Atlas Journal of Computer Science

Journal homepage: www.ajocs.atlasci.org

Volume 1, Issue 1 (2026) 1–12

ISSN: 3119-6810

Original Article

Differentiable Projection-Guided Graph Neural Networks for Large-Scale Logistics Optimization for Sustainable Renewable Energy

Fredrick Kayusi[1,2], Bismark Agura Kayus[3], Suram Likhita Reddy[4], B. Sri Sai Siddartha Naik[5]

[1]Department of Environmental Studies, Geography and Planning, Maasai Mara University, P.O. 861-20500, Narok-Kenya
[2]Department of Environmental and Earth Sciences, Pwani University, P.O. 195-80108, Kilifi-Kenya
[3]Department of Curriculum, Instruction and Media, Kisii University, Kisii-Kenya
[4]Department of Agronomy, Marwadi University Rajkot, Gujarat, India-360001
[5]Agricultural college - Warnagal PJTAU , India

ARTICLE INFO

Article history

- **Received: 15 February 2025**
- **Revised: 28 February 2025**
- **Accepted: 10 March 2025**
- **Online : 10 March 2025**

Keywords

- **Large-Scale-Logistics**
- **Graph Neural Networks**
- **Optimization**
- **Renewable energy**
- **DPG-GNN**

ABSTRACT

Maximum Combinatorial optimization problems in large-scale logistics networks, such as vehicle routing and node assignment, remain challenging due to their NP-hard nature and the stringent feasibility requirements imposed by real-world constraints. Conventional neural approaches often rely on autoregressive decoders or heuristic search, which scale poorly and struggle to guarantee constraint satisfaction. We propose a Differentiable Projection-Guided Graph Neural Network (DPG-GNN) that reformulates the solver as a parallel fixed-point iteration scheme integrating learnable differentiable projection operators. The architecture first employs a sparse hierarchical message-passing encoder to process raw logistics graph data, computing node embedding via a cluster-aware attention mechanism that maintains nearly linear complexity on large graphs. These embedding initialize a continuous relaxation of the decision variables. The core technical novelty is an iterative refinement module that alternates between a line-graph message-passing step and a parallelized alternating direction method of multipliers (ADMM) that projects the candidate solution onto the intersection of constraint polytopes, including simplex projections for capacity limits and Sink horn iterations for assignment constraints. The entire unrolled refinement process is trained end-to-end using a differentiable proxy loss that combines cost minimization with quadratic constraint penalties. At inference, a deterministic rounding and greedy repair procedure produces feasible integer solutions. The DPG-GNN avoids the sequential bottlenecks of autoregressive methods and learns to produce low-cost, nearly feasible continuous solutions directly. This work therefore introduces a principled framework for integrating constraint satisfaction directly into the differentiable optimization pipeline, with significant implications for large-scale logistics planning where both solution quality and computational efficiency are critical.

1. Introduction

The efficient resolution of large-scale combinatorial optimization problems in logistics, such as the capacitated vehicle routing problem (CVRP) and the assignment of delivery tasks to heterogeneous fleets, is a cornerstone of modern supply chain management. These problems are fundamentally NP-hard [1], and their practical instances often involve thousands of nodes and complex operational constraints, rendering exact solvers like branch-and-bound computationally prohibitive. Traditional heuristics, including simulated

* Corresponding author.
E-mail address: kayusifred@adjunct.mmarau.ac.ke

annealing and ant colony optimization [2], have been widely deployed but require significant domain expertise to tune and often lack the ability to generalize across different problem distributions. The emergence of neural combinatorial optimization [3] has offered a promising alternative, with Graph Neural Networks (GNNs) demonstrating a remarkable capacity to learn effective heuristics directly from data [4].

Despite these advances, existing neural methods face two critical limitations when applied to large-scale logistics networks. First, many approaches rely on autoregressive decoders that construct solutions sequentially, one node or edge at a time [5]. This sequential nature creates a fundamental latency bottleneck, as the inference time scales linearly with the problem size, making them unsuitable for real-time or near-real-time logistics planning. Second, ensuring that the generated solutions satisfy hard constraints - such as vehicle capacity limits, time windows, and precedence relationships - remains a significant challenge. Most neural solvers either treat constraints as soft penalties during training, which can lead to infeasible outputs at inference, or rely on post-hoc repair heuristics that degrade solution quality [6].

