Advances in Applied Mathematics 应用数学进展, 2025, 14(4), 916-926
Published Online April 2025 in Hans. https://www.hanspub.org/journal/aam
https://doi.org/10.12677/aam.2025.144216

Hans汉斯

DSP-Transformer：基于注意力机制的最小支配集问题求解方法

刘斯豪，何伟骅*

广东工业大学数学与统计学院，广东 广州

收稿日期：2025年3月28日；录用日期：2025年4月23日；发布日期：2025年4月29日

摘　要

最小支配集问题在计算机科学与网络优化领域具有广泛应用，但其NP-完全性使得传统启发式方法计算成本高且难以泛化。本文提出了DSP-Transformer，一种基于Transformer结构的深度强化学习求解方法，通过编码器－解码器架构与近似策略优化方法，高效求解最小支配集问题。由于注意力机制更适合捕获图上的潜在关系，相较于传统基于消息传递机制的编码器，本方法在性能上也有显著提升。实验结果表明，该方法在T1基准数据集和ER随机图合成数据集上的表现优于传统启发式算法，能够提供更优质的解，同时显著减少计算成本。DSP-Transformer预训练后可泛化到更大规模的图实例，无需每次重新搜索，展现出深度学习在求解组合优化问题中的巨大潜力。

关键词

组合优化，最小支配集，Transformer，注意力机制

DSP-Transformer: An Approach for Solving the Minimum Dominating Set Problem Based on Attention Mechanism

Sihao Liu, Weihua He*

School of Mathematics and Statistics, Guangdong University of Technology, Guangzhou Guangdong

Received: Mar. 28th, 2025; accepted: Apr. 23rd, 2025; published: Apr. 29th, 2025

Abstract

The Minimum Dominating Set Problem has widespread applications in computer science and

*通讯作者。

文章引用: 刘斯豪, 何伟骅. DSP-Transformer: 基于注意力机制的最小支配集问题求解方法[J]. 应用数学进展, 2025, 14(4): 916-926. DOI: 10.12677/aam.2025.144216

network optimization. However, its NP-completeness makes traditional heuristic methods computationally expensive and difficult to generalize. In this paper, we propose DSP-Transformer, a deep reinforcement learning approach based on the Transformer architecture, which efficiently solves DSP through an encoder-decoder framework and approximate policy optimization. Since the attention mechanism is better suited for capturing latent relationships in graphs, our approach achieves significant performance improvements over traditional message-passing-based encoders. Experimental results demonstrate that DSP-Transformer outperforms conventional heuristic algorithms on both the T1 benchmark dataset and ER random graphs, providing higher-quality solutions while significantly reducing computational costs. Once pretrained, DSP-Transformer can generalize to larger graph instances without requiring repeated searches, highlighting the potential of deep learning in tackling combinatorial optimization problems.

Keywords

Combinatorial Optimization, Minimum Dominating Set, Transformer, Attention Mechanism

1. 引言

组合优化问题在计算机科学与工业应用领域具有广泛的重要性，其应用场景涵盖操作系统，任务调度，社交网络等多个方面。在电力系统与无线网络领域中，支配集问题(Dominating Set Problem, DSP)是组合优化的典型范例。具体而言，对于给定图 G，其支配集定义为顶点集的子集 D，满足图 G 中任一顶点要么属于 D，要么与 D 中至少一个顶点邻接。该问题的核心目标在于寻找具有最小基数的支配集。作为组合优化的重要分支，支配集问题及其变体在现实场景中展现出丰富的应用价值[1]。

然而，寻找支配集问题的最优解属于 NP 完全问题，Haynes 等人[2]已对此给出了标准证明。因此，过去十年间学界提出了大量算法以求解该问题。事实上，现有算法多依赖人工设计的启发式策略来高效获取解。但此类方法的显著缺陷在于：当实际问题的数据发生微小变化时，启发式算法需反复重新搜索，导致计算资源的极大浪费。

基于此，我们期望构建能够自动识别并理解支配集问题核心特征的算法。众所周知，深度学习擅长从欧氏空间数据(如图像、文本、视频)中提取隐含特征。然而，针对图结构数据的学习对传统深度学习算法提出了诸多挑战。图神经网络(Graph Neural Networks, GNNs) [3]为深度学习向图域迁移提供了有效途径。近期大量研究表明，GNNs 能够从不同规模、不同类型的图中捕获隐含的复杂特征。因此，我们认为 GNNs 是实现图表示学习与特征自动提取的理想工具。

引入深度学习后，模型训练成为另一关键挑战。对于支配集问题，现有数据集难以覆盖所有规模及其变体的问题实例。受 AlphaGo [4]启发，我们发现强化学习(Reinforcement Learning, RL) [5]可使算法在无监督条件下自主学习决策机制。强化学习的核心思想是通过训练智能体与环境持续交互，使其最大化累积回报。由于强化学习直接利用智能体与环境交互产生的经验样本，该范式在无标注数据的场景中具有显著优势。这为缺乏标注数据集的 NP 难组合优化问题提供了新的解决思路。本文重点研究基于深度学习方法求解支配集问题的解决方案。

