ASP：学习通用神经求解器！

Chenguang Wang、Zhouliang Yu、Stephen McAleer、Tianshu Yu 和 Yaodong Yang

摘要——将机器学习应用于组合优化问题，有望提升效率和准确性。然而，现有的基于学习的求解器在问题分布和规模发生变化时，往往难以实现泛化。本文提出一种名为 ASP 的新方法：**自**适应**阶**梯式**策**略空间响应预言机，旨在解决这些泛化问题并学习通用神经求解器。ASP 由两个部分组成：分布探索，通过策略空间响应预言机增强求解器处理未知分布的能力；持久尺度适应，则通过课程学习提升可扩展性。我们在多个具有挑战性的组合优化问题上测试了 ASP，包括旅行商问题、车辆路径问题和奖品收集旅行商问题，以及来自 TSPLib 和 CVRPLib 的真实世界实例。结果表明，即使模型规模相同且训练信号较弱，ASP 仍能帮助神经求解器探索并适应未见分布和不同规模，从而取得更优性能。具体而言，与采用标准训练流程的相同神经求解器相比，ASP 在旅行商问题的生成实例和真实世界实例上，分别将最优性差距显著降低了 90.9% 和 47.43%；在容量约束车辆路径问题上则分别降低了 19% 和 45.57%。

索引词—组合优化问题、课程学习、策略空间响应预言机、泛化能力与可扩展性

arXiv:2303.00466v1 [cs.LG] 1 Mar 2023

1 引言

COMBINATORIAL 优化问题（COPs）因其广泛的应用，受到运筹学（OR）和理论计算机科学（TCS）领域的高度关注。OR 和 TCS 均开发了多种求解器来处理 COPs，包括精确方法和启发式方法。然而，这些求解器的设计极其耗费人力，且需要丰富的领域知识，因此很难开发出新的求解器。

近年来，COPs 也引起了机器学习领域的关注，因为研究发现，基于深度学习的“神经求解器”能够通过分析大量问题实例，捕捉复杂结构和启发式方法 [1]。研究表明，这些神经求解器在大规模问题上效率很高，而且还能够减轻设计传统求解器所需的繁重工作。然而，当前的神经求解器在两个主要方面存在泛化问题：问题分布和可扩展性。

几乎所有以往的神经求解器都是在定义明确的问题分布和规模上进行训练和测试的，当应用于问题分布和规模可能发生显著变化的现实场景时，性能并不理想 [1], [2], [3], [4], [5]。这限制了神经求解器作为传统求解器替代方案或替代品的应用。针对以往神经求解器所面临的泛化问题，一种潜在解决方案是针对不同设置训练不同的求解器，或者使用能够处理广泛问题实例的超大模型，可以处理广泛的问题实例。然而，这些方法也存在缺点。训练多个求解器可能需要大量计算并消耗大量资源，而使用超大模型可能会牺牲效率，并且在适应不同设置时导致性能下降。因此，我们关注以下问题：

- 我们能否在克服分布问题和规模问题的同时，获得一种容量高效且不牺牲性能的神经求解器？

近期有几项研究尝试从特定角度解决以往组合优化问题神经求解器面临的泛化问题。例如，一些研究着重于在训练期间创建新的分布，以解决数据分布问题 [6], [7],；另一些则探讨了如何使神经求解器适应不同规模 [8]。然而，这些方法都未能提供全面、系统且具有普适性的泛化问题解决方案。

本文提出了一种名为 ASP：自适应阶梯策略空间响应预言机（PSRO）的新方法，旨在解决以往神经求解器面临的泛化问题，并提供一种可应用于广泛问题分布和问题规模的“通用神经求解器”。我们的方法融合了博弈论和课程学习，与传统求解器及以往的神经求解器相比，在多种组合优化问题上展现出更优的性能。具体而言，我们遵循 [6]，提出的双人零和元博弈框架，并采用策略空间响应预言机（PSRO）来适应不同的问题分布和问题规模。我们还引入了阶梯策略 [9]，的概念，使方法能够同时学习多个策略，并更高效地利用模型容量。

- *Chenguang Wang*、*Zhouliang Yu* 和 *TianshuYu* 就职于中国香港中文大学（深圳）数据科学学院。电子邮箱：{*chenguangwang, zhouliangyu*}*@link.cuhk.edu.cn, yu- tianshu@cuhk.edu.cn*
- *Stephen McAleer* 就职于美国卡内基梅隆大学。电子邮箱：*smcaleer@cs.cmu.edu*
- *Yaodong Yang* 就职于中国北京大学人工智能研究院。电子邮箱：*yaodong.yang@pku.edu.cn*

ASP 由两个主要组件构成：分布探索（DE）和持久尺度适应（PSA）。DE 使用策略空间响应预言机（PSRO）使神经求解器适应不同的问题分布，而 PSA 则采用自适应阶梯过程，逐步提高问题规模的难度，使神经求解器能够在广泛的问题规模上持续提升性能。不同于 [6] 通过学习混合高斯扰动并利用蒙特卡洛方法近似策略梯度来生成攻击分布，我们采用 Real-NVP [10], ，这是一种基于归一化流的更通用生成方法，能够以精确的概率生成各种数据分布。

我们使用两种典型的基于强化学习的神经求解器：注意力模型（AM） [2] 和 POMO [5]，在若干经典组合优化问题上测试了 ASP，包括旅行商问题（TSP）和容量约束车辆路径问题（CVRP）。结果表明，与采用标准范式训练的原始求解器相比，ASP 能够取得显著改进，并且在 TSPLib [11] 和 CVRPLib [12]数据集上优于所有其他基于深度学习的神经求解器。此外，我们还进行了消融研究，以验证所提方法的有效性以及模型容量对训练的影响。

总而言之，我们的主要贡献如下：

- 我们提出了 ASP框架，旨在克服以往组合优化问题神经求解器面临的泛化问题。该框架与模型和问题无关，因此可广泛应用于各种神经求解器和组合优化问题。
- 在不增加模型容量的情况下，ASP 通过交替从不同的分布和尺度中学习，使神经求解器能够发现复杂结构；即使训练资源与其他在固定设置下训练的方法相当，也能取得最先进的性能。
- 我们开展了广泛的消融研究，并深入分析了训练策略设计、模型容量、组合优化问题的函数景观等方面。这些发现能够拓展未来组合优化问题领域研究的视野。

2 相关工作

组合优化问题的深度学习- 指针网络 [13]首次尝试使用深度学习解决组合优化问题，具体而言，通过训练一对基于循环神经网络的编码器和解码器，输出其输入的排列。 [14] 采用了类似的模型，但没有像指针网络那样使用监督学习，而是通过将沿节点序列行进的路径长度视为奖励信号，使用强化学习训练模型。受 Transformer 成功的启发， [15]，注意力模型 [2] 采用注意力机制，以类似于图注意力网络 [16],的方式对节点表示进行编码，并解码出各种问题的解决方案。[17] 提出了“学习改进”（L2I）模型，该模型通过改进算子迭代优化解决方案，并利用扰动算子跳出局部最优。同样， [3] 提出了一个强化学习框架用于学习 TSP 和容量约束车辆路径问题（CVRP）的改进启发式算法，并在随机生成实例和真实世界实例上都取得了出色的结果。近期一些研究 [4], [18], [19], [20]专注于预训练模型，并构建热图来引导搜索方法。尽管基于深度学习或强化学习的方法在各种组合优化问题中取得了显著进展，但除泛化问题外，还存在其他一些重要限制，例如在大规模问题上的性能退化，以及训练过程中的高计算开销。最后，我们参考 [21] ，以全面了解该领域现有的挑战。

生成困难实例 - 以往已有一些工作试图通过生成难以求解的实例来获得鲁棒算法 [22], [23], [24]。 [22] 提出了一种生成问题实例空间表示的方法，据此可以量化某些算法的性能，进而推断其泛化能力。利用这些可测量特征，他们能够分析实例之间的异同，以及算法的优势与弱点。与我们的工作密切相关的是， [23] 聚焦于 AdWords 问题，提出了一种受博弈论启发的对抗训练框架，该框架采用生成对抗网络 [25] 生成对抗实例，以暴露给定算法的弱点。 [24] 以对抗方式考虑实例生成和组合构建，以减小训练数据偏差的影响。不同于这些工作，我们采用 PSRO [26] 框架，通过元博弈获得一组可学习的求解器，并设计了一种模型混合方法来组合这些求解器，从而取得最先进的性能。未来，我们将研究应用新的 PSRO 变体以提升性能 [27],[28], [29]。此外，也可以考虑将其他用于双人零和博弈的算法应用于我们的目标 [30], [31], [32]。

课程学习- [33] 首次提出了课程学习（CL）的概念，即机器学习算法应从简单任务逐步训练到困难任务。课程学习具有高度灵活性，可作为即插即用模块，因此已广泛应用于多个研究领域，例如计算机视觉 [34], [35],自然语言处理 [36], [37], 强化学习（RL） [38], [39], [40], [41], [42],等。众多成功的 CL 应用证明，它能够提升目标任务的性能，并加快训练过程的收敛。具体而言，对于强化学习，CL 能够提高样本效率和渐近性能，这有利于探索和泛化[41]。与我们的工作密切相关的是， [7] 引入了“自适应难度”来评估神经求解器的泛化能力，并提出了一种课程学习器，以提升神经求解器对数据分布的泛化能力。 [8]则利用课程学习来提高神经求解器的可扩展性。但这两项工作分别仅关注泛化问题的一个维度： [7]着重于生成相对于数据分布而言的困难实例； [8] 则更关注问题规模。在本研究中，我们将这两个维度结合起来考量。

3 记号与预备知识

标准式博弈（NFG）- 一个元组 $(\Pi, \mathbf{U}^{\Pi}, n)$ 其中，n是玩家数量，$\Pi = (\Pi_1, \Pi_2, ..., \Pi_n)$ 是联合策略集，而$\mathbf{U}^{\Pi} = (\mathbf{U}_1^{\Pi}, \mathbf{U}_2^{\Pi}, ...\mathbf{U}_n^{\Pi})$：$\Pi \to \mathbb{R}^n$ 是每种联合策略对应的效用矩阵。如果所有玩家拥有相同的策略集（$\Pi_i = \Pi_j, i \neq j$）和相同的收益结构，则该博弈是对称的，因此玩家可以互换。

最佳响应- 针对固定对手策略，能够取得最佳期望性能的策略。σ_i^* =br（Π_{-i}，σ_{-i}）是对 σ_{-i} 的最佳响应，条件是：

$$\mathbf{U}_i^{\Pi}(\sigma_i^*, \sigma_{-i}) \geq \mathbf{U}_i^{\Pi}(\sigma_i, \sigma_{-i}), \forall i, \sigma_i \neq \sigma_i^*$$

纳什均衡- 满足以下条件的策略组合 $\sigma^* = (\sigma_1^*, \sigma_2^*, ... \sigma_n^*)$：

$$\mathbf{U}_i^{\Pi}(\sigma_i^*, \sigma_{-i}^*) \geq \mathbf{\Pi}_i(\sigma_i, \sigma_{-i}^*), \forall i, \sigma_i \neq \sigma_i^*$$

直观而言，如果所有参与者都在采用各自的纳什均衡策略，那么任何参与者都没有动机偏离其当前策略。

旅行商问题（TSP）- 目标是寻找一条最短路径，使其仅访问每个位置一次，并返回原始位置。在本文中，我们仅考虑二维欧几里得情形，即每个位置的信息由（x_i，y_i）$\in \mathbb{R}^2$组成。

车辆路径问题（VRP）- 在带容量限制的车辆路径问题（CVRP）中，[43],存在一个仓库节点和若干需求节点，车辆在多条路径中从仓库节点出发并最终返回该节点，同时每条路径上需求节点的总需求不超过车辆容量。CVRP的目标是在满足所有约束的同时，最小化总路径成本。我们还考虑分批配送车辆路径问题（SDVRP），该问题允许将客户需求拆分到多条路径中。

奖品收集旅行商问题 (PCTSP)- 在 PCTSP [44], 中，销售员穿行于给定的二维位置，在每个访问的位置获得奖励，并为未访问的位置支付惩罚。目标是在访问足够多的城市以收集规定数量的奖励的同时，最小化行程长度和惩罚。我们还考虑随机奖品收集旅行商问题 (SPCTSP)，其中位置的预期奖品预先已知，但实际收集到的奖励只有在访问该位置时才会得知。