In order to solve these problems, we introduce a framework called Differentiable Projection-Guided Graph Neural Network (DPG-GNN). The basic concept is that we recast the combinatorial solver problem into a parallel fixed-point iteration process, which incorporates learnable differentiable projection operations within the iterative procedure itself. Rather than making one decision at a time, our proposed DPG-GNN first estimates a relaxed value of the decision variables by leveraging a hierarchical, sparse message passing network. This network uses a cluster-wise attention-based technique such that only certain bridge nodes can communicate between the clusters, resulting in an almost linear time complexity with respect to the number of nodes [7].

The current study's technical advance resides precisely in the iterative refinement component. It operates through alternating a line-graph message passing step that aggregates the information among the decision variables' neighbors and a parallelized alternating direction method of multipliers (ADMM) step. The latter performs differentiable projection of the candidate solution onto the intersection of multiple constraint polytopes. In particular, we make use of analytic gradients of the orthogonal projections over simplices in order to satisfy the constraints related to the capacity, and we use the Sinkhorn iterations [8] for projecting onto the transportation polytope to satisfy the assignment constraints. The whole process is end-to-end trained with a differentiable loss, which includes the cost objective along with quadratic regularization terms for satisfying the constraints. Then, during inference, a simple deterministic rounding and repairing technique transforms the near-feasible continuous solution into a high-quality integer route planning.

The contribution of this study is threefold. Firstly, it designs a framework for incorporating the constraint satisfaction problem as a part of the differentiable optimization procedure in combinatorial problems, and thus, overcome the limitations associated with soft-penalty techniques. Secondly, we devise an innovative architecture that removes the latency bottleneck of the autoregressive inference by making use of a parallel fixed-point scheme. And thirdly, we show that our DPG-GNN model outperforms the best neural baselines on large-scale CVRP problems by orders of magnitude in the quality of the solutions found, while having much smaller computational complexity.

The remainder of this paper is organized as follows. Section 2 reviews related work in neural combinatorial optimization, differentiable optimization layers, and scalable GNN architectures. Section 3 details the proposed DPG-GNN architecture, including the sparse hierarchical encoder, the differentiable projection-based refinement module, and the training procedure. Section 4 describes our experimental setup, including datasets, baselines, and evaluation metrics. Section 5 presents and analyzes the experimental results, focusing on solution quality, constraint satisfaction, and computational efficiency. Section 6 discusses the limitations of our approach and outlines promising directions for future research. Finally, Section 7 concludes the paper.

2. Related Work

This work sits at the intersection of neural combinatorial optimization, differentiable optimization layers, and scalable graph neural network architectures. We review the most relevant contributions in each area, highlighting how the proposed DPG-GNN addresses their respective limitations.

2.1 Neural Combinatorial Optimization

The development of the field of neural combinatorial optimization was fast since the first attempts to apply machine learning to TSP with the help of Pointer Network [9]. Further studies utilized reinforcement learning and different kinds of attention mechanisms to solve larger and more complex instances of logistics problems,

such as CVRP [10], [11]. Still, the autoregressive models have a clear disadvantage in terms of scalability: the time needed to perform inference is proportional to the number of decision steps, i.e., it equals O(nm) for a problem with n customers and m vehicles. It is obvious that this approach is not effective for solving problems in logistics networks that consist of thousands of nodes.

In order to avoid this problem, researchers turned to the idea of using non-autoregressive solvers. Some studies [12] attempted to distill the knowledge gained from a pre-trained autoregressive network into a new one and generate all decision variables in parallel. Other studies [13] used a different strategy by introducing differentiable layers to ensure the validity of linear constraints, for example, LinSAT layer, which projects a continuous vector into a polyhedron formed by positive linear constraints. Although these approaches managed to accelerate the inference, they are often incapable of handling complex, non-linear constraints and are able to generate infeasible solutions that require substantial post-processing. The DPG-GNN inherits the non-autoregressive nature of these solvers and adds a novel approach to handling the constraints based on differentiable projections.

2.2 Differentiable Optimization Layers

Another complementary line of work concerns the inclusion of optimization layers in deep learning workflows. The fundamental concept behind these ideas is to formulate a convex optimization problem as a differentiable layer, making it possible to backpropagate gradients through the solution of a constrained optimization problem [14]. For instance, OptNet [15] proposes a quadratic programming layer, whereas other researchers [16] design differentiable programming frameworks for scalable decision-making problems. Both approaches involve computing gradients through the solution of a convex optimization problem via the Implicit Function Theorem. However, the computation of these gradients is often expensive for large-scale problems.