2. 相关工作

2.1. 支配集的相关研究

关于支配集问题的研究可以追溯到 1962 年，O. Ore 首次给出了支配集问题的定义[6]。Haynes 等人[2]和 Bondy [7]都证明了求解 DSP 的精确解是 NP 完全的。在过去二十年里，许多研究致力于为 DSP 寻找更紧的界限。最初，研究者们尝试探讨不同图类的界限[8] [9]。特别是，Alanko 等人[10]展示了对大小为 29 × 29 的方格图的结果。然而，计算 29 × 29 方格图的结果耗费了 31 个 CPU-Day。

2.2. 基于学习的组合优化求解方案

在过去几十年中，组合优化问题一直是优化领域的研究热点。由于大多数组合优化问题可以形式化为整数规划(或混合整数规划)问题，因此该领域中陆续开发了如 Gurobi、CPLEX [11]等数值求解器，用以求解这些问题的近似解。然而，对于大规模的 NP 难组合优化问题，这些求解器的运行效率明显无法满足实际需求。因此，近年来研究者开始探索基于机器学习或深度学习的组合优化求解方法，这些方法充分利用深度学习捕捉问题隐藏特征的优势，旨在从大量问题实例中提取关键特征，从而显著提升大规模组合优化问题的求解效率。

早在 1985 年，Hopfield 及其合作者 Tank [12]就首次提出了利用 Hopfield 网络求解组合优化问题，最早将其应用于旅行商问题，这为后续神经网络在组合优化中的应用奠定了基础。随后，Bello 等人[13]于 2016 年提出了“神经组合优化”方法，通过结合深度学习和强化学习，开创性地实现了端到端求解组合优化问题的新范式。2015 年，Vinyals 等人[14]提出了指针网络，这是一种采用编码器–解码器结构的模型，利用注意力机制解决了训练图与测试图大小不一导致输出序列长度不一致的问题，并在求解平面凸包、Delaunay 三角剖分等问题上取得了良好效果。因此，我们在设计模型时也采用了编码器–解码器的基本架构。

随着图神经网络(GNNs)的快速发展，GNNs 被引入到编码器模块中，以更高效地提取图的结构信息和特征。Dai 等人率先将 GNN 应用于组合优化问题，他们通过图结构过滤当前状态，并基于得到的特征向量计算可选动作的值，再利用贪心策略选出最佳动作。在我们的模型中，同样利用了 GNN 来捕捉图的拓扑信息，提升决策效果。此外，Deudon 等人[15]在沿用了 Bello 等人方法的基础上，对编码器–解码器架构进行了改进，其解码器依旧采用指针网络，但其 GNN 编码器完全基于注意力机制，将输入视作集合而非序列，以增强模型对启发式信息的泛化能力。针对直接训练求解器存在的局限，Chen 等人[16]进一步提出了 NeuRewriter 网络，该方法不直接训练求解器，而是通过学习如何选择启发式策略，对预训练结果进行局部重写，从而不断提升解的质量。

为了解决支配集问题，Chen 等人[17]提出了一种基于双重深度 Q 网络(DDQN)的神经网络训练方法，并且使用信息传递网络(Message Passing Network)来进行图表示，以便更好地捕捉图的结构和特征，所提出的模型在求解支配集问题时表现出良好的效果。

总体来说，这些工作从不同角度探索了神经网络在组合优化问题中的应用路径，从最初的 Hopfield 网络到指针网络，再到图神经网络和启发式重写方法，为我们设计更高效的组合优化模型提供了丰富的理论基础和实践经验。

3. DSP-Transformer

根据支配集问题的形式，我们构建一个用于求解图上的支配集问题的 Transformer 模型，模型框架如图 1 所示。对于同样的顶点子集的选择问题，也可以进行类似的构建，本文以支配集问题为例。一个问

题实例 s 被定义为一个有 n 个顶点的图 G，每个顶点被标号为 v_i，其中 $i \in \{1,2,\cdots,n\}$ 并且对应顶点的特征为 x_i，其中 $x_i \in \mathbb{R}^k$。实际上，x_i 是顶点 v_i 的顶点嵌入(Embedding)，也就是顶点的一个低维向量表示。我们定义支配集的解 $\boldsymbol{\pi} \subseteq V$，进一步基于 Transformer 的编码器－解码器模型定义了一个随机策略 $p(\boldsymbol{\pi} \mid s)$，用于在给定问题实例 s 时选择解 π。该策略通过参数 θ 进行分解和参数化，其表达式为：

$$p_\theta(\boldsymbol{\pi} \mid s) = \prod_{t=1}^{n} p_\theta(\pi_t \mid s, \boldsymbol{\pi}_{1:t-1}).$$

模型中的编码器会产生所有输入顶点的嵌入，而解码器根据编码器得到的顶点嵌入做出对应的决策最终得到这个实例的解。解码器每次会选择一个顶点，并根据策略 p_θ 得到的分布，采样得到最后这个顶点是否被加入到解集 π 中。