实例 - 组合优化问题的单个样本。例如，给定 n 点的二维坐标，寻找一条穿过所有点的最短路径，这是 TSP 的一个实例。下文中，我们用 $\mathcal{I}^n$ 表示一个问题规模为 n 的实例，该实例来自某个分布$\mathbf{P}_{\mathcal{I}^n}$。

最优性差距 - 用于衡量求解器相对于最优预言机的质量。给定一个实例 $\mathcal{I}^n$ 和一个求解器 S :$\{\mathcal{I}^n\} \to \mathbb{R}$，最优性差距定义为：

$$g(S, \mathcal{I}^n, \text{Oracle}) = \frac{S(\mathcal{I}^n) - \text{Oracle}(\mathcal{I}^n)}{\text{Oracle}(\mathcal{I}^n)} \tag{1}$$

其中 Oracle$(\mathcal{I}^n)$ 给出该实例的真实最优值。此外，求解器在实例分布$\mathbf{P}_{\mathcal{I}^n}$ 下的期望最优性差距定义为：

$$G(S, \mathbf{P}_{\mathcal{I}^n}, \text{Oracle}) = \mathbf{E}_{\mathcal{I}^n \sim \mathbf{P}_{\mathcal{I}^n}} g(S, \mathcal{I}^n, \text{Oracle}). \tag{2}$$

算法 1 基于 PSRO 的分布探索（**差分进化**）

输入： 神经求解器 S，问题规模 n
初始化： 联合策略集 $\Pi = \{(S^n, \mathbf{P}_{\mathcal{I}^n})\}$，效用矩阵 $\mathbf{U}^{\Pi}$，元策略 $\sigma = (\sigma_{\text{SS}}^n, \sigma_{\text{DG}})$
当 轮次 e 属于 $\{1, 2, ...\}$ **时**
  训练预言机：$(S^{'}, \mathbf{P}_{\mathcal{I}^n}^{'})$ 相对于问题规模 n 公式 5 和公式 8
  更新策略集：$\Pi \leftarrow \Pi \cup \{(S^{'}, \mathbf{P}_{\mathcal{I}^n}^{'})\}$
  从新的 Π 更新 $\mathbf{U}^{\Pi}$，并计算新的元策略 σ
结束 while 循环
输出： 从 Π_{SS} 中选出的最佳求解器 S_{best}^n，并进行混合分布 $\mathbf{P}^{\mathbf{n}} = \sum_i \sigma_{\text{DG},i} \mathbf{P}_{\mathcal{I}^n, i}$

4 方法

本节介绍框架 ASP，旨在获得一个在分布和规模上具有强泛化能力的通用神经求解器。一般而言，ASP包含两个主要组成部分：**(1)** 分布探索（DE）：基于PSRO的训练，用于在特定问题规模下应对分布变化并加以改进；以及**(2)** 持久尺度适应（PSA）：基于自适应阶梯的课程学习，用于增强模型在一系列问题规模上的泛化能力。ASP的概览见图1。下文将详细阐述这两个组成部分。

4.1 分布探索

在本节中，我们将分布探索构建为元层面的双人博弈，以提升神经求解器在分布维度上的泛化能力。在处理问题规模为 n 的组合优化问题实例时，我们假设元博弈中有两名参与者：

- $\Pi_{\text{SS}} = \{S_i^n \mid i = 1, 2, ...\}$ 是求解器选择器的策略集，其中 S_i^n是神经求解器；
- $\Pi_{\text{DG}} = \{\mathbf{P}_{\mathcal{I}^n,\ i},\ i = 1,\ 2,\ ...\}$ 是实例分布策略集，其中$\mathbf{P}_{\mathcal{I}^n,\ i}$ 是问题规模 n下的实例分布。

形式上，我们有一个双人非对称正则形式博弈$(\Pi,\ \mathbf{U}^{\Pi},\ 2)$，$\Pi = (\Pi_{\text{SS}},\ \Pi_{\text{DG}})$ 是联合策略集，$\mathbf{U}^{\Pi}:\ \Pi \to \mathbb{R}^{|\Pi_{\text{SS}}| \times |\Pi_{\text{DG}}|}$ 是效用矩阵。给定一个联合策略 $\pi = (S^n,\ \mathbf{P}_{\mathcal{I}^n}) \in \Pi$，其效用为

$$\mathbf{U}^{\Pi}(\pi) = \left(\mathbf{U}^{\Pi_{\text{SS}}}(\pi), \mathbf{U}^{\Pi_{\text{DG}}}(\pi)\right)$$

其中$\mathbf{U}^{\Pi_{\text{SS}}}(\pi) = -\mathbf{U}^{\Pi_{\text{DG}}}(\pi) = -G(\pi,\ \text{Oracle})$ 是在联合策略 π 下的期望最优性差距，定义如式 2 所示。元层级的总体目标为：

$$\min_{\sigma_{\text{SS}} \in \Delta(\Pi_{\text{SS}}^n)} \max_{\sigma_{\text{DG}} \in \Delta(\Pi_{\text{DG}}^n)} \mathrm{E}_{\pi \sim (\sigma_{\text{SS}}, \sigma_{\text{DG}})} G\big(\pi, \text{Oracle}\big). \tag{3}$$

其中 $\Delta(\cdot)$ 表示给定集合上的分布。我们按如下方式遵循 PSRO 框架：在每次迭代中，给定策略集 $\Pi = (\Pi_{\text{SS}},\ \Pi_{\text{DG}})$ 和元策略 σ =$(\sigma_{\text{SS}},\ \sigma_{\text{DG}})$，我们训练两个预言机：

- $S^{'}$ 表示一个新的神经求解器，它是对元策略σDG的最佳响应。• $\mathbf{P}_{\mathcal{I}^n}^{'}$ 表示一个新的实例分布，在该分布下，求解器选择器遵循σ_{SS}时表现不佳。

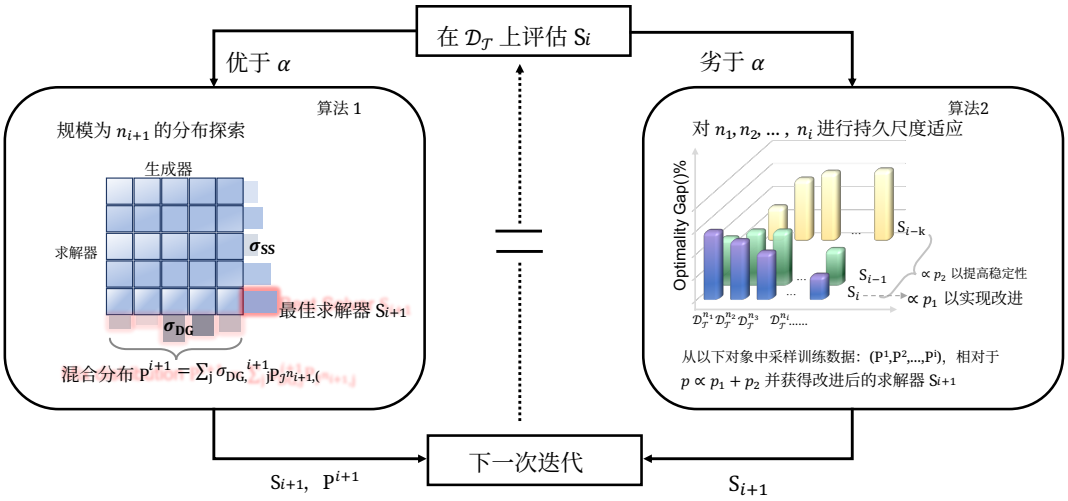

图1：ASP框架的流程。在第 i次迭代中，ASP根据当前神经求解器在评估集上的性能是否优于预设阈值 α，选择在更大的问题规模 n_{i+1} 上提升泛化能力，或增强其在一组预先选定的问题规模上的泛化能力。如果性能优于 α，我们将在规模 n_{i+1}上执行DE 流程，如算法 1所详述，并根据元策略获得最佳求解器 S_{i+1} 和混合数据分布P^{i+1} 。否则，我们通过在问题规模 n_1、n_2、……、n_i上进行训练，采用PSA过程来提升 S_i的能力，如算法2所详述。为了从问题规模 n_j下的P^j中采样训练数据，我们提出了一种采样策略（公式12），该策略基于评估集上的当前性能和历史性能，在改进倾向与稳定性之间取得平衡。随后得到一个增强的求解器 S_{i+1}，用于下一次迭代。

给定这些预言机，我们根据新的 Π更新联合策略集 $\Pi \leftarrow \Pi \cup (S', \mathbf{P}'_{\mathcal{I}_n})$ 以及元博弈 $\mathbf{U}^{\Pi}$。扩展联合策略集具有双重目的：既有助于找到难以求解的实例分布，也能提升求解器选择器处理这些实例的能力。总体算法框架见算法 1。

上述公式还留下了三个需要处理的算法组件：**(1)** 如何获得元策略σ；**(2)** 如何训练每位玩家的预言机；**(3)** 如何评估效用$\mathbf{U}^{\Pi}$。在接下来的部分中，我们将描述算法的流程。

元策略求解器 -鉴于我们在式 3 中将其表述为一个双人博弈，采用元博弈的纳什均衡作为元策略解更为合适，因为这种方法计算效率更高。

预言机训练 - 现在，我们给出训练最佳响应预言机的高层结果，更详细的推导见附录 A。这里，我们将 Π_{SS} 中的神经求解器表示为 S_θ，将Π_{DG} 中的实例分布表示为$\mathbf{P}_{\mathcal{I}^n,\gamma}$，其中 θ 和 γ 是可训练参数。

- 对于求解器预言机：给定元策略σDG，神经求解器的预言机训练目标为：

$$\min_{\theta} L_{SS}(\theta) = \mathbf{E}_{\mathbf{P}_{\mathcal{I}^n} \sim \sigma_{DG}} G(S_\theta, \mathbf{P}_{\mathcal{I}^n}, \text{Oracle}). \quad (4)$$

我们可以通过随机梯度下降来优化这一目标，其梯度为：

$$\nabla_\theta L_{SS}(\theta) = \mathbf{E}_{\mathbf{P}_{\mathcal{I}^n} \sim \sigma_{DG}} \mathbf{E}_{\mathcal{I}^n \sim \mathbf{P}_{\mathcal{I}^n}} \frac{\nabla_\theta S_\theta(\mathcal{I}^n)}{\text{Oracle}(\mathcal{I}^n)}. \quad (5)$$

鉴于在训练期间不可能在公式 5 中调用“Oracle”进行过多次迭代，我们可以改为训练求解器 Oracle，以获得给定分布$\mathbf{P}_{\mathcal{I}}$ 下功能强大的神经求解器，其梯度为：

$$\nabla_\theta L_{SS}(\theta) = \mathbf{E}_{\mathbf{P}_{\mathcal{I}^n} \sim \sigma_{DG}} \mathbf{E}_{\mathcal{I}^n \sim \mathbf{P}_{\mathcal{I}^n}} \nabla_\theta S_\theta(x_1, \ldots, x_n). \quad (6)$$

- 对于数据生成器预言机：给定元策略σ_{SS}，DG 的预言机训练目标为：

$$\max_{\gamma} L_{DG}(\gamma) = \mathbf{E}_{S \sim \sigma_{SS}} G(S, \mathbf{P}_{\mathcal{I}^n,\gamma}, \text{Oracle}). \quad (7)$$

关于 γ 的梯度为：

$$\nabla_{\gamma_C} L_{DG}(\gamma) = \mathbf{E}_{S^n \sim \sigma_{SS}} \mathbf{E}_{\mathcal{I}^n \sim \mathbf{P}_{\mathcal{I}^n,\gamma}} \Big[\nabla_\gamma \big(\log \mathbf{P}_{\mathcal{I}^n,\gamma}(\mathcal{I}^n) \big) g(S, \mathcal{I}^n, \text{Oracle}) \Big]. \quad (8)$$

与神经求解器的训练预言机类似，我们也可以省略对 Oracle 求解器的调用。否则，为了准确计算式 8 中的概率$\mathbf{P}_{\mathcal{I}^n,\ \gamma}(\mathcal{I}^n)$，我们应用 Real-NVP [10],这一基于归一化流的生成模型来生成对抗数据。具体来说，我们从一个简单的先验概率分布开始

$$z \sim p_Z = Uni.([0,1]^k) \quad (9)$$

以概率 $p_Z(z) = 1, \forall z \in [0,1]^k$ 其中k 是输入特征的维度。我们通过参数化双射获得对抗数据

$$x = G_\gamma(z) : Z \to X,$$

那么，获得 x 的确切概率由变量替换公式给出：

$$p_X(x) = p_Z(G^{-1}(x))\left|\det\left(\frac{\partial G_\gamma^{-1}}{\partial x}\right)\right| = \left|\det\left(\frac{\partial G_\gamma^{-1}}{\partial x}\right)\right|.$$