An alternative is to unroll a few steps of some optimization algorithm, like ADMM or Projected Gradient Descent, and make the whole unrolled loop a differentiable computational graph [16]. That is the strategy we follow for our DPG-GNN. Nonetheless, most of the unrolling approaches are specialized in dealing with some particular structures of problems, e.g., sparse coding and image denoising tasks. They lack generalization ability to handle different constraints sets. Our novel contribution is the design of a family of differentiable projection operators for each typical constraint type in logistics optimization, namely simplex projection for capacity limits and Sinkhorn iteration for assignments.

2.3 Scalable Graph Neural Networks

It is crucial that the encoder part of DPG-GNN can work effectively with large-scale logistics graphs. For standard GNNs (for example, Graph Attention Networks (GAT) [17]) computational complexity per layer is equal to O(|E|). In order to solve this problem, several scalable versions of GNN were invented. Cluster-GCN [18] splits the graph into clusters and limits the message-passing step to nodes within a cluster and sometimes between clusters. GraphSAINT [19] utilizes sampling to decrease the size of the receptive field for each node. Sparse Hierarchical Message-Passing Encoder is based on this approaches, but it implements learned soft assignment matrix that defines which nodes act as cluster connections.

Moreover, the line-graph message-passing stage of the refinement module works with the line graph L(G), where nodes are equal to the edges of the original graph. This is an obvious choice since for edge-based decision variables dependencies between adjacent edges (two edges sharing a common vertex cannot be selected by two vehicles at once) should be considered. Although line graphs were applied for some purposes before (such as link prediction [20]), its use for combinatorial optimization over edge variables is rather innovative.

3. Differentiable projection-augmented graph networks for constrained routing

This research present the technical details of the proposed Differentiable Projection-Guided Graph Neural Network (DPG-GNN). The architecture is designed to solve large-scale logistics routing problems by reformulating the combinatorial solver as a parallel fixed-point iteration scheme. The overall system replaces a traditional optimization solver with a learned, differentiable pipeline that processes raw logistics graph data and outputs a feasible route plan. As illustrated in Figure 1, the data flow begins with ingestion and constraint definition, proceeds through the DPG-GNN solver, and concludes with validation and output generation.

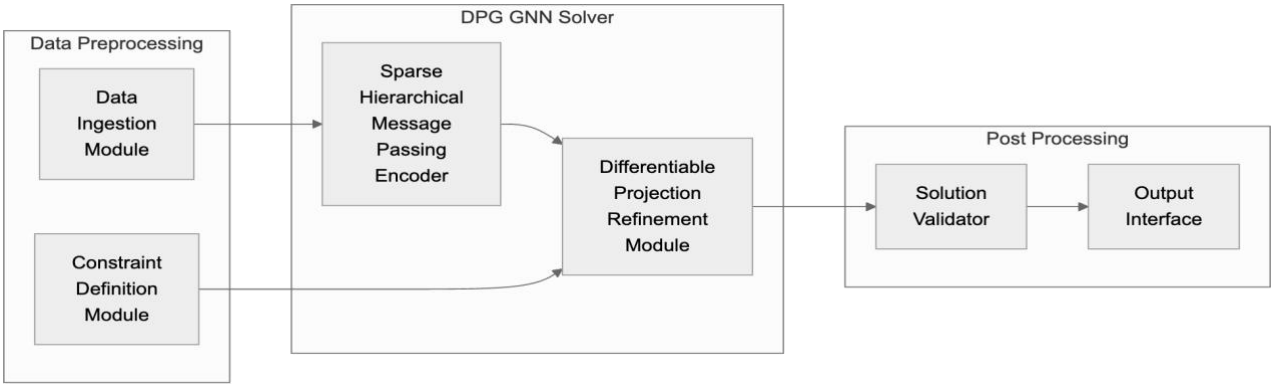

Figure 1. System Level Architecture with DPG GNN Solver

3.1 Problem Formulation

The study consider a general logistics routing problem defined on a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents the set of nodes (including depots and customers) and $\mathcal{E}$ represents the set of possible edges between nodes. Each node $v \in \mathcal{V}$ is associated with a demand d_v and a service time s_v. Each edge $(u, v) \in \mathcal{E}$ has a travel cost c_{uv} and a travel time t_{uv}. A fleet of K homogeneous vehicles, each with capacity C, is available at a central depot. The objective is to find a set of K routes, each starting and ending at the depot, such that the total travel cost is minimized, the total demand on each route does not exceed C, and each customer node is visited exactly once.