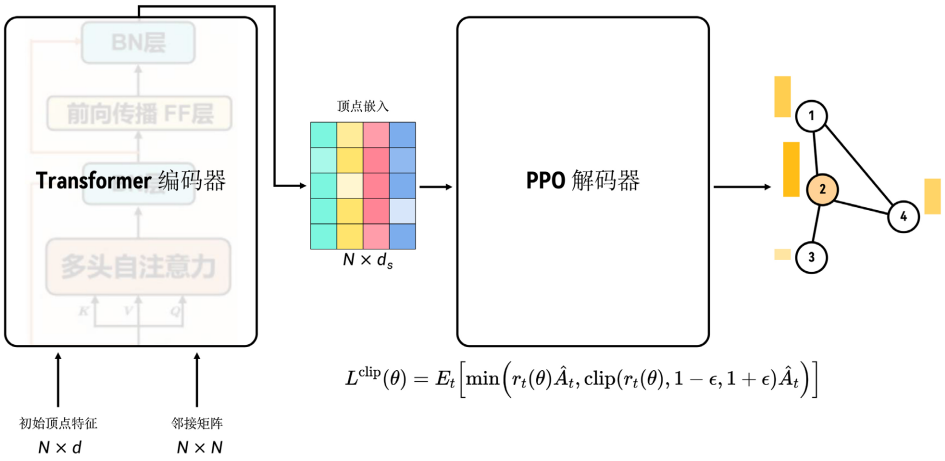

Figure 1. The framework of DSP-Transformer
图 1. DSP-Transformer 的模型框架

3.1. 编码器

受到[18] [19]的启发，我们选择了和 Transformer 相似的编码器架构。相较于传统的消息传递模式的图神经网络编码器(如：GCN、GAT 等)，Transformer 不仅可以捕获更长距离的依赖信息，而且得益于多头注意力机制，使其能够在不同的子空间上同时捕捉多种关系，这种多视角的建模方式在处理结构复杂的 DSP 数据时，能更全面地捕捉多种信号特征和关系模式。此外，Transformer 架构由于其并行计算的优势，在处理大规模图数据时更具可扩展性。本模型中每层注意力网络由两个子层构成：

多头注意力层(Multi-Head Attention)对于第 ℓ 层，给定输入序列 $\left\{\boldsymbol{h}_1^{(\ell-1)},\cdots,\boldsymbol{h}_n^{(\ell-1)}\right\} \in \mathbb{R}^{d_h}$，首先计算其每个注意力头的注意力向量

$$\text{head}_m = \text{Attention}\left(\boldsymbol{Q}W_m^Q, \boldsymbol{K}W_m^K, \boldsymbol{V}W_m^V, \boldsymbol{M}\right)$$

$$\text{Attention}(Q,K,V,M) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}} + \boldsymbol{M}\right)V$$

其中 $W_m^Q \in \mathbb{R}^{d_h \times d_k}$ 是第 m 个头的查询投影矩阵，$W_m^K \in \mathbb{R}^{d_k}$ 是第 m 个头的键投影矩阵，$W_m^V \in \mathbb{R}^{d_h \times d_v}$ 是第 m 个头的值投影矩阵，A 是图的邻接矩阵，$\boldsymbol{M}$ 是掩码矩阵，该矩阵的元素可以表示为：

$$M_{ij} = \begin{cases} 0, & \text{若 } A_{ij} = 1, \\ -\infty, & \text{若 } A_{ij} = 0. \end{cases}$$

接着拼接所有注意力头的注意力向量得到全局的注意力向量

$$\mathrm{MHA}^{\ell}(\cdot)=\mathrm{Concat}\left(\mathrm{head}_1,\cdots,\mathrm{head}_M\right)W^{O},$$

其中 $W^{O}\in\mathbb{R}^{Md_v\times d_h}$ 为输出投影矩阵，M 是注意力头数。

全连接层对于第 ℓ 层，给定输入 $\hat{\boldsymbol{h}}_i\in\mathbb{R}^{d_h}$，其计算为：

$$\mathrm{FF}^{\ell}\left(\hat{\boldsymbol{h}}_i\right)=W_2^{\ell}\cdot\mathrm{ReLU}\left(W_1^{\ell}\hat{\boldsymbol{h}}_i+b_1^{\ell}\right)+b_2^{\ell},$$

其中 W_j^{ℓ} 是第 ℓ 层的全连接层的权重参数，b_j^{ℓ} 是对应层的偏置项。

由于注意力机制的特性，每一层中的计算都涉及到图中的所有节点，但其操作是节点级别的。这意味着无论输入图有多少个节点，编码器都可以进行相同的操作流程，只是操作的顶点数量会相应地随着图规模的变化而动态调整。为了降低神经网络在训练过程中的波动并提高模型的泛化能力，在每一层全连接层的基础上使用了一些训练技巧。为了更好平衡深度网络带来的性能退化，我们加入了跳跃连接机制，避免深层网络中的梯度消失问题；进一步对每一个批量使用归一化，减小了内部协变量偏移的问题，进而在加速训练速度的同时，提高了模型的泛化性。综上，

$$\hat{\boldsymbol{h}}_i=\mathrm{BN}^{\ell}\left(\boldsymbol{h}_i^{(\ell-1)}+\mathrm{MHA}_i^{\ell}\left(\boldsymbol{h}_1^{(\ell-1)},\cdots,\boldsymbol{h}_n^{(\ell-1)}\right)\right),$$

$$\boldsymbol{h}_i^{(\ell)}=\mathrm{BN}^{\ell}\left(\hat{\boldsymbol{h}}_i+\mathrm{FF}^{\ell}\left(\hat{\boldsymbol{h}}_i\right)\right).$$