得益于 f_γ 在 Real-NVP 中的构造，我们能够高效且精确地计算该概率，因此可以通过这种方式计算概率 $\mathbf{P}_{\mathcal{I}^n,\ \gamma}(\mathcal{I}^n)$ 。

评估 - 给定联合策略 $\pi \in \Pi$，我们可以通过近似式 2 中定义的期望最优性差距，计算效用矩阵中的元素 $\mathbf{U}^{\Pi}(\pi) = (-G(S,\ \mathbf{P}_{\mathcal{I}^n},\ \text{Oracle})\ \ G(S,\ \mathbf{P}_{\mathcal{I}^n},\ \text{Oracle}))$:

$$\begin{aligned} G(S, \mathbf{P}_{\mathcal{I}^n}, \text{Oracle}) &= \mathbf{E}_{\mathcal{I}^n \sim \mathbf{P}_{\mathcal{I}^n}} g(S, \mathcal{I}^n, \text{Oracle}) \\ &\approx \frac{1}{M}\sum_{i=1}^{M} g(S, \mathcal{I}_i^n, \text{Oracle}). \end{aligned} \tag{10}$$

其中 $S \in \Pi_{\text{SS}}$，$\mathbf{P}_{\mathcal{I}^n} \in \Pi_{\text{DG}}$。

完成问题规模 n下 PSRO 的训练后，我们得到最佳神经求解器 S_{best}^n 以及一个混合分布 $\mathbf{P^n} = \sum_i \sigma_{\text{DG},\ i}\mathbf{P}_{\mathcal{I}^n,\ i}$，以供后续使用。

4.2 持续尺度自适应

算法2 持续尺度自适应任务选择 (**PSA**)

输入： 神经求解器 S，一系列问题规模 n_1，...，n_i 以及混合分布 $\mathbf{P^1}$，...，$\mathbf{P^i}$
初始化： $p_0 \in \mathbb{R}^i$，使用均匀概率向量
当 轮次 t 属于 $\{1, 2, ...\}$ **时**
  根据公式 10，在每个 $(\mathbf{P^k},\ n_k)$ 上评估 S，以获得成本向量 $c^t = (c_k^t)_k$
  计算任务选择策略 p_t，如公式所示。
  12，并针对公式 11 训练新的神经求解器 S'，其中 $P_{\text{PS}} = p_t$
  分配 $S \leftarrow S'$
结束 while
输出： 新训练的神经求解器 S。

考虑一系列问题规模 （n_1、n_2、.....、n_i），我们希望获得一个在全部问题规模上都具有持久性能的通用神经求解器。在本节中，我们在课程学习框架下提出持久尺度适应（PSA）过程来实现这一目标。具体而言，我们可以将其表述为一个多目标问题：

$$\min_\theta L_{\text{PSA}}^M(\theta) = \big(G(S_\theta, \mathbf{P}^{n_1}, \text{Oracle}), ..., G(S_\theta, \mathbf{P}^{n_i}, \text{Oracle})\big).$$

一种常见的优化方法是为每个目标分配一个权重，将其转化为单目标问题，从而得到期望形式：

$$\min_\theta L_{\text{PSA}}(\theta) = \mathbf{E}_{n \sim P_{\text{PS}}} G(S_\theta, \mathbf{P}^n, \text{Oracle}). \tag{11}$$

其中，P_{PS} 是任务选择策略，即问题规模上的离散概率分布 $(n_1,\ n_2,\ .....,\ n_i)$，满足$P_{\text{PS}}(n = n_k) = p_k$，$\sum_k p_k = 1$。 P_{PS}有许多候选形式，例如原始自适应阶梯过程中使用的均匀分布，但这会误导神经求解器在不适当任务上的训练。在此，我们提出一种基于动量的机制，引导神经求解器专注于适当的任务。

我们将采样概率 $\{p_i\}$ 视为一种弱信号，用于引导训练过程朝着改进目标分布 $\mathcal{D}_{\mathcal{T}}$的方向进行。通过在每一对 （$\mathcal{D}_{\mathcal{T}}$、$n_k$）上评估神经求解器 S_θ，我们可以获得一个成本向量 $c \in \mathbb{R}^i$:

$$c = (c_k)_k, \text{where } c_k \text{ is evaluated by Eq. 10,}$$

用于衡量 S_θ 在问题规模 n_k下对数据分布$\mathcal{D}_{\mathcal{T}}$ 的性能。在实施过程中，我们将执行PSA过程，直到满足某些条件（例如达到预设性能，或进行若干次迭代的训练），因此我们将第 t次PSA迭代时的成本向量记为 $c^t = (c_k^t)_k$，并按如下方式构造 $\{p_i^t\}$：

$$p_i^t \propto \lambda p_i^1 + (1-\lambda)p_i^2, \lambda \in [0, 1] \tag{12}$$

其中，若 $x > 0$，则 $p_i^1 \propto \lfloor c_i^t - c_i^{t-1} \rfloor_+$ with $\lfloor x \rfloor_+ = x$；否则为 0；$p_i^2 \propto \sqrt{Var_t[c_i^t]}$。 p_i^1有助于我们关注神经求解器表现不佳的问题规模，而 p_i^2 衡量历史性能的稳定性，从而提升训练过程的稳定性。 λ 是用于平衡这两项的权重。

由于 c^t 事先未知，且会随训练时间而变化，我们会保留之前 10 步的成本向量历史记录，并在训练过程中动态更新 $Var_t[c_i^t]$ ，以便引导训练过程重点关注性能不理想且不稳定的任务。在训练开始时，我们均匀采样 n_i。整个过程见算法 2。

4.3 ASP 的完整流程

本节将整合前面各部分的内容，详细介绍自适应阶梯式 PSRO（ASP）框架的整体方案。

借鉴课程学习的理念，从简单任务开始并逐步提高难度，我们从最小的问题规模开始训练神经求解器，因为在组合优化问题的背景下，规模较大的问题被认为比规模较小的问题更难。受自适应阶梯过程启发，在每次 ASP 迭代中，我们考虑两种可选流程：算法 1 中描述的 DE 流程和算法2中描述的 PSA 过程。给定迄今已处理的问题规模（n_1、n_2、.....、n_i）以及迄今训练得到的神经求解器 S_θ，如果 S_θ 在目标分布 $\mathcal{D}_{\mathcal{T}}$ 上的评估结果优于预设阈值 α，我们便在更大的问题规模 n_{i+1} 上执行 DE 流程，这意味着迈上更高的阶梯，处理更困难的任务；否则，我们会在（n_1、n_2、.....、n_i）上执行 PSA过程，以增强 S_θ 处理当前任务的能力。我们重复这一过程，直到满足某个停止准则。整个流程如算法 3 所示。

5 实验

本节介绍我们在若干组合优化问题上的结果：旅行商问题、容量约束车辆路径问题、拆分配送车辆路径问题、PCTSP 和 SPCTSP。与以往工作不同，以往工作由于

算法 3 ASP（自适应式阶梯PSRO）

输入： 神经求解器 S，最小和最大问题规模 n_1，n_K，增量步长 n_{step}，评估数据集 $D = \{\mathcal{D}_{\mathcal{T}}^{n_1}, \mathcal{D}_{\mathcal{T}}^{n_1+n_{\text{step}}}, ..., \mathcal{D}_{\mathcal{T}}^{n_K}\}$ 和性能阈值 α，训练耐心值 γ

初始化： 问题规模列表 $p_l = [n_1]$ S_{best}，$\mathbf{P^1} =$ 通过算法 1 对 S，n_1 进行**差分进化**，混合分布列表 $p_{\text{dist}} = [\mathbf{P^1}]$，计数$= 0$，在 $\mathcal{D}_{\mathcal{T}}^{n_1}$ 上评估 S_{best}，得到评估结果 r

当为真 **时执行**

  如果 ($r \leq \alpha$ 或计数$> \gamma$) 且 $p_l[-1] \leq n_K$ **则**

    $n' = p_l[-1] + n_{\text{step}}$

    如果 $n' < n_K$ **则**

      $p_l \leftarrow p_l + [n']$

    else

      跳出

    结束 if

    S_{best},$\mathbf{P}^{\text{new}}$ = **差分进化**($S_{\text{best}}, p_l[-1]$) 由算法 1

    $p_{\text{dist}} \leftarrow p_{\text{dist}} + [\mathbf{P}^{\text{new}}]$

    计数=0

  else

    S_{best} = **PSA**($S_{\text{best}}, p_l, p_{\text{dist}}$) 由算法 2

    计数+=1

  结束条件判断

  在 $\{\mathcal{D}_{\mathcal{T}}^{n_1}, ..., \mathcal{D}_{\mathcal{T}}^{p_l[-1]}\}$ 上评估 S_{best}，得到评估结果 r

结束循环

输出： 最佳神经求解器 S_{best}

在相同的分布（均匀分布）上进行训练和测试，且针对每种问题规模训练一个模型，因此未能展现泛化能力；我们则展示了模型在从未训练过的分布上的性能，并在各种问题规模上评估同一个模型。

5.1 实验设置

基础求解器 - 我们的方法与模型和问题无关，因此我们在各种组合优化问题上采用现有的神经求解器，以证明该框架的有效性。我们将未经过我们的方法训练的神经求解器称为基础求解器，并将使用我们的方法的神经求解器标注为基础求解器(ASP)。

COP	基础求解器	阈值
TSP	注意力模型 [2]	3
	POMO [5]	0.5
CVRP	注意力模型 [2]	5
	POMO [5]	1
拆分配送车辆路径问题	注意力模型 [2]	10
PCTSP	注意力模型 [2]	10
SPCTSP	注意力模型 [2]	10

表 1：各 COP 对应的基础求解器和阈值

训练设置 - 不同基础求解器和组合优化问题的所有训练均从头开始。考虑到神经求解器在不同组合优化问题上的性能各异，每种组合优化问题所选用的基础求解器以及性能阈值 α 的设置也有所不同，如表 1 所示。对于公式 12 中的权重，我们在全部实验中均使用 $\lambda = 0.5$。对于所有组合优化问题，我们将问题规模的最小值和最大值设为 $n_1 = 20$，$n_K = 100$，并将增量步长设为 $n_{\text{step}} = 20$。在训练过程中，我们会在目标分布 $\mathcal{D}_{\mathcal{T}}$ 下，对从未训练过的问题规模对应的神经求解器进行评估，并获得平均评估结果 r。我们还设置了耐心参数 $\gamma = 5$，以避免神经求解器停滞在当前任务上；也就是说，如果 PSA过程经过 5 次训练迭代后仍未观察到进一步改进，我们就会转向难度更高的任务。

数据生成器是一个基于归一化流的模型 Real-NVP [10]，包含 5 个耦合层；我们使用 Adam优化器 [45] 训练其参数，学习率为 1e-4，训练 100 轮。此外，在开始针对更大问题规模进行 DE训练时，我们会重置参数。基础求解器和数据生成器的模型结构及训练配置的详细设置见附录 B。

基线 – 对于旅行商问题，我们将框架训练的神经求解器与精确求解器、非学习型启发式算法以及基于学习的方法进行比较，评估其解的质量和推理速度。具体而言，对比求解器包括 Concorde、Gurobi [46]、LKH3 [47]（一种最先进的启发式求解器）以及基于元启发式算法的近似求解器 OR-Tools。此外，我们还与 Nearest、Random 和最远插入法进行比较。我们还将所提方法与多种基于学习的方法进行比较：LIH [17]、注意力模型 [2]、MDAM [48]、POMO [5]、图卷积网络 [49] 和 CVAE-Opt [50]。对于容量约束车辆路径问题，我们将 ASP 训练的神经求解器与以下求解器进行性能比较：LKH3、不同时间限制下的 Gurobi，以及以下基于学习的方法：注意力模型、POMO、RL(gr.) [51]、NeuRewriter [52] 和 CVAE-Opt。对于拆分配送车辆路径问题，我们与 RL(gr.)、注意力模型 [2],和 MDAM[48]进行比较。对于 PCTSP，我们与 Gurobi、OR-Tools 以及迭代局部搜索（ILS）的 Python 版本进行比较。此外，我们还将神经求解器 AM(ASP) 与其基线注意力模型 [2]进行比较。对于 SPCTSP，我们将神经求解器 AM(ASP) 与其基线 AM 和 MDAM 进行比较。