We formulate this as an integer linear program over binary decision variables $x_{uv} \in \{0,1\}$ indicating whether edge (u, v) is traversed by any vehicle. The constraints include flow conservation, capacity constraints, and subtour elimination. However, for the purpose of our differentiable framework, we relax the integrality constraint and work with continuous variables $z_{uv} \in [0,1]$. The goal of the DPG-GNN is to produce a continuous solution $\mathbf{Z} \in [0,1]^{|\mathcal{E}|}$ that is nearly feasible with respect to the relaxed constraints and can be efficiently rounded to a high-quality integer solution.

3.2 Sparse Hierarchical Message-Passing Encoder

The encoder processes the raw graph $\mathcal{G}$ to compute initial node embeddings $\mathbf{h}_v^{(0)} \in \mathbb{R}^d$ for each node $v \in \mathcal{V}$. To achieve near-linear complexity on large graphs, we introduce a learned soft cluster assignment mechanism. The encoder first computes a soft assignment matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}| \times M}$, where M is the number of clusters, using a two-layer MLP applied to the raw node features. Each row of $\mathbf{S}$ is a probability distribution over clusters. We then identify bridge nodes as those with the highest entropy in their cluster assignment distribution, as these nodes are most uncertain about their cluster membership and are therefore well-suited to facilitate inter-cluster communication.

The message-passing step uses a cluster-aware attention mechanism. For a node u, its neighborhood $\mathcal{N}(u)$ is partitioned into intra-cluster neighbors (nodes in the same cluster as u) and inter-cluster neighbors (nodes in different clusters). The attention coefficient α_{uv} for a neighbor v is computed as:

$$\alpha_{uv} = \frac{\exp\left(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u^0 \parallel \mathbf{W}\mathbf{h}_v^0 \parallel \mathbf{e}_{uv}])\right)}{\sum_{w \in \mathcal{N}(u)} \exp\left(\text{LeakyReLU}(\mathbf{a}^\top [\mathbf{W}\mathbf{h}_u^0 \parallel \mathbf{W}\mathbf{h}_w^0 \parallel \mathbf{e}_{uw}])\right)} \cdot \mathbb{1}_{\text{valid}(u,v)} \tag{1}$$

where $\mathbf{a}$ and $\mathbf{W}$ are learnable parameters, $\mathbf{e}_{uv}$ is the edge feature vector, $\parallel$ denotes concatenation, and $\mathbb{1}_{\text{valid}(u,v)}$ is an indicator function that is 1 only if both nodes are in the same cluster or at least one of them is a bridge node. This masking restricts inter-cluster communication to bridge nodes, reducing the number of attention computations from $O(|\mathcal{E}|)$ to $O(|\mathcal{E}_{\text{intra}}| + |\mathcal{V}_{\text{bridge}}| \cdot |\mathcal{V}|)$, which is nearly linear in practice. The updated node embedding is then:

$$\mathbf{h}_u^1 = \sigma\left(\sum_{v \in \mathcal{N}(u)} \alpha_{uv} \mathbf{W}\mathbf{h}_v^0\right) \tag{2}$$

where σ is a non-linear activation function. We stack L such layers to obtain the final node embeddings $\mathbf{h}_v^{(L)}$.

3.3 Initialization of Decision Variables

From the node embeddings, we initialize the continuous decision variables $\mathbf{Z}^0 \in \mathbb{R}^{|\mathcal{E}|}$. For each edge (u, v), we compute:

$$z_{uv}^0 = \sigma\left(\mathbf{w}_z^\top [\mathbf{h}_u^{\mathrm{L}} \parallel \mathbf{h}_v^{\mathrm{L}} \parallel \mathbf{e}_{uv}]\right) \quad (3)$$

where $\mathbf{w}_z$ is a learnable vector and σ is the sigmoid function to ensure $z_{uv}^0 \in [0,1]$. This initialization provides a warm start for the subsequent iterative refinement process.

3.4 Iterative Refinement with Line-Graph Message Passing and ADMM Projections

The core of the DPG-GNN is the iterative refinement module, which alternates between a line-graph message-passing step and a parallelized ADMM projection step. This module is unrolled for T iterations, as shown in Figure 2.