3.2. 解码器

解码器根据编码器输出的顶点嵌入来决策哪些顶点将被标记为支配顶点，加入到支配集 π_s 中，若将决策算法视为一个**智能体**，要求他根据实例 S 的状态 x_S 来确定选择哪些支配顶点这个动作 $\mathcal{A}_s$，那么这个系统可以利用强化学习来进行优化，从而找到最优的决策策略(解码器) p^{*}，在每个不同的实例 S 下，都能找到最优的解 π_S^{*}。

首先，定义图 $G=(V,E)$ 作为实例 S。对于每个图 G，智能体根据每个顶点的嵌入信息来选择哪些顶点应当被标记为支配顶点，从而构成支配集 π_s。解码器在此过程中的角色是根据图 G 的顶点嵌入信息决定选择哪些顶点加入到支配集 π_s。中，因此可以将支配集问题转化为一个马尔可夫决策过程。

- **状态空间：**状态空间可以表示为图 $G=(V,E)$ 中的顶点的选择情况。每个状态 x_S 是一个长度为 $|V|$ 的二进制向量，其中每个元素表示图中相应顶点的状态：$x_S[v]=1$ 表示顶点 v 已加入支配集 π_s；反之，$x_S[v]=0$ 表示顶点 v 尚未加入支配集。状态空间的定义为

$$\mathcal{S}=\left\{x_S\in\{0,1\}^{|V|}\mid x_S[v]\in\{0,1\},\forall v\in V\right\}.$$

- **动作空间：**智能体的动作 $\mathcal{A}_s$ 表示对图中顶点的选择，即解码器决定将哪些顶点加入到支配集 π_s 中。
- **转移函数：**转移函数描述了智能体在状态 x_S 下选择动作 a_v 选择顶点 v 加入支配集)后，图的状态如何转移到新状态 $x_{S'}$。当智能体选择顶点 v 加入支配集时，状态 x_S 会更新为新的状态 $x_{S'}$，即顶点 v 被添加到支配集 π_s 中，且与 v 相邻的顶点也被“支配”。转移函数 $P\left(x_{S'}\mid x_S,a_v\right)$ 定义为：

$$P\left(x_{S'}\mid x_S,a_v\right)=\begin{cases}1 & \text{if?}x_{S'}=x_S\cup\{v\},\forall u\in N(v),x_{S'}[u]=1\\ 0 & \text{otherwise}\end{cases}.$$

- **奖励函数：**奖励函数衡量了当前选择的动作对解决支配集问题的贡献。具体的奖励函数可以表示为：

$$R\left(x_S,\mathcal{A}_S\right)=\frac{\sum_{v\in\pi_S}w_v}{\sum_{v\in V\setminus\pi_S}w_v},$$

其中，奖励函数的分子是支配顶点集 π_s 中所有顶点的权重和，分母是被支配顶点集中的所有顶点的权重和。若实例为无权图，则默认每个顶点权重都为 1，换言之，分子是支配顶点集的大小，分母是被支配顶点集的大小。

- **策略：**策略 $\pi_\theta(x_S)$ 通过一个神经网络表示，输出每个可能动作的概率。目标是通过训练使策略能够在每个实例中选择出最优的支配集 π_S^*。

对于这个 MDP 问题，我们的目标是优化策略 π_θ，使得智能体能够在每个图实例中选择最优的支配集 π_S^*，最大化累积奖励，从而得到最小的支配集，即：

$$\max_{\pi_\theta} \mathbb{E}\left[\sum_{t=0}^{T} \gamma^t R\left(x_{S_t}, a_{S_t}\right)\right]$$

其中，T 是训练过程的总步数，γ 是折扣因子，$R\left(x_{S_t}, a_{S_t}\right)$ 是时间步 t 的奖励。

3.3. 近似策略优化算法

根据上面的 MDP 定义，使用近似策略优化算法(Proximal Policy Optimization, PPO) [20]训练解码器的策略，使得其能够选择出最优的支配集。

1. 初始化策略网络：首先，初始化策略网络 $\pi_\theta(x_S)$，通常我们使用一个神经网络来近似策略。网络输入是图的状态 x_S，输出是每个顶点选择的概率分布。策略网络的目标是学习一个映射：

$$\pi_\theta : \mathcal{S} \to \mathcal{A}, \pi_\theta(a_v \mid x_S) = P(a_v = 1 \mid x_S)$$

其中 $P(a_v = 1 \mid x_S)$ 表示在状态 x_S 下选择顶点 v 加入支配集 π_S 的概率。

2. 采样轨迹：在训练过程中，智能体与环境交互，按照当前的策略 π_θ 采样多个轨迹。每个轨迹由一系列的状态 $\{x_0, x_1, \cdots, x_T\}$、动作 $\{a_0, a_1, \cdots, a_T\}$ 和奖励 $\{R_0, R_1, \cdots, R_T\}$ 组成。轨迹采样过程的目标是收集尽可能多的经验，以便在后续步骤中利用这些数据来估计优势函数和优化策略。