时间消耗——训练时间是神经求解器的一个重要方面。应用 ASP 时，除神经求解器的常规训练消耗外，还会额外耗费时间用于训练数据生成器以及在 DE 中评估元收益。所有实验均在单张 GeForce RTX 3090 上进行。在我们的设置下，在 TSP 和 CVRP 上训练 POMO 分别需要 1.67 天和 2.25 天；在 TSP 和 CVRP 上训练注意力模型分别需要 4 天和 3.3 天。在针对其他组合优化问题训练注意力模型时，一旦不再有进一步改进，我们便提前停止训练，因此训练时长约为 2 天。与基础求解器的训练成本相比，在 ASP 下训练神经求解器在问题规模为 50 时所需时间略多于该基础求解器，而在问题规模为 100 时则少得多。

5.2 生成数据上的结果

数据生成 – 对于旅行商问题，我们通过从单位正方形中随机采样 $x \in \mathbb{R}^2$，并从 N($\mathbf{0}, \Sigma$) 中采样 $y \in \mathbb{R}^2$来生成数据，其中 $\Sigma \in \mathbb{R}^{2\times2}$ 是一个对角矩阵，其元素从 $[0, \lambda]$ 和 $\lambda \sim$ U(0,1)中采样。接下来，通过 $z = x + y$生成二维坐标，我们可以通过执行这种采样 n来获得任意规模的旅行商问题 n

表2：我们的方法与基线方法的对比。差距百分比是相对于所有方法中的最佳值计算的。粗体数字表示基础求解器（ASP）比基础求解器取得了更好的性能，下划线数字表示所有启发式方法中的最佳性能。我们尤其关注平均差距，因为相比于单个问题规模，我们更关注不同方法在多个问题规模上的性能。

	方法	$n=20$ Gap	$n=20$ Time	$n=50$ Gap	$n=50$ Time	$n=100$ Gap	$n=100$ Time	平均差距
TSP	Concorde	0。00%	(18.50s)	0。00%	(46.38s)	0。00%	(34.52s)	0。00%
	LKH3	0。00%	(3.01s)	0。00%	(50.79s)	0。00%	(1.83m)	0。00%
	Gurobi	0。00%	(1.24s)	0。00%	(30.34s)	0。00%	(2.16m)	0。00%
	OR-Tools	1。96%	(10.47s)	3.50%	(1.07m)	4。16%	(5.11m)	3.21%
	最远插入法	2.91%	(1.41s)	6。01%	(5.73s)	7.49%	(9.09s)	5.61%
	随机插入法	5。10%	(0.99s)	8。33%	(3.24s)	10.09%	(5.85s)	8.16%
	最近插入法	10。94%	(1.48s)	17。59%	(5.09s)	20.19%	(9.49s)	16。66%
	-							
	LIH(T=1000)	14。49%	(28.24s)	145.03%	(36.73s)	300。00%	(2.05m)	176。64%
	图卷积网络(gr.)	17。82%	(17.30s)	75.92%	(18.55s)	52。92%	(1.01m)	48.37%
	图卷积网络(bs)	16。16%	(1.03m)	74。36%	(1.32m)	48.05%	(4.97m)	46.70%
	MDAM（gr.）	0.72%	(9.61s)	12.57%	(32.36s)	13.97%	(12.06m)	9。09%
	MDAM（bs.）	0。65%	(52.18s)	5.56%	(20.95m)	12。99%	(38.87m)	5.91%
	CVAE-Opt	2.90%	(1.39m)	7。33%	(3.01m)	3.62%	(4.74m)	4。62%
	AM	5.95%	(0.02s)	65。77%	(0.04s)	53。30%	(0.12s)	41。67%
	AM(ASP)	**1.81**%	(0.03s)	**3.98**%	(0.07s)	**5.60**%	(0.16s)	**3.79**%
	POMO	0。01%	(0.45秒)	0.42%	(1.43秒)	2.81%	（5.67秒）	1.09%
	POMO(ASP)	0.09%	（0.45秒）	0.59%	(1.44秒)	**1.67**%	(5.67秒)	**0.78**%
CVRP	LKH3	0。00%	(4.81m)	0。00%	(8.23m)	0。00%	(16.09m)	0。00%
	Gurobi(10秒)	1。74%	(11.69s)	33.08%	(12.15s)	40。44%	(12.70s)	32.38%
	Gurobi(50秒)	1。39%	(51.80s)	14.93%	(53.80s)	32。58%	(53.65s)	14。87%
	Gurobi(100秒)	2。78%	(1.70m)	10.20%	(1.72m)	21。54%	(1.78m)	10.82%
	RL(gr.)	42。51%	(2.72秒)	14。42%	(5.04秒)	47.94%	(6.12秒)	39。54%
	强化学习(bs.)	34。84%	(6.06s)	9.95%	(28.42s)	41。57%	(35.48秒)	33。67%
	MDAM（gr.）	3。48%	(3.95秒)	11.95%	(6.15秒)	12。49%	(18.18秒)	9.31%
	MDAM（bs.）	1。74%	(9.48秒)	9。19%	(24.19秒)	10.45%	(1.02分钟)	7。13%
	CVAE-Opt	5.57%	(4.08m)	9.43%	(3.71m)	16.79%	(3.92m)	10.60%
	LoRew	29。95%	(0.34s)	84。70%	(2.64s)	152。41%	(5.77s)	85.08%
	AM	5.72%	(0.02s)	11。35%	(0.06s)	18.09%	(0.11s)	11。72%
	AM(ASP)	9。48%	(0.04s)	**7.73**%	(0.11s)	**11.27**%	(0.21s)	**9.49%**
	POMO	0.32%	(0.04s)	1。27%	(0.14s)	3。00%	(0.51 秒)	1。53%
	POMO(ASP)	0.37%	(0.04 秒)	**1。26**%	(0.11秒)	**2。79**%	(0.44秒)	**1.47%**
SDVRP	MDAM（gr.）	3.83%	(4.17s)	10.80%	(9.37s)	16.79%	(21.59s)	10.47%
	MDAM（bs.）	2。43%	(11.01s)	8。27%	(23.61s)	13。43%	(1.30m)	8.03%
	AM	6.20%	(0.03秒)	26.79%	(0.08秒)	19.56%	(0.15秒)	17.52%
	AM(ASP)	13。32%	（0.05秒）	**8.34**%	（0.13秒）	**11.88**%	（0.26秒）	**11。18**%
PCTSP	Gurobi	0。00%	(0.30秒)	0。00%	(0.95秒)	0。00%	(2.98m)	0。00%
	OR-Tools	2.20%	(11.06s)	5.33%	（2.00米）	8。33%	（6.00米）	5.30%
	ILS（Python 10 倍）	63。23%	（3.05秒）	148。05%	（4.70秒）	209.78%	（5.27秒）	137。94%
	-							
	MDAM（gr.）	11。76%	(41.10s)	24。73%	(1.31m)	30。07%	(1.96m)	22。19%
	MDAM（bs.）	5.88%	(2.70m)	18.81%	(4.77m)	26。09%	(6.97m)	16。93%
	AM	2.88%	(0.02秒)	17.95%	(0.06秒)	29。24%	(0.14秒)	16。69%
	AM(ASP)	12.05%	(0.03秒)	**10.34**%	(0.08秒)	**11.56**%	(0.18秒)	**11。32**%
SPCTSP	MDAM（gr.）	11。76%	(20.33s)	24.73%	(11.36s)	30.07%	(16.97s)	22。19%
	MDAM（bs.）	7.35%	(1.06m)	20.43%	(2.02m)	26.81%	(3.10m)	18.20%
	AM	3.60%	(0.02秒)	15.68%	(0.06秒)	33.09%	(0.13秒)	17.45%
	AM(ASP)	10.74%	(0.04s)	**7.69**%	(0.07s)	**8.16**%	（0.16秒）	**8.86**%

次。我们采样了 10000 个归一化实例，组成 10 组由不同 λ 值生成的数据。对于

容量约束车辆路径问题中，二维坐标的生成方式与旅行商问题相同，需求 δ 为离散值，并从均匀分布中采样

从 $\{1, ..., 9\}$ 中采样，容量为 $D^{20} = 30$，$D^{30} = D^{40} = D^{50} = 30$，$D^{60} = D^{70} = ... = D^{100} = 50$，最终需求归一化为 $\hat{\delta} = \frac{\delta}{D}$。评估规模为 128。对于 PCTSP 和 SPCTSP，最大长度为$L^{20} = 2$，$L^{30} = L^{40} = L^{50} = 3$，$L^{60} = L^{70} = ... = L^{100} = 4$。随后，我们在注意力模型 [2], 中对奖励和惩罚采用相同的生成设置：二维坐标的生成方式与旅行商问题相同，评估规模为 1000。问题规模为 20、50、100 时的总体结果见表2。

备注– 值得一提的是，开头所述的比较情形：我们在不同问题规模上评估同一个基础求解器(ASP)，而其他神经求解器则使用在各自问题规模上单独训练的模型进行评估。也就是说，我们使用一个通用神经求解器，其性能优于多个基础求解器（本例中为 3 个）。

TSP 上的结果– 我们将多种基线方法与几个神经求解器进行比较。首先，在均匀分布上训练的神经求解器在未见分布上的性能大幅下降。具体而言，就平均差距而言，LIH [17], GCN [49], MDAM [48]和 AM [2] 的性能甚至不如经典启发式方法，例如最远插入法和随机插入法。这尤其令人不满意，因为神经求解器需要额外的训练成本。CVAE-Opt [50] 和 POMO [5] 的表现相对稳定。比较基础求解器和我们的方法时，与 AM 和 POMO 相比，AM(ASP) 和 POMO(ASP) 的差距分别大幅降低90.9% 和49.4%，而耗时几乎没有进一步增加[1]。此外，除精确方法外，POMO(ASP) 在所有求解器中取得了最佳结果。

VRP 上的结果– 对于 CVRP，与经典求解器相比，神经求解器在时间消耗方面具有显著优势。在解的质量方面，除 POMO [5] 外，神经求解器仍然受到数据分布的严重影响。比较基础求解器与我们的方法可以发现，在不增加时间消耗的情况下，AM(ASP) 和 POMO(ASP) 相较于 AM 和 POMO 分别降低了 19% 和 5% 。此外，即使在解的质量和时间消耗方面与 Gurobi（10s、50s、100s） [46] 以及 OR-Tools [53] 进行比较，POMO(ASP) 仍在所有这些基线方法中取得了最佳性能，展现出神经求解器处理更复杂任务的潜力。对于 SDVRP，AM(ASP) 相较于 AM 也有 36.2% 的改进。

PCTSP 上的结果 – 与经典求解器和启发式算法相比，神经求解器在时间消耗方面仍具有优势。在解的质量方面，AM 和 AM(ASP) 大幅优于迭代局部搜索（ILS）。此外，AM(ASP) 的性能优于 AM，降低了 32.2% 。与 SPCTSP 上的结果类似，AM(ASP) 的性能超过 AM 29.2%，表明其对实际问题具有更强的泛化能力。

5.3 真实世界数据上的结果

我们在表 3–4 中展示了真实数据集上的实验结果。请注意，AM 和 POMO 均在问题规模为 100 的条件下进行训练，因此所需训练时间远多于 AM(ASP) 和 POMO(ASP)。

