	UDC-$\boldsymbol{x}_{250}$(α=1)	**13.17**	-	80s	**18.29**	-	1.5m	**25.04**	-	53s
SPCTSP	AM-bs1024	33.91	156.11%	1.2m	47.55	156.42%	4m	76.02	199.48%	14m
	MDAM	14.80	11.80%	6.5m	21.96	18.42%	53m	35.15	38.48%	43m
	UDC-$\boldsymbol{x}_2$(α=1)	14.21	7.32%	42s	20.01	7.92%	1.1m	28.15	10.91%	20s
	UDC-$\boldsymbol{x}_{50}$(α=1)	13.40	1.12%	1m	18.93	2.06%	1.4m	26.24	3.38%	29s
	UDC-$\boldsymbol{x}_{250}$(α=1)	**13.24**	-	2.3m	**18.54**	-	2.6m	**25.38**	-	1.1m

图 4. OP 与 TSP 变种问题

Table 4: TSPLib and CVRPLib results, the best learning-based result is marked by shade

TSPLib, Gap to Best Known Solution						
Dataset, $N \in$	LEHD	ELG aug×8	GLOP-LKH3	TAM(LKH3)	UDC-$\boldsymbol{x}_2$	UDC-$\boldsymbol{x}_{250}$
TSPLib,(500,1,000]	4.1%	8.7%	4.0%	-	9.5%	6.0%
TSPLib,(1,000,5,000]	11.3%	15.5%	6.9%	-	12.8%	7.6%
CVRPLib, Gap to Best Known Solution						
Set-X,(500,1,000]	17.4%	7.8%	16.8%	9.9%	16.5%	7.1%
Set-XXL,(1,000,10,000]	22.2%	15.2%	19.1%	20.4%	31.3%	13.2%

图 5. Lib

5 实验结果分析

对论文中 n=500 节点的规模进行了实验结果复现。

深圳大学 SHENZHEN UNIVERSITY

实验结果复现（N=500）

		原文（V100S）		复现(RTX4090)	
		Obj.	Time	Obj.	Time
TSP	LKH3	16.52	5.5m		
	UDC-x_2（α=50）	16.94	20s	16.9412	16s
	UDC-x_{50}（α=50）	16.78	4m	16.7853	2m 43s
CVRP	LKH3	37.23	7.1h		
	UDC-x_2（α=50）	40.04	1m	40.900	53s
	UDC-x_{50}（α=50）	38.34	7.7m	38.349	5m 22s
	UDC-x_{250}（α=50）	37.99	35m	37.977	18m 20s

存在的问题：

1. 时间更短可能是显卡影响。
2. 1000节点时间过长只验证了一个未放出。
3. 2000节点的数据集未得到未验证。

11.25: Fixed a bug in CVRP, I forgot to disable the legality check, which previously caused super low efficiency. Plan to provide better code implementation by the end of Jan 2025

图 6. 复现实验结果 N=500

6 总结与展望

这篇文章主要是解决神经分治法的缺陷的，分割策略导致次优解，而作者设计的 reunion（个人理解为错开步幅的 conquer）能起到优化解的效果。同时，采用了 a=50（数据增强）的做法。并且基于该创新点（DCR）的 udc 框架具有优秀的泛化性，可在 10 个大规模 co 问题上有效。

局限性和未来工作：虽然 UDC 已经取得了显著的性能提升，但我们相信通过设计更好的损失函数，UDC 可以取得更好的效果。在未来，我们将关注更适合 UDC 框架的损失，以进一步提高培训效率。此外，我们将尝试扩展 UDC 不适用于当前版本的 CO 问题的适用性。

参考文献

[1] Darko Drakulic, Sofia Michel, Florian Mai, Arnaud Sors, and Jean-Marc Andreoli. Bq-nco: Bisimulation quotienting for efficient neural combinatorial optimization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[2] Chengrui Gao, Haopu Shang, Ke Xue, Dong Li, and Chao Qian. Towards generalizable neural solvers for vehicle routing problems via ensemble with transferrable local policy. 2023.

[3] Stephan Held, Bernhard Korte, Dieter Rautenbach, and Jens Vygen. Combinatorial optimization in vlsi design. In *Combinatorial Optimization*, pages 33–96. 2011.

[4] Qingchun Hou, Jingwei Yang, Yiqiang Su, Xiaoqing Wang, and Yuming Deng. Generalize learned heuristics to solve large-scale vehicle routing problems in real-time. In *The Eleventh International Conference on Learning Representations*, 2023.

[5] Yan Jin, Yuandong Ding, Xuanhao Pan, Kun He, Li Zhao, Tao Qin, Lei Song, and Jiang Bian. Pointerformer: Deep reinforced multi-pointer transformer for the traveling salesman problem. *arXiv preprint arXiv:2304.09407*, 2023.

[6] Yeong-Dae Kwon, Jinho Choo, Byoungjip Kim, Iljoo Yoon, Youngjune Gwon, and Seungjai Min. Pomo: Policy optimization with multiple optima for reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 33, pages 21188–21198, 2020.

[7] Shengcai Liu, Yu Zhang, Ke Tang, and Xin Yao. How good is neural combinatorial optimization? *arXiv preprint arXiv:2209.10913*, 2022.

[8] Fu Luo, Xi Lin, Fei Liu, Qingfu Zhang, and Zhenkun Wang. Neural combinatorial optimization with heavy decoder: Toward large scale generalization. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.