3. 估计优势函数：在强化学习中，优势函数 $\hat{A}_t$ 用来衡量在给定状态 x_t 下选择某个动作 a_t 相比于基准策略能够带来的额外收益。优势函数的主要目的是评估一个动作相较于其他可能的动作在当前状态下的好坏程度，帮助模型判断是否该采取该动作。由于优势函数依赖未来的状态和奖励，所以我们需要通过采样估计的形式来估计优势函数 A_t。为了进一步减少方差并提升训练稳定性，我们使用广义优势估计(GAE)来计算优势函数。GAE 结合了多步时间差分误差，使用一个衰减因子 λ 来加权计算：

$$A_t = \sum_{l=0}^{\infty} (\gamma\lambda)^l \delta_{t+l}$$

其中 δ_{t+l} 是每步的时间差分误差，定义为：

$$\delta_t = R_t + \mu V(x_{t+1}) - V(x_t),$$

其中，R_t 是时间步 t 的即时奖励，μ 是折扣因子，控制未来奖励的影响程度，$V(x_t)$ 是状态 x_t 的值函数，表示从状态 x_t 开始的期望回报。GAE 通过控制衰减因子 μ 来平衡方差和偏差。

4. 更新策略

本文通过 PPO 最原始的裁剪目标函数来更新策略网络的参数，PPO 的目标函数定义为：

$$L^{\text{clip}}(\theta) = E_t\left[\min\left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t\right)\right],$$

其中 $r_t(\theta) = \dfrac{\pi_\theta(a_t \mid x_t)}{\pi_\theta^{\text{old}}(a_t \mid x_t)}$ 是当前策略和旧策略的比率，ϵ 是裁剪因子，用来限制策略更新的步幅。通过最小化 $L^{\text{clip}}(\theta)$，PPO 能够稳定地优化策略，避免过大的策略更新步伐，保证训练过程中的稳定性。在每次迭

代中，智能体持续与环境交互，采样新的轨迹并计算新的优势函数。然后，我们使用上述的 PPO 优化目标函数来更新策略网络的参数。重复这个过程，直到训练误差小于设定值则视为训练收敛。

4. 实验结果与分析

在本节中，我们进行了一系列实验来评估 DSP-Transformer 的性能，实验分为两个主要部分：在合成数据集上的表现以及标准数据集测试。首先，我们使用合成数据集进行模型的训练，并在另外一部分模型没见过的数据上进行测试，并将其与其他主流的算法和模型进行比较。接下来，我们使用了控制集问题的一个标准测试数据集：T1，在 T1 上直接使用我们上一步预训练好的模型进行推理求解，并将其结果与性能和其他模型进行对比。

4.1. 数据生成

- **合成数据集：**为了训练和评估所提议的嵌入方法和强化学习模型，我们为支配集问题(DSP)生成了多个实例。在本文中，我们生成了 Erdös-Renyi (ER)图[21]和网格图，这些图在许多模型评估中得到了广泛应用[22]。如果所提模型能够在这些图上表现良好，则意味着该模型是有效的。
- **T1 基准数据集：**T1 是一个著名的数据集，包含 520 个图实例，每个实例都有对应的支配集问题(DSP)的最优解。T1 中的每个实例均来源于文献[23]。与之不同的是，在本文中，我们特别选择了原始图，其中每个顶点的权重均设置为 1。考虑到顶点和边的大小不同，T1 可以被划分为 52 个图族，每个图族包含 10 个相同大小的实例。在这些图族中，我们从不同顶点数量的实例中选择了 7 个实例来评估我们的模型。同时，我们还进行了各种进化算法与贪心算法的性能对比。我们将所有模型应用于选定的 T1 数据集，每种图大小下运行 10 个实例，然后计算支配数的平均值和运行时间。

4.2. 实验设定

1. 训练部分：本文中，我们在 ER 图上训练所提出的模型，其中 n 表示图中顶点的数量，p 是连接概率，且 p 服从 0 到 1 之间的正态分布。我们分别设置 $n=20,40,80,100$ 。在训练过程中，我们使用另一组随机图来测试模型，并通过获取测试得分来评估模型的效果。

2. 测试部分：为了测试训练后的模型，首先，我们在 ER 图 $G(n,p)$ 上对模型进行测试，其中 p 取值为 0.1，0.5，0.8，$n=20,40,100,200,300,400,500,800$ 来测试模型的泛化能力，并通过与贪心算法进行比较来评估模型的表现。如果模型的解优于贪心算法，甚至花费的时间更少，则认为模型有效。其次我们在已知最优解的 T1 基准数据集上评估模型。最后，我们还在现实网络上进行了模型测试。