表 3：TSP 的 TSPlib 实例结果。带下划线和加粗的数字分别表示结果优于 OR-Tools 和原始基础求解器。

实例	Opt.	OR-Tools	AM	AM(ASP)	POMO	POMO(ASP)
pr226	80,369	82,968	86,200	**83,570**	81,903	83,117
ts225	126,643	128,564	143,994	**140,783**	134,184	138,846
kroD100	21,294	21,636	21,913	**21,817**	21,878	**21,508**
eil51	426	436	434	438	428	429
kroA100	21,282	21,448	23,500	**21,573**	21,691	**21,332**
pr264	49,135	51,954	60,952	**57,398**	51,812	55,643
pr152	73,682	75,834	76,984	77,008	74,088	75,086
rat99	1,211	1,232	1,329	**1,298**	1,273	**1,221**
kroA150	26,524	32,474	29,674	**27,799**	27,087	**26,982**
lin105	14,382	14,824	16,423	**14,640**	14,853	**14,531**
pr124	59,030	62,519	62,350	**59,981**	59,385	59,399
st70	675	683	701	**677**	677	682
a280	2,586	2,713	3,127	**2,913**	2,844	2,891
rd100	7,910	8,189	8,191	8,316	7,917	7,923
pr136	96,772	102,213	103,878	**100,927**	98,183	99,671
pr76	108,159	111,104	109,958	**109,123**	108,280	108,328
kroA200	29,368	29,714	33,033	**31,673**	29,946	30,313
kroB200	29,437	30,516	32,501	**31,309**	30,999	**30,499**
pr107	44,303	45,072	48,115	**47,321**	44,856	45,222
kroB150	26,130	27,572	27,837	**26,925**	26,680	**26,441**
u159	42,080	45,778	45,302	**43,178**	42,716	42,768
berlin52	7,542	7,945	8,483	**7,593**	7,548	**7,544**
rat195	2,323	2,389	2,855	**2,699**	2,506	2,527
d198	15,780	15,963	24,324	**18,482**	19,864	**17,737**
eil101	642	664	663	**659**	643	643
pr144	58,537	59,286	62,038	**59,702**	59,173	59,173
pr299	48,191	48,447	60,506	**56,955**	52,807	53,213
kroC100	20,750	21,583	22,491	**20,896**	20,906	**20,753**
tsp225	3,859	4,046	4,563	**4,361**	4,146	4,180
eil76	538	561	562	**555**	549	553
kroB100	22,141	23,006	23,293	**22,923**	22,336	22,376
kroE100	22,068	22,598	23,311	**22,733**	22,415	**22,183**
ch150	6,532	6,729	6,851	**6,756**	6,590	6,609
bier127	118,282	137,893	128,308	**121,687**	124,391	**122,874**
ch130	6,110	6,284	6,452	**6,218**	6,137	6,137
平均差距（%）	0	4.19	10.69	**5.62**	3.37	**3.23**

TSP 上的结果– 我们进一步在 TSPLib [11] 上验证了通过 ASP 训练的神经求解器。如表 3 所示，AM(ASP) 在几乎所有实例上都优于 AM，平均最优性差距降低了 5%。具体而言，对于以下几个实例：lin105、st70、pr136、kroB150、berlin52、eil101、kroC100、eil76 和 ch130，ASP 可以帮助 AM 超越 OR-Tools。对于 POMO [5], , ASP 仍能在一定程度上对其进行改进。此外，POMO(ASP) 优于所有列出的方法，并且具有更小的平均最优性差距。上述结果表明，通过 ASP 训练的神经求解器能够通过泛化到复杂分布和一系列问题规模，处理困难的真实实例。

CVRP上的结果– 我们将 ASP 训练的神经求解器的解质量，与基础求解器 AM[2], POMO [5], 和 OR-Tools 在 CVRPLib [12] B、E、F 和 X 的真实世界实例上的表现进行了比较，这些实例涵盖的问题规模范围为 22 至 181。在这些真实实例中，就平均最优性差距而言，AM(ASP) 的性能优于 AM。神经求解器 POMO(ASP) 在所选 CVRPLib 实例中的约70% （53 个实例中的 37 个）上优于 POMO [5] ，并且取得了更小的平均差距3.32%，优于基线6.10%。此外，POMO(ASP) 在 53 个实例中的 47 个上也取得了优于 OR-Tools 的性能，平均差距降低了65% 。

根据真实基准测试的结果，我们可以得出结论：在模型容量相同且训练时间大幅缩短的情况下，经过 ASP 训练的神经求解器能够显著提升泛化能力。

1. AM(ASP) 的时间开销略高于 AM，因为我们使用了更大的模型容量。原因见第 6.4 节和 B

表 4：CVRP 的 CVRPlib 实例结果。带下划线和粗体的数字分别表示结果优于 OR-Tools 和原始基础求解器。

实例	Opt.	OR-Tools	AM	AM(ASP)	POMO	POMO(ASP)
B-n52-k7	747	759	806	850	753	755
B-n66-k9	1,316	1,390	1,437	**1,418**	1,335	**1,331**
B-n68-k9	1,272	1,389	1,316	1,419	1,301	1,313
B-n45-k6	678	769	748	**735**	717	**689**
B-n50-k8	1,312	1,335	1,378	**1,358**	1,329	1,359
B-n57-k9	1,598	1,689	1,674	**1,647**	1,607	1,617
B-n51-k7	1,032	1,150	1,034	1,165	1,030	**1,025**
B-n50-k7	741	748	788	857	748	764
B-n39-k5	549	590	577	645	556	**554**
B-n31-k5	672	677	717	**702**	729	**681**
B-n63-k10	1,496	1,627	1,633	**1,597**	1,540	1,579
B-n64-k9	861	1,105	1,010	**915**	910	924
B-n67-k10	1,032	1,105	1,144	1,146	1,101	**1,070**
B-n41-k6	829	861	862	940	843	**842**
B-n44-k7	909	948	1,030	**993**	934	950
B-n56-k7	707	779	742	766	715	**721**
B-n34-k5	788	798	825	864	803	**796**
B-n35-k5	955	996	1,097	**975**	976	**974**
B-n45-k5	751	804	864	**812**	763	**762**
B-n38-k6	805	881	847	852	820	**817**
B-n78-k10	1,221	1,299	1,336	**1,313**	1,255	1,265
B-n43-k6	742	771	830	**770**	763	**747**
E-n22-k4	375	375	441	**408**	417	**375**
E-n23-k3	569	570	620	620	702	**572**
E-n33-k4	835	937	855	956	870	**839**
E-n51-k5	521	587	582	**572**	553	**532**
E-n76-k7	682	740	718	764	702	**700**
E-n76-k8	735	824	790	822	750	**747**
E-n76-k10	830	992	880	926	853	**851**
E-n76-k14	1021	1,586	1,110	1,120	1,051	**1,042**
E-n101-k8	815	993	863	900	845	845
E-n101-k14	1,067	1,238	1,119	1,242	1,118	**1,117**
F-n45-k4	724	733	784	**758**	761	**725**
F-n72-k4	237	315	308	**267**	260	266
F-n135-k7	1,162	1,427	1,592	**1,398**	1,284	1,334
X-n101-k25	27,591	29,405	30,140	**29,783**	35,824	**29,137**
X-n106-k14	26,362	27,343	27,661	28,676	27,994	**27,304**
X-n110-k13	14,971	16,149	15,896	16,367	15,132	15,366
X-n115-k10	12,747	13,320	15,155	**14,635**	13,429	13,885
X-n120-k6	13,332	14,242	14,709	16,167	14,137	**13,805**
X-n125-k30	55,539	58,665	61,176	**59,603**	68,326	**58,317**
X-n129-k18	28,940	31,361	30,900	**30,560**	29,450	29,469
X-n134-k13	10,916	13,275	12,643	**12,108**	11,421	**11,303**
X-n139-k10	13,590	15,223	15,042	**14,873**	13,982	**13,972**
X-n143-k7	15,700	17,470	17,661	**17,450**	16,157	16,379
X-n148-k46	43,448	46,836	52,201	**47,545**	54,698	**45,456**
X-n153-k22	21,220	22,919	25,832	**24,449**	23,581	**23,317**
X-n157-k13	16,876	17,309	18,923	**18,374**	17,424	**17,371**
X-n162-k11	14,138	15,030	14,880	15,786	15,022	**14,814**
X-n167-k10	20,557	22,477	22,677	**22,478**	21,454	**21,113**
X-n172-k51	45,607	50,505	55,098	**50,044**	60,646	**48,850**
X-n176-k26	47,812	52,111	56,449	56,582	53,150	**52,012**
X-n181-k23	25,569	26,321	27,114	**26,951**	28,176	**26,208**
平均差距（%）	0	9.52	10.74	**9.88**	6.10	**3.32**

6 讨论

本节针对一些关键训练细节进行了多项消融研究：训练范式、任务选择策略、性能阈值、模型容量和训练样本规模。根据这些结果，我们全面介绍了 ASP框架，并为今后的神经求解器研究提供了一些训练方面的见解。对于所有消融研究，除重点考察的因素外，我们均保持其他设置不变，并使用 5 个随机生成的种子进行实验。

6.1 不同训练范式的比较

为验证 ASP框架的有效性，我们在均匀分布上使用混合问题规模 20、40、60、80 和 100 对 POMO 和注意力模型进行训练，并将其称为“均匀训练”。如图 2 所示，经过均匀训练的 POMO 能够取得与 ASP框架相当的性能。然而，经过均匀训练的注意力模型性能明显较差，甚至在旅行商问题和容量约束车辆路径问题上训练失败。这一观察表明

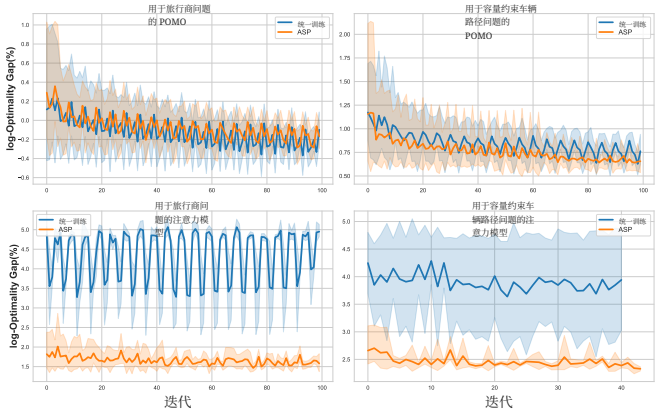

图 2：均匀训练与 ASP 的比较。均匀训练指同时训练所有问题规模，且每种问题规模的训练样本数量相同。采用均匀训练时，POMO 在旅行商问题上可取得与 ASP 相当的结果，在容量约束车辆路径问题上的性能则略差。然而，对于注意力模型，均匀训练在旅行商问题上的训练会失败，在容量约束车辆路径问题上的性能也较差；而 ASP 能够持续改进 POMO 和注意力模型在旅行商问题和容量约束车辆路径问题上的表现。

神经求解器并不能始终从混合不同类型问题实例的朴素训练流程中获益。然而，ASP框架展现出出色的迁移能力，因此适用于多种神经求解器和组合优化问题。

6.2 任务选择策略的影响

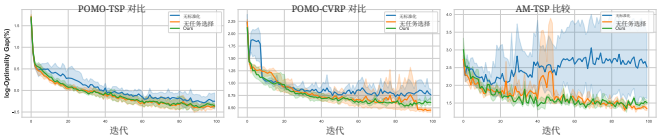

图 3：不同训练策略的比较。图例中的“无任务选择”“无标准化”和“我们的方法”分别表示训练时使用均匀策略、公式 12 中的 $\lambda = 1$ 和 $\lambda = 0.5$ 。在任务选择和无标准化策略下，训练有时会失败，例如，在容量约束车辆路径问题上训练 POMO，以及在旅行商问题上训练注意力模型。但在所有展示的案例中，ASP 均表现出训练稳定性。

我们在公式 12 中提出了任务选择策略：

$$p_i^t \propto \lambda p_i^1 + (1-\lambda)p_i^2, \lambda \in [0,1]$$

该策略从目标分布中获取信息，以引导训练过程。任务选择策略有两种简单的选择：**(1)** 均匀策略，即不偏向任何特定任务；以及**(2)** $\lambda = 1$，这是一种短视选择，强调当前性能。与这些候选策略相比，我们的方法同时考虑了改进需求和稳定性，显然更具优势。实验结果表明，这两种候选策略在某些情况下可能导致训练失败。如图 3 所示，我们在评估数据集上评估训练过程中的神经求解器：基础求解器-组合优化问题表示

我们在ASP框架下，针对特定组合优化问题，采用不同的任务选择策略训练基础求解器；“无标准化”、“无任务选择”和“我们的方法”分别表示 $\lambda = 1$、均匀策略和 $\lambda = 0.5$的情况。三个子图表明，这三种策略在某些情况下都可能表现良好，例如最左侧子图中的POMO-TSP。然而，在训练POMO-CVRP时，“无标准化”和“无任务选择”在不同随机种子下表现出不稳定性；在训练AM-TSP时甚至会导致训练失败，因为仅考虑改进的短视策略和均匀策略都可能误导训练。不过，在不同随机种子下，ASP对所有神经求解器和组合优化问题都表现稳定且效果良好。

6.3 性能阈值的影响

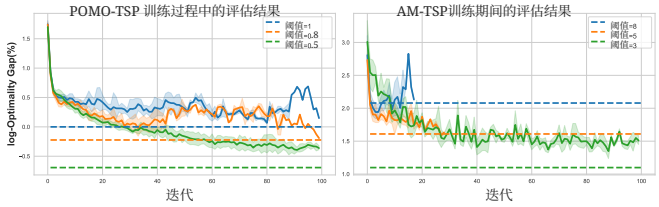

图 4：以对数刻度比较 POMO-TSP 和 AM-TSP 在不同阈值下的最优性差距（%）。较低的标准（此处即较大的阈值）会导致训练提前停止，例如，AM-TSP 在非常早的迭代中就达到了阈值 8。此外，合理设置阈值可以提升性能，例如，尽管最终未达到阈值 0.5，POMO-TSP 在阈值为 0.5 时仍能比阈值为 0.8 时取得更好的性能。

合适的阈值对性能大有裨益。期望过高会导致犹豫不决，而期望过低则会使人懈怠。图4展示了在不同性能阈值下，神经求解器在训练阶段的评估曲线。对于 POMO-TSP，我们选择了三个性能阈值：1、0.8、0.5，并保持其他设置不变；对于 AM-TSP，我们选择阈值=8、5、3进行比较。一方面，如果性能阈值过高，ASP框架的训练过程会提前停止，导致PSA过程（算法2）的迭代次数减少。例如，当我们将 AM-TSP 的阈值设为=8 时，训练过程在非常早期的迭代阶段就停止了。另一方面，如果阈值过于苛刻，神经求解器会受限于自身条件（例如模型容量、训练样本规模等），永远无法达到预期。例如，POMO-TSP 的阈值=0.5和 AM-TSP 的阈值=3都过于严苛，以至于它们在训练结束时仍未能实现目标。不过，苛刻的阈值仍能促使神经求解器获得更好的性能。例如，尽管 POMO-TSP 并未

虽然能够达到 0.5 的阈值，但在使用阈值=0.8进行训练时，仍可获得更好的性能。

设计阈值的一个明显指导依据是原始神经求解器的性能，而该性能在整个训练过程中始终固定。我们认为，动态调整阈值的过程将大有裨益

因为神经求解器在不同问题规模下的性能各不相同。

6.4 模型容量的影响

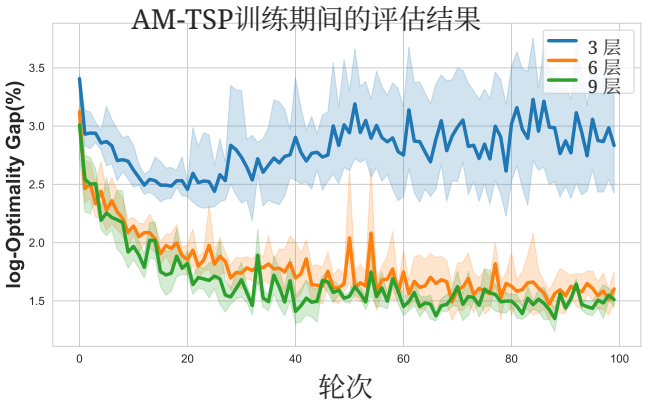

图5：不同编码器层数的 AM-TSP 在最优性差距（%）对数刻度下的评估结果。可以看出，模型容量较小会导致性能变差，甚至训练失败。

在整个训练过程中，我们需要处理一系列问题规模和更复杂的分布，这比只关注特定问题规模和均匀分布要困难得多。如图5所示，在处理更困难的任务时，模型容量是影响最终结果的重要因素。可以看出，具有3个编码器层的注意力模型训练失败，而具有6层和9层的注意力模型则能稳定收敛，这进一步验证了我们的假设。此外，具有9个编码器层的 AM-TSP 能够取得比具有6层的 AM-TSP 更好的结果，这意味着更大的容量会带来更好的性能。

6.5 训练样本规模的影响

训练样本规模也是影响最终性能的重要因素。我们在图6中展示了 POMO-TSP 和 AM-TSP 的消融结果，POMO-TSP 的性能阈值为0.5，AM-TSP 的性能阈值为3。在该图的图例中，“正常”表示我们训练默认设置下的样本规模 N，“小”和“极小”分别表示样本规模为 $\frac{N}{10}$ 和 $\frac{N}{100}$。相同的阈值可以作为标准，使比较更加清晰、公平。显然，较大的样本规模有助于提高收敛速度，并增加停止迭代次数。然而，较大的样本规模也意味着更大的计算负担。图6右侧的柱状图以对数小时尺度显示了不同样本规模下的平均训练时间消耗，因此在实际应用中需要权衡这一点。

6.6 ASP 对景观的影响

关于度量景观的形状是否与泛化能力有关，一直存在广泛讨论的观点 [54], [55], [56], [57]。一些研究人员认为，“尖锐的”



图6：不同样本规模的比较。正常、小和极小表示不同的样本规模。正常样本规模是小样本规模的10倍，而小样本规模是极小样本规模的10倍。左侧子图展示训练期间的评估结果，右侧子图展示不同随机种子的平均训练时间，单位为小时并采用对数刻度。

景观会导致泛化能力较差 [58], [59]，原因是训练时采用了大批量大小；但也有研究人员提出专门训练方法，以便在大批量大小下实现良好的泛化能力 [59]。在本部分中，我们通过可视化回报景观并讨论一些见解，深入探讨注意力模型和 POMO 的泛化能力。具体的可视化配置和更多可视化结果见附录F。

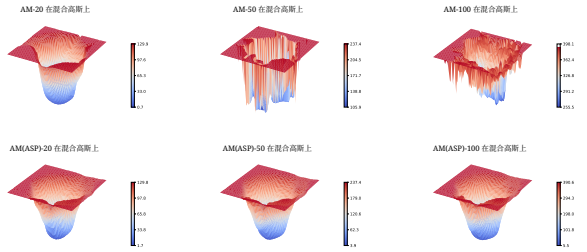

图7：注意力模型在混合高斯分布上测试时的最优性差距（%）景观图。

AM 的讨论。我们在图7中展示了 AM 和 AM(ASP) 在混合高斯分布上的最优性差距（%）景观。在对参数采用相同扰动设置的情况下，可以看到，随着问题规模增大，景观变得更加陡峭或尖锐。例如，AM(ASP) 在 TSP20、TSP50 和 TSP100 上的最优性差距（%）范围分别为 1.7% ～ 129.8%、3.9% ～ 237.4% 和5.5% ～ 390.6%，这表明问题规模越大，范围就越宽。此外，极小值越尖锐，就意味着越难达到 [60], ，因此我们需要更加关注优化水平或投入更多训练时间。结合已有认知——大规模组合优化问题比小规模组合优化问题更难——我们可以得出结论：大规模组合优化问题（更困难的任务）比小规模组合优化问题（更简单的任务）更难训练，这与常见做法一致：我们总是需要

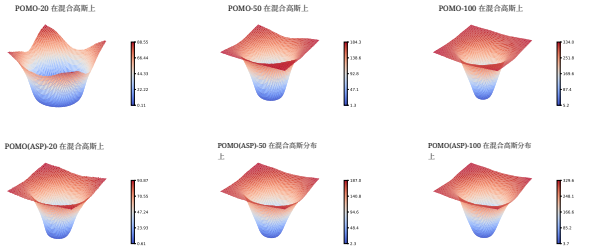

图8：POMO在混合高斯分布上测试时的最优性差距（%）景观图。

在大规模组合优化问题上投入更多训练时间，并且其性能比小规模组合优化问题更差。有趣的是，对优化水平的观察也符合课程训练中从简单任务开始的原则：课程训练所做的，就是从“宽极小值”开始，逐步转向“尖锐极小值”。

关于注意力模型和AM(ASP)之间的比较，我们可以看到，注意力模型在TSP50和TSP100上的景观图相当杂乱，而在TSP20上的景观图则相对平滑，这与表2和表6中的结果一致：注意力模型在TSP20上表现良好，但在TSP50和TSP100上表现不佳。然而，AM(ASP)在所有问题规模上得到的景观图都很平滑。

POMO 的讨论。我们在图8中可视化了混合高斯分布上 POMO 和 POMO(ASP) 的最优性差距（%）景观。POMO 在问题规模上的泛化能力与景观曲率之间的关系遵循相同规律：问题规模越大，最优性差距景观越陡峭。此外，与 AM 不同，POMO 在所有问题规模上都呈现出平滑的景观，与 POMO(ASP) 一样，这与表2和表6中的结果一致。根据景观的形状，我们可以说，在某种意义上，POMO 是比 AM 更强大的神经求解器。

7 结论

受 PSRO 和自适应阶梯框架的启发，本文提出了基于博弈论和课程学习的解决方案 ASP，该方案首次从多个方面提升了基于强化学习的神经求解器在组合优化问题上的泛化能力。我们在若干典型的组合优化问题上评估了该方法，并采用了不同的神经求解器。在随机生成的基准测试和真实世界基准测试上，我们表明，与一系列基线方法相比，在我们框架下训练的神经求解器展现出了令人印象深刻的泛化性能。因此，通过 ASP 训练的神经求解器可以避免过拟合，即使面对从未训练过的分布和问题规模，也能始终实现鲁棒性能，这是现有神经求解器从未实现过的。基于此，我们有充分理由相信，ASP 能够成为将神经求解器应用于工业应用的第一步。

参考文献

[1] E. Khalil、H. Dai、Y. Zhang、B. Dilkina 和 L. Song，《在图上学习组合优化算法》，神经信息处理系统进展，2017。