4.3. 实验结果

我们首先生成了不同规模的 ER 图数据集，每个规模($n=20,40,60,80,100$)各 300 个，并在这些不同规模的图上分别训练了一系列基础模型。接着，我们利用 GUROBI 求解器对这些图求解最小支配集，以获得近似最优解，并另外使用 100 张各规模的图进行测试，每张图重复跑 100 次，计算其平均值和标准差。如图 2 所示，在 $n=40$ 的图上训练的基础模型准确率最高，这意味着我们只需在 $n=40$ 的图上预训练模型，后续测试均采用这一预训练模型。

4.3.1. 合成数据集

我们同时将 DSP-Transformer 和其他 SOTA 算法应用于 4.1 生成的图数据集上，以评估模型性能。表 1 展示了评估结果。结果表明，DSP-Transformer 的表现优于贪心算法，尤其是在几乎所有 ER 图上表现更好。表 1 的实验结果说明了 DSP-TF 在大多数条件下均取得了更优且稳定的表现，而与 Greedy 的差异更加显著。

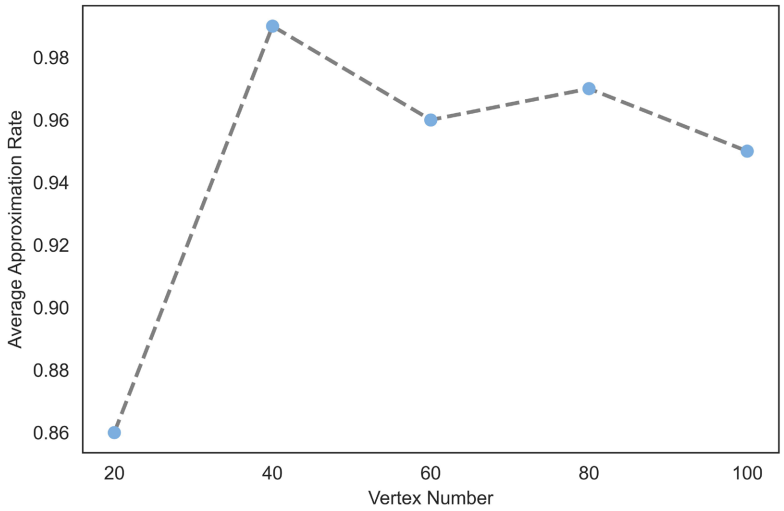

Figure 2. Training graph size and average accuracy for the base model
图 2. 基础模型的训练图规模与平均准确率

Table 1. Evaluation results on ER graphs
表 1. ER 图上的评估结果

顶点数	$p=0.1$			$p=0.5$			$p=0.8$		
	DSP-TF	DSP-RL	Greedy	**DSP-TF**	DSP-RL	Greedy	**DSP-TF**	DSP-RL	Greedy
20	**7.18 ± 0.02**	7.23	7.79	**3.19 ± 0.02**	3.19	3.47	**1.52 ± 0.08**	1.82	1.81
40	**12.05 ± 0.01**	12.12	13.11	**4.09 ± 0.02**	4.11	4.26	**2.22 ± 0.05**	2.24	2.29
80	**17.13 ± 0.02**	17.13	18.22	**5.02 ± 0.04**	5.14	5.28	**2.65 ± 0.04**	2.69	2.78
100	**18.95 ± 0.12**	19.11	19.76	**5.48 ± 0.01**	5.48	5.50	**2.91 ± 0.05**	2.91	2.91
200	**23.12 ± 0.05**	24.07	25.09	**6.21 ± 0.04**	6.43	6.73	**3.27 ± 0.02**	3.29	3.27
300	**27.21 ± 0.07**	27.76	28.90	**6.95 ± 0.05**	7.17	7.15	**3.64 ± 0.04**	3.73	3.64
400	**30.24 ± 0.13**	30.48	31.83	**7.55 ± 0.04**	7.58	7.60	**3.64 ± 0.02**	3.81	3.87
500	**31.89 ± 0.18**	32.90	33.64	**8.00 ± 0.02**	8.00	7.92	**3.52 ± 0.06**	3.88	4.15
800	**35.94 ± 0.21**	37.47	38.11	**8.64 ± 0.08**	8.81	8.67	**4.15 ± 0.03**	4.16	4.29

Alanko, Samu 等人[10]已求得网格图 ■(m,n) 的最优的最优支配数，其中 $m,n\le 29$ 。为了进一步评估，我们在表 2 中部分呈现了将我们模型的解与网格图上已知的最优解进行比较。结果表明，DSP-Transformer 与其他的模型比较下都达到了 SOTA，在绝大多数情况下都能等于或更接近最优解。

Table 2. Evaluation results on grid charts
表 2. 网格图上的评估结果

m	n	$\gamma_{\text{DSP-TF}}$	$\gamma_{\text{DSP-RL}}$	γ_{greedy}	γ_{OPT}
5	4	**6**	6	8	6
5	5	**7**	8	8	7
5	6	**9**	10	11	8
5	7	**10**	11	13	9

续表

5	8	**11**	13	13	11
5	9	**13**	13	16	12
5	10	**14**	15	18	13
6	6	**10**	10	13	10
6	7	**12**	12	14	11
6	8	**12**	13	16	12
6	9	**15**	16	17	14
6	10	**16**	17	20	16
7	7	**15**	17	17	12
7	8	**14**	19	19	14
7	9	**17**	19	21	16
7	10	**19**	19	23	17
8	8	**16**	18	20	16
8	9	**20**	22	22	18
8	10	**23**	23	26	20
9	9	**21**	23	27	20
9	10	**25**	25	29	24
10	10	**25**	27	33	24