[2] W. Kool、H. van Hoof 和 M. Welling，《注意力，学会解决路径规划问题！》，载于第七届国际学习表征会议（*ICLR 2019*），美国路易斯安那州新奥尔良，*2019*年*5*月*6*日至*9*日，OpenReview.net，2019。[3] Y. Wu、W. Song、Z. Cao、J. Zhang 和 A. Lim，《用于解决路径规划问题的学习改进启发式算法》，*IEEE*神经网络与学习系统汇刊，2021。[4] W. Kool、H. van Hoof、J. A. S. Gromicho 和 M. Welling，《车辆路径问题的深度策略动态规划》，载于约束规划、人工智能与运筹学集成——第*19*届国际会议（*CPAIOR 2022*），美国加利福尼亚州洛杉矶，*2022*年*6*月*20*日至*23*日，论文集（P. Schaus 编），第13292卷，计算机科学讲义，第190–213页，施普林格，2022。[5] Y.-D. Kwon、J. Choo、B. Kim、I. Yoon、Y. Gwon 和 S. Min，《POMO：面向强化学习的多最优解策略优化》，神经信息处理系统进展，第33卷，第21188–21198页，2020。[6] C. Wang、Y. Yang、O. Slumbers、C. Han、T. Guo、H. Zhang 和 J. Wang，《一种提升旅行商问题求解器泛化能力的博弈论方法》，*arXiv*预印本 *arXiv:2110.15105*，2021。[7] Z. Zhang、Z. Zhang、X. Wang 和 W. Zhu，《通过难度自适应课程学习求解旅行商问题》，载于*AAAI*，2022。[8] M. Lisicki、A. Afkanpour 和 G. W. Taylor，《评估神经组合优化中的课程学习策略》，*arXiv*预印本 *arXiv:2011.06188*，2020。[9] B. Treutwein，《自适应心理物理程序》，视觉研究，第35卷，第17期，第2503–2522页，1995。[10] L. Dinh、J. Sohl-Dickstein 和 S. Bengio，《使用 real NVP 进行密度估计》，载于第五届国际学习表征会议（*ICLR 2017*），法国土伦，*2017*年*4*月*24*日至*26*日，会议论文集，OpenReview.net，2017。[11] G. Reinelt，《TSPLib——旅行商问题库》，*ORSA*计算期刊，第3卷，第4期，第376–384页，1991。[12] E. Uchoa、D. Pecin、A. Pessoa、M. Poggi、T. Vidal 和 A. Subramanian，《带容量约束的车辆路径问题的新基准实例》，*European Journal of Operational Research*，第257卷，第3期，第845–858页，2017。[13] O. Vinyals、M. Fortunato 和 N. Jaitly，《指针网络》，载于神经信息处理系统大会第*28*届：*2015*年神经信息处理系统年会，*2015*年*12*月7日，蒙特利尔，魁北克省，加拿大（C. Cortes、N. D. Lawrence、D. D. Lee、M. Sugiyama 和 R. Garnett 编），第2692–2700页，2015。[14] I. Bello、H. Pham、Q. V. Le、M. Norouzi 和 S. Bengio，《结合强化学习的神经组合优化》，载于第五届国际学习表征会议（*ICLR 2017*），法国土伦，*2017*年*4*月*24*日至*26*日，研讨会论文集，OpenReview.net，2017。[15] A. Vaswani、N. Shazeer、N. Parmar、J. Uszkoreit、L. Jones、A. N. Gomez、L. Kaiser 和 I. Polosukhin，《注意力机制就是你所需的一切》，载于神经信息处理系统大会第*30*届：*2017*年神经信息处理系统年会，*2017*年*12*月*4*日，长滩，美国加利福尼亚州（I. Guyon、U. von Luxburg、S. Bengio、H. M. Wallach、R. Fergus、S. V. N. Vishwanathan 和 R. Garnett 编），第5998–6008页，2017。[16] P. Veli˘ckovi´c、G. Cucurull、A. Casanova、A. Romero、P. Lio 和 Y. Bengio，《图注意力网络》，*arXiv*预印本 *arXiv:1710.10903*，2017。[17] H. Lu、X. Zhang 和 S. Yang，《一种用于求解车辆路径问题的基于学习的迭代方法》，载于第八届国际学习表征会议（*ICLR 2020*），埃塞俄比亚亚的斯亚贝巴，*2020*年*4*月*26*日至*30*日，OpenReview.net，2020。[18] Z. Fu、K. Qiu 和 H. Zha，《将小型预训练模型泛化至任意大型旅行商问题实例》，载于第三十五届*AAAI*人工智能会议（*AAAI 2021*）、第三十三届人工智能创新应用会议（*IAAI 2021*）、第十一届人工智能教育进展研讨会（*EAAI 2021*），线上会议，*2021*年*2*月*2*日至*9*日，第7474–7482页，AAAI出版社，2021。[19] F. Agostinelli、S. McAleer、A. Shmakov 和 P. Baldi，《利用深度强化学习和搜索求解魔方》，自然 · 机器智能，第1卷，第8期，第356–363页，2019。[20] F. Agostinelli、A. Shmakov、S. McAleer、R. Fox 和 P. Baldi，《无需扩展的 A*搜索：利用深度Q网络学习启发式函数》，*arXiv*预印本 *arXiv:2102.04518*，2021。[21] Y. Bengio、A. Lodi 和 A. Prouvost，“组合优化的机器学习：方法论概览”，European Journal of Operational Research，2020年。[22] K. Smith-Miles 和 S. Bowly，“通过在实例空间中演化生成新的测试实例”，Computers & Operations Research，第63卷，第102–113页，2015年。[23] G. Zuzic、D. Wang、A. Mehta 和 D. Sivakumar，“使用对抗训练为在线分配问题学习鲁棒算法”，arXiv预印本 arXiv:2010.08418，2020年。[24] S. Liu、K. Tang 和 X. Yao，“并行算法组合的生成对抗构造”，IEEE Transactions on Cybernetics，2020年。[25] I. Goodfellow、J. Pouget-Abadie、M. Mirza、B. Xu、D. Warde-Farley、S. Ozair、A. Courville 和 Y. Bengio，“生成对抗网络”，神经信息处理系统进展，第27卷，2014年。[26] M. Lanctot、V. F. Zambaldi、A. Gruslys、A. Lazaridou、K. Tuyls、J. P´erolat、D. Silver 和 T. Graepel，“多智能体强化学习的统一博弈论方法”，载于神经信息处理系统大会第30届：2017年神经信息处理系统大会年会，2017年12月4日至9日，美国加利福尼亚州长滩（I. Guyon、U. von Luxburg、S. Bengio、H. M. Wallach、R. Fergus、S. V. N. Vishwanathan 和 R. Garnett 编），第4190–4203页，2017年。[27] S. McAleer、K. Wang、M. Lanctot、J. Lanier、P. Baldi 和 R. Fox，“双人零和博弈中的随时最优PSRO”，arXiv预印本 arXiv:2201.07700，2022年。[28] N. Perez-Nieves、Y. Yang、O. Slumbers、D. H. Mguni、Y. Wen 和 J. Wang，“为开放式博弈中的学习建模行为多样性”，载于国际机器学习会议，第8514–8524页，机器学习研究论文集，2021年。[29] S. McAleer、J. Lanier、K. Wang、P. Baldi、R. Fox 和 T. Sandholm，“自博弈PSRO：迈向双人零和博弈中的最优策略群体”，arXiv预印本 arXiv:2207.06541，2022年。[30] J. Perolat、B. De Vylder、D. Hennes、E. Tarassov、F. Strub、V. de Boer、P. Muller、J. T. Connor、N. Burch、T. Anthony 等，“使用无模型多智能体强化学习掌握军棋”，Science，第378卷，第6623期，第990–996页，2022年。[31] H. Fu、W. Liu、S. Wu、Y. Wang、T. Yang、K. Li、J. Xing、B. Li、B. Ma、Q. Fu 等，“大规模不完美信息博弈中的演员-评论家策略优化”，载于国际学习表征会议，2021年。[32] S. McAleer、G. Farina、M. Lanctot 和 T. Sandholm，“Escher：通过计算历史价值函数估计遗憾，避免在博弈中使用重要性采样”，arXiv预印本 arXiv:2206 04122，2022年。。[33] Y. Bengio、J. Louradour、R. Collobert 和 J. Weston，“课程学习”，载于第26届国际机器学习年度会议论文集，第41–48页，2009年。[34] A. Pentina、V. Sharmanska 和 C. H. Lampert，“多任务的课程学习”，2015 IEEE计算机视觉与模式识别会议（CVPR），第5492–5500页，2015年。[35] S. Guo、W. Huang、H. Zhang、C. Zhuang、D. Dong、M. R. Scott 和 D. Huang，“CurriculumNet：从大规模网络图像中进行弱监督学习”，载于欧洲计算机视觉会议（ECCV）论文集，第135–150页，2018年。[36] Y. Tay、S. Wang、A. T. Luu、J. Fu、M. C. Phan、X. Yuan、J. Rao、S. C. Hui 和 A. Zhang，“用于长篇叙事阅读理解的简单有效的课程指针-生成器网络”，载于第57届计算语言学协会会议论文集，ACL 2019，意大利佛罗伦萨，2019年7月28日至8月2日，第1卷：长篇论文（A. Korhonen、D. R. Traum 和 L. M`arquez 编），第4922–4931页，计算语言学协会，2019年。[37] E. A. Platanios、O. Stretcu、G. Neubig、B. P´oczos 和 T. M. Mitchell，“神经机器翻译的基于能力的课程学习”，载于2019年计算语言学协会北美分会会议论文集：人类语言技术，NAACL-HLT 2019，美国明尼苏达州明尼阿波利斯，2019年6月2日至7日，第1卷（长篇与短篇论文）（J. Burstein、C. Doran 和 T. Solorio 编），第1162–1172页，计算语言学协会，2019年。[38] C. Florensa、D. Held、M. Wulfmeier、M. Zhang 和 P. Abbeel，“用于强化学习的反向课程生成”，载于机器人学习会议，第482–495页，机器学习研究论文集，2017年。[39] Z. Ren、D. Dong、H. Li 和 C. Chen，“深度强化学习中的带覆盖惩罚的自适应优先课程学习”

学习”，第29卷，第6期，第2216–2226页，2018年。[40] S. Narvekar、B. Peng、M. Leonetti、J. Sinapov、M. E. Taylor和P. Stone，“强化学习领域的课程学习：框架与综述”，机器学习研究期刊，第21卷，第181:1–181:50页，2020年。[41] R. Portelas、C. Colas、L. Weng、K. Hofmann和P.-Y. Oudeyer，“深度强化学习的自动课程学习：简要综述”，载于国际人工智能联合会议，2020年。[42] H. Y. Huang、D. Ye、L. Shen和W. Liu，“基于课程的非对称多任务强化学习”，arXiv，第abs/2211.03352卷，2022年。[43] P. Toth和D. Vigo，《车辆路径规划：问题、方法与应用》。工业与应用数学学会，2014年。[44] E. Balas，“奖金收集旅行商问题”，《Networks》，第19卷，第6期，第621–636页，1989年。[45] D. P. Kingma和J. Ba，“Adam：一种随机优化方法”，载于第3届国际学习表征会议，ICLR 2015，圣迭戈，加利福尼亚州，美国，2015年5月7日至9日，会议论文集（Y. Bengio和Y. LeCun编），2015年。[46] Gurobi Optimization, LLC，“Gurobi优化器参考手册”，2021年。[47] K. Helsgaun，“用于受约束旅行商问题和车辆路径问题的Lin-Kernighan-Helsgaun旅行商问题求解器扩展”，罗斯基勒：罗斯基勒大学，2017年。[48] L. Xin、W. Song、Z. Cao和J. Zhang，“采用嵌入式窥视机制的多解码器注意力模型求解车辆路径问题”，载于AAAI人工智能会议论文集，第35卷，第12042–12049页，2021年。[49] C. K. Joshi、T. Laurent和X. Bresson，“一种用于旅行商问题的高效图卷积网络技术”，arXiv预印本arXiv:1906.01227，2019年。[50] A. Hottung、B. Bhandari和K. Tierney，“使用变分自编码器为路径规划问题学习潜在搜索空间”，载于国际学习表征会议，2020年。[51] M. Nazari、A. Oroojlooy、L. V. Snyder和M. Tak´ac，“用于求解车辆路径问题的强化学习”，载于《神经信息处理系统进展》第31卷：神经信息处理系统会议年度会议2018，NeurIPS 2018，2018年12月3日至8日，加拿大蒙特´利尔（S. Bengio、H. M. Wallach、H. Larochelle、K. Grauman、N. Cesa-Bianchi和R. Garnett编），第9861–9871页，2018年。[52] X. Chen和Y. Tian，“学习执行组合优化的局部重写”，《神经信息处理系统进展》，第32卷，2019年。[53] L. Perron和V. Furnon，“OR-Tools”。[54] L. Dinh、R. Pascanu、S. Bengio和Y. Bengio，“尖锐极小值可以实现深度神经网络的泛化”，载于国际机器学习会议，第1019–1028页，机器学习研究论文集，2017年。[55] N. S. Keskar、D. Mudigere、J. Nocedal、M. Smelyanskiy和P. T. P. Tang，“关于深度学习的大批量训练：泛化差距与尖锐极小值”，载于第5届国际学习表征会议，ICLR 2017，法国土伦，2017年4月24日至26日，会议论文集，OpenReview.net，2017年。[56] N. S. Keskar、D. Mudigere、J. Nocedal、M. Smelyanskiy和P. T. P. Tang，“关于深度学习的大批量训练：泛化差距与尖锐极小值”，载于第5届国际学习表征会议，ICLR 2017，法国土伦，2017年4月24日至26日，会议论文集，OpenReview.net，2017年。[57] X. Chen和C.-J. Hsieh，“通过基于扰动的正则化稳定可微架构搜索”，载于国际机器学习会议，第1554–1565页，机器学习研究论文集，2020年。[58] P. Chaudhari、A. Choromanska、S. Soatto、Y. LeCun、C. Baldassi、C. Borgs、J. Chayes、L. Sagun和R. Zecchina，“熵随机梯度下降：将梯度下降偏向宽谷”，《统计力学杂志：理论与实验》，第2019卷，第12期，第124018页，2019年。[59] S. Hochreiter和J. Schmidhuber，“平坦极小值”，《Neural computation》，第9卷，第1期，第1–42页，1997年。[60] H. Li、Z. Xu、G. Taylor、C. Studer和T. Goldstein，“可视化神经网络的损失景观”，《神经信息处理系统进展》，第31卷，2018年。

致谢

本研究部分得到国家科技创新2030——“新一代人工智能”重大项目（编号：2022ZD0116408）的资助。

附录 A 预言机训练

在算法1中，我们需要训练两个预言机： $S^{'}$ 和$\mathbf{P}^{'}_{\mathcal{I}^n}$，作为要添加到相应策略集中的新策略。下面将具体推导训练该预言机所需的梯度。

根据公式（5），显然可以得到梯度：

$$\begin{aligned}\nabla_\theta L_{\text{SS}}(\theta) &= \nabla_\theta \mathbf{E}_{\mathbf{P}_{\mathcal{I}^n}\sim\sigma_{\text{DG}}}\mathbf{E}_{\mathcal{I}^n\sim\mathbf{P}_{\mathcal{I}^n}} g(S_\theta,\mathcal{I}^n,\text{Oracle})\\ &= \mathbf{E}_{\mathbf{P}_{\mathcal{I}^n}\sim\sigma_{\text{DG}}}\mathbf{E}_{\mathcal{I}^n\sim\mathbf{P}_{\mathcal{I}^n}} \nabla_\theta g(S_\theta,\mathcal{I}^n,\text{Oracle})\\ &= \mathbf{E}_{\mathbf{P}_{\mathcal{I}^n}\sim\sigma_{\text{DG}}}\mathbf{E}_{\mathcal{I}^n\sim\mathbf{P}_{\mathcal{I}^n}} \frac{\nabla_\theta S_\theta(\mathcal{I}^n)}{\text{Oracle}(\mathcal{I})}.\end{aligned} \tag{13}$$