4.3.2. T1 基准数据集

在本文中，我们在 T1 数据中选择了原始图，其中每个顶点的权重均设为 1。考虑到图中顶点和边的规模存在差异，T1 可以划分为 52 个图族，每个家族包含 10 个相同大小的实例。我们从这些家族中选取了不同顶点规模的 7 个实例来评估我们的模型。同时，我们还对多种进化算法和贪心算法进行了性能比较。我们将所有模型应用于选定的 T1 数据集，每种图大小下运行 10 个实例，并计算了平均支配数和运行时间。正如表 3 所示，DSP-Transformer 在各实例规模下与最先进的启发式算法表现相当，但在求解时间上明显优于其他方法。

Table 3. Evaluation results on T1 benchmark dataset
表 3. T1 基准数据集上的评估结果

Instance Family	CC^2FS	MWDS	RLS_0	ScBpw	FastDS	GREEDY	DSP-RL	**DSP-TF**
V100E100	33.6	33.6	33.6	33.6	33.6	39.5	33.6	**33.6**
V200E250	61.1	61.1	61.1	61.7	61.1	72.4	65	**61.1**
V250E250	83.3	83.3	83.3	83.3	83.3	100	83.3	**83.3**
V300E300	100	100	100	100	100.1	120.2	100	**100**
V500E500	167	167	167	167	167	201.3	167	**167**
V800E1000	242.5	242.5	242.5	246.5	242.5	288.5	257.7	**242.7**
V1000E1000	333.7	334	333.7	333.7	333.7	399.6	333.7	**333.7**
Average run time (s)	4.88	8.35	//	//	11.23	//	1.13	**1.02**

利用 T1 数据集我们还对 DSP-Transformer 的求解性能进行了分析。从图 3 中，我们可以直观地观察到各算法与 DSP-Transformer 在效率上的比较。在图 3 中，我们将近似比定义为最优支配数与各算法实际结果的比值，近似率越接近 1 代表算法性能越好。同时，图中另一坐标轴表示各算法求解最优解所需的时间，数值越低越好。正如图 3 所示，我们的算法在求解结果上已与启发式算法不相上下，远超贪心算法，而在效率上，则明显优于传统方法。值得注意的是，每当启发式算法遇到新实例时，都需要重新进行搜索；而 DSP-Transformer 只需在小规模图上进行预训练，即可应用于大规模图实例。

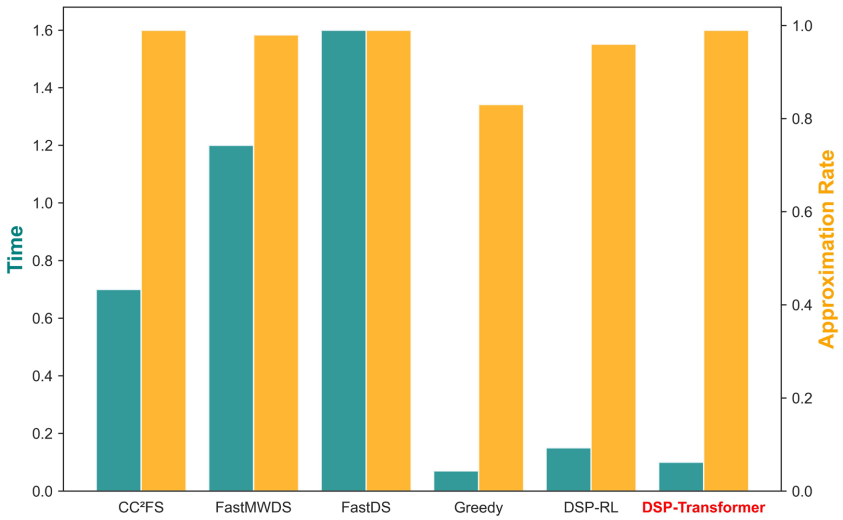

Figure 3. Performance analysis of DSP-Transformer algorithm
图 3. DSP-Transformer 算法性能分析

4.3.3. 真实网络数据集

为了评估我们模型在实际应用中的表现，我们在大规模真实网络上进行了实验，并将实验结果与已知最优解进行了对比，结果见表 4。

Table 4. Evaluation results on real world dataset
表 4. 真实网络数据集上的评估结果

	OPT	ACP-LS	ACO-PP-LS	ACO-LS-S	RLS_0	DSP-RL	**DSP-TF**
david	2	2	2	2	2	2	2
dophins	14	14	14	14	14	15	14
football	12	12	12	13	12	13	12
huck	9	9	9	9	9	12	10

5. 结论

在本文中，我们提出了 DSP-Transformer，一种基于注意力机制的最小支配集问题求解方法。通过结合消息传递机制与 Transformer 结构，我们利用注意力机制更有效地捕获图结构中的潜在关系，相较于传统的消息传递机制编码器，在性能上取得了显著提升，在各个数据集的表现上都更接近最优解。实验结果表明，DSP-Transformer 在合成数据集和标准测试集上均优于现有的启发式算法，在保证解的质量的同时大幅降低求解时间，为求解最小支配集问题提供了一种高效且可扩展的解决方案。

参考文献

[1] Du, D.-Z. and Wan, P.-J. (2012) Connected Dominating Set: Theory and Applications. Springer Science & Business Media.

[2] Haynes, T. W. Hedetniemi, S. and Slater, P. (2013) Fundamentals of Domination in Graphs. CRC Press. https://doi.org/10.1201/9781482246582

[3] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C. and Yu, P.S. (2021) A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, **32**, 4-24. https://doi.org/10.1109/tnnls.2020.2978386

[4] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., *et al*. (2016) Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, **529**, 484-489. https://doi.org/10.1038/nature16961

[5] Sutton, R.S. and Barto, A.G. (2018) Reinforcement Learning: An Introduction, 2nd Edition, The MIT Press.

[6] Ore, O. (1962) Theory of Graphs. American Mathematical Society. https://books.google.com.sg/books?id=g85UAAAAYAAJ

[7] Bondy, J.A. and Murty, U.S.R. (2010) Graph Theory. Springer.

[8] Haynes, T.W., Hedetniemi, S.T. and Henning, M.A. (2020) Topics in Domination in Graphs. Springer. https://link.springer.com/book/10.1007/978-3-030-51117-3

[9] Tarr, J. (2010) Domination in Graphs. USF Tampa Graduate Theses and Dissertations. https://digitalcommons.usf.edu/etd/1786

[10] Alanko, S., Crevals, S., Isopoussu, A., Östergård, P. and Pettersson, V. (2011) Computing the Domination Number of Grid Graphs. *The Electronic Journal of Combinatorics*, **18**, 1-18. https://doi.org/10.37236/628

[11] Anand, R., Aggarwal, D. and Kumar, V. (2017) A Comparative Analysis of Optimization Solvers. *Journal of Statistics and Management Systems*, **20**, 623-635. https://doi.org/10.1080/09720510.2017.1395182

[12] Hopfield, J.J. and Tank, D.W. (1985) "Neural" Computation of Decisions in Optimization Problems. *Biological Cybernetics*, **52**, 141-152. https://doi.org/10.1007/bf00339943

[13] Bello, I., Pham, H., Le, Q.V., Norouzi, M. and Bengio, S. (2016) Neural Combinatorial Optimization with Reinforcement Learning. arXiv: 1611.09940. https://doi.org/10.48550/arXiv.1611.09940

[14] Vinyals, O., Fortunato, M. and Jaitly, N. (2015) Pointer Networks. Advances in Neural Information Processing Systems 28 (NIPS 2015). https://papers.nips.cc/paper/2015/hash/29921001f2f04bd3baee84a12e98098f-Abstract.html

[15] Deudon, M., Cournut, P., Lacoste, A., Adulyasak, Y. and Rousseau, L. (2018) Learning Heuristics for the TSP by Policy Gradient. *Integration of Constraint Programming, Artificial Intelligence, and Operations Research*, Delft, 26-29 June 2018, 170-181. https://doi.org/10.1007/978-3-319-93031-2_12

[16] Chen, X. and Tian, Y. (2019) Learning to Perform Local Rewriting for Combinatorial Optimization. *Proceedings of the* 33*rd International Conference on Neural Information Processing Systems*, Vancouver, 8-14 December 2019, 6281-6292.

[17] Chen, M., Liu, S. and He, W. (2024) Learn to Solve Dominating Set Problem with GNN and Reinforcement Learning. *Applied Mathematics and Computation*, **474**, Article 128717. https://doi.org/10.1016/j.amc.2024.128717

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., *et al*. (2017) Attention Is All You Need. *Proceedings of the* 31*st International Conference on Neural Information Processing Systems*, Long Beach, 4-9 December 2017, 6000-6010. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[19] Kool, W., van Hoof, H. and Welling, M. (2019) Attention, Learn to Solve Routing Problems! arXiv: 1803.08475. https://doi.org/10.48550/arXiv.1803.08475

[20] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. (2017) Proximal Policy Optimization Algorithms. arXiv: 1707.06347. https://doi.org/10.48550/arXiv.1707.06347

[21] Erdös, P. and Rényi, A. (2011) On the Evolution of Random Graphs. In: Newman, M., Barabási, A.-L. and Watts, D.J., Eds., *The Structure and Dynamics of Networks*, Princeton University Press, 38-82. https://doi.org/10.1515/9781400841356.38

[22] Dai, H., Khalil, E.B., Zhang, Y., Dilkina, B. and Song, L. (2017) Learning Combinatorial Optimization Algorithms over Graphs. arXiv: 1704.01665. https://doi.org/10.48550/ARXIV.1704.01665

[23] Jovanovic, R., Tuba, M. and Simian, D. (2010) Ant Colony Optimization Applied to Minimum Weight Dominating Set Problem. *Proceedings of the* 12*th WSEAS International Conference on Automatic Control, Modelling & Simulation*, Catania, 29-31 May 2010, 322-326.