同样，对于公式（8），该梯度的计算方式为：

$$\begin{aligned}\nabla_\gamma L_{\text{DG}}(\gamma) &= \mathbf{E}_{S\sim\sigma_{\text{SS}}}\nabla_\gamma \mathbf{E}_{\mathcal{I}^n\sim\mathbf{P}_{\mathcal{I}^n,\gamma}} g(S,\mathcal{I}^n,\text{Oracle})\\ &= \mathbf{E}_{S\sim\sigma_{\text{SS}}}\int_{\mathcal{I}} \nabla_\gamma \mathbf{P}_{\mathcal{I}^n,\gamma}(\mathcal{I}) g(S,\mathcal{I},\text{Oracle})\mathbf{d}\mathcal{I}\\ &= \mathbf{E}_{S\sim\sigma_{\text{SS}}}\int_{\mathcal{I}} \mathbf{P}_{\mathcal{I}^n,\gamma}(\mathcal{I})\frac{\nabla_\gamma \mathbf{P}_{\mathcal{I}^n,\gamma}(\mathcal{I})}{\mathbf{P}_{\mathcal{I}^n,\gamma}(\mathcal{I})} g(S,\mathcal{I},\text{Oracle})\mathbf{d}\mathcal{I}\\ &= \mathbf{E}_{S\sim\sigma_{\text{SS}}}\mathbf{E}_{\mathcal{I}\sim\mathbf{P}_{\mathcal{I},\gamma}}\nabla_\gamma \log \mathbf{P}_{\mathcal{I}^n,\gamma}(\mathcal{I}) g(S,\mathcal{I},\text{Oracle}).\end{aligned} \tag{14}$$

附录 B

训练细节

B.1 神经求解器的训练设置

基础求解器	编码器层	批量大小	样本规模	训练轮次
AM	9	512	512×2500	600
POMO	6	64	100×1000	600

表5：AM [2]和 POMO [5]在全部 COP 上的训练配置。

训练神经求解器时，ASP框架在每次迭代中运行 DE 流程（算法 1）或 PSA过程（算法 2）。我们运行 5 次 DE 流程迭代和 1 次 PSA过程迭代，每次迭代包含 5 个训练轮次。我们训练 100 次迭代，因此神经求解器的总训练轮次为 600。为进行公平比较，表2、表3和表4中的 POMO 结果均采用与 POMO(ASP) 相同的训练设置。对于 AM，我们采用原始训练设置（编码器层数为 3，训练轮次为 100）以及作者提供的预训练模型参数，因为我们发现，在与 AM(ASP) 相同的设置（9 个编码器层和 600 个训练轮次）下训练的 AM，性能相当不稳定。结果见附录C。部分关键参数的具体设置见表5。

B.2 数据生成器的训练设置

我们使用 Real-NVP [10] G_γ（一种典型的归一化流生成模型）来生成数据分布。不同的组合优化问题需要对数据生成过程进行具体规定。

对于旅行商问题- 在经典欧几里得旅行商问题中，我们只需生成二维坐标。具体来说，我们采样$x \sim \mathbf{U}([0,1]\times[0,1])$ 并得到数据 $y = G_\gamma(x) \sim P_{g,\gamma_0}$

对于车辆路径问题- 与旅行商问题相比，容量约束车辆路径问题和拆分配送车辆路径问题除了二维坐标外，还包含对客户需求和车辆容量的额外约束。在将神经求解器用于求解容量约束车辆路径问题和拆分配送车辆路径问题的一般设置中，通常会以相对稳妥的方式设定这些约束。为避免因这两项约束而产生不可行或质量较差的实例，我们沿用以往工作的做法，生成客户需求和车辆 [2]，并重点关注欧几里得坐标的影响。与旅行商问题相同，我们从二维单位正方形中采样客户和仓库坐标 x ，并得到数据 $y = G_\gamma(x) \sim P_{g,\gamma}$。**对于 PCTSP 和 SPCTSP-** 我们沿用注意力模型中的相同设置 [2] 来生成奖励和惩罚，并与旅行商问题、车辆路径问题和定向问题一样，重点关注二维坐标。

附录C 注意力模型的进一步结果

表6：训练600轮次后，采用9个编码器层的注意力模型的进一步结果。间隙百分比是相对于全部方法中的最佳值计算的。

		$n = 20$	$n = 40$	$n = 60$	$n = 80$	$n = 100$
TSP	AM-20	0.73%	2。09%	8.02%	12。33%	18.09%
	AM-50	83.35%	115.83%	150.54%	198.55%	235.07%
	AM-100	117。38%	165。39%	200。92%	281。61%	312。47%
CVRP	AM-20	3。64%	10。27%	19。01%	33。31%	31。38%
	AM-50	223.24%	277.00%	451.86%	407.61%	414.29%
	AM-100	223.24%	277.00%	451.86%	407.61%	414.29%
SDVRP	AM-20	3。35%	10.85%	17.20%	27.77%	28.38%
	AM-50	223.24%	277.00%	451.86%	407.61%	414.29%
	AM-100	148.69%	211.00%	254.66%	267.96%	297.00%
PCTSP	AM-20	0.26%	22.07%	53.28%	66.86%	81.58%
	AM-50	38.12%	15.11%	13.76%	7.14%	11.77%
	AM-100	148.69%	211.00%	254.66%	267.96%	297.00%
SPCTSP	AM-20	1.29%	21。77%	46。33%	63。09%	68.25%
	AM-50	38.12%	15。11%	13.76%	7。14%	11。77%
	AM-100	148.69%	211.00%	254.66%	267.96%	297.00%

表6展示了由 9 个编码器层训练、并在均匀分布上训练 600 轮次的 AM [2] 的结果。AM-N 表示在问题规模为 N 的情况下训练的 AM；我们在问题规模为 20、40、60、80 和 100 的生成实例上评估 AM-N。结果表明，AM-50 和 AM-100 表现不佳，这是由训练轮次过多导致的过拟合造成的。然而，有趣的是，AM-20 似乎较少受到这个问题的影响。

附录D 泛化演示

我们在图 9 中展示了注意力模型 [2]和 POMO[5] 在不同问题规模（20、30、40、50、60、70、80、90 和 100）下的泛化结果。在不同问题规模下，随着差异增大，基础求解器的性能会下降；然而，基础求解器（ASP）能够很好地覆盖全部问题规模。

附录E 旅行商问题弱点分布可视化

本部分展示数据生成器为旅行商问题找到的弱点分布。根据算法 1，弱点分布为

$$\mathbf{P^n} = \sum_i \sigma_{\mathrm{DG},i} \mathbf{P}_{\mathcal{I}^n,i}$$

其中$\mathbf{P}_{\mathcal{I}^n,i}$ 是数据生成器在每次 PSRO 迭代中训练得到的 Oracle，$\sigma_{\mathrm{DG},i}$ 是元策略。得益于归一化流的使用，我们能够精确计算概率。概率密度函数（pdf）是在 $[0,1] \times [0,1]$ 的范围内获得的。在图 10 中，我们可以看到，AM 和 POMO 的弱点分布相较于均匀分布只有极其微小的变化，这可以视为一种特定类型的数据增强。

附录F 景观的进一步展示

我们使用“滤波器归一化” [60] ，在图 11 和图 12 中可视化 AM 和 POMO 的 COP 目标函数曲率。对于每组置换参数，我们评估 1000 个实例；置换范围为 -1 到 1，步长为 50。我们结合以下设置对 AM 和 POMO 进行评估：

- 不同的分布：“Uni.”和“Mix Gau.”表示我们绘制了在从均匀分布和混合高斯分布中采样的实例上评估得到的目标函数景观
- 不同的训练范式：“神经求解器（ASP）”表示结果来自在 ASP 框架下训练的神经求解器，否则则是在原始设置下训练的求解器所得
- 不同的问题规模：我们在不同的问题规模下评估神经求解器：20、50 和 100
- 不同的困难程度：我们根据最优性差距将 1000 个实例分为三个困难程度：简单（最优性差距最小的前$\frac{1}{3}$个实例）、中等（$\frac{1}{3} \sim \frac{2}{3}$）和困难（$\frac{2}{3} \sim 1$）。

例如，“AM(ASP)-20 on Mix Gau. (all)”表示：该景观是通过评估在 ASP框架上针对 TSP20 训练的注意力模型得到的，且这 1000 个实例是从混合高斯分布中采样的。

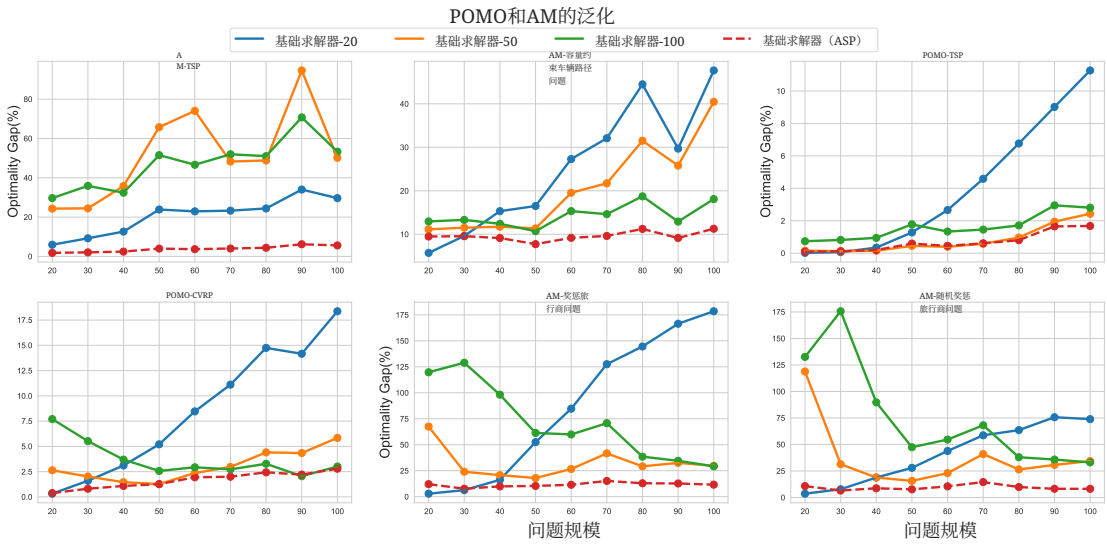

图 9：泛化能力演示。注意力模型和 POMO 的最优性差距（%）随问题规模变化的情况 $n \in \{20, 30, 40, 50, 60, 70, 80, 90, 100\}$。基础求解器（ASP）的结果以虚线绘制，基础求解器的结果则以实线绘制。直观来看，虚线（基础求解器（ASP）的结果）大致位于所有实线（基础求解器的结果）之下。

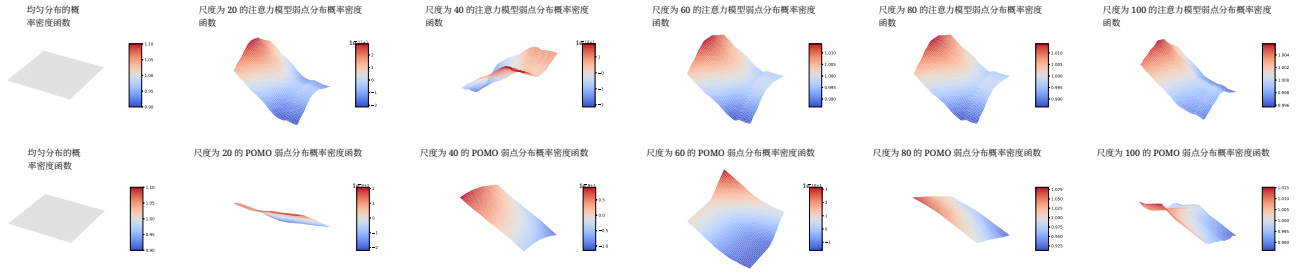

图 10：注意力模型和 POMO 的弱点分布演示。概率密度函数（概率密度函数）取值范围为 $[0, 1] \times [0, 1]$。



图 11：注意力模型的所有最优性差距（%）景观图。图 12：POMO 的所有最优性差距（%）景观图。