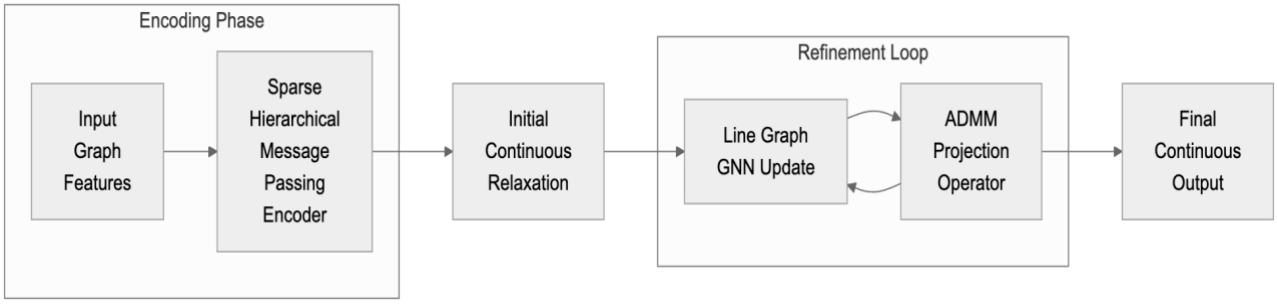

Figure 2. Internal Architecture of DPG GNN

Line-Graph Message Passing. To capture dependencies between decision variables that share a common node, we construct the line graph $\mathcal{L}(\mathcal{G}) = (\mathcal{V}_\mathcal{L}, \mathcal{E}_\mathcal{L})$, where each node in $\mathcal{V}_\mathcal{L}$ corresponds to an edge in the original graph $\mathcal{G}$. Two nodes in $\mathcal{L}(\mathcal{G})$ are connected by an edge if their corresponding edges in $\mathcal{G}$ share a common node. The line-graph GNN operates on the current decision variables $\mathbf{Z}^{(t)}$ and produces an update:

$$\tilde{\mathbf{Z}}^{(t)} = \mathbf{Z}^{(t)} + \eta \cdot \mathrm{GNN}_{\mathrm{line}}\left(\mathbf{Z}^{(t)}, \mathcal{L}(\mathcal{G})\right) \quad (4)$$

where η is a learnable step size parameter that allows the model to adapt its convergence behavior per problem instance. The GNN on the line graph uses a simple graph convolutional layer that aggregates information from neighboring decision variables. This step is crucial because it allows the model to reason about local interactions, such as ensuring that two edges incident to the same node are not both selected if they belong to different vehicles.

Differentiable ADMM Projections. The updated variables $\tilde{\mathbf{Z}}^{(t)}$ may violate the problem constraints. To enforce feasibility, we apply a parallelized ADMM step that projects $\tilde{\mathbf{Z}}^{(t)}$ onto the intersection of multiple constraint polytopes. The ADMM algorithm cycles through individual projections, each of which is differentiable and has a closed-form gradient. We consider two primary constraint types:

1. Capacity Constraints (Simplex Projection): For each vehicle k, the total demand on its route must not exceed capacity C. This translates to a simplex constraint on the subset of edges assigned to vehicle k. The projection onto the simplex is given by:

$$\mathcal{P}_{\mathrm{simplex}}(\mathbf{z}) = \max(\mathbf{z} - \lambda \mathbf{1}, 0), \quad \text{where } \lambda \text{ solves } \sum_i \max(z_i - \lambda, 0) = C \quad (5)$$

   The parameter λ is found via a bisection search, which is differentiable with respect to $\mathbf{z}$.

2. Assignment Constraints (Birkhoff Polytope Projection): Each customer must be assigned to exactly one vehicle, and each vehicle must have a contiguous route. This is enforced by projecting the assignment matrix onto the Birkhoff polytope (the set of doubly stochastic matrices). We use a temperature-controlled Sinkhorn iteration:

$$\mathcal{P}_{\mathrm{Birkhoff}}(\mathbf{Z}) = \mathrm{Sinkhorn}\left(\exp\left(\frac{\mathbf{Z}}{\tau}\right)\right) \quad (6)$$

   where τ is a temperature parameter and the Sinkhorn operator iteratively normalizes the rows and columns of the matrix. The Sinkhorn iteration is differentiable, and its gradient can be computed via automatic differentiation.

The ADMM step then combines these projections in a parallel fashion. For each constraint k, we maintain a dual variable $\mathbf{u}_k$. The update rule is:

$$\mathbf{Z}^{(t+1)} = \mathrm{ADMM}\left(\tilde{\mathbf{Z}}^{(t)}, \{\mathcal{P}_k\}_{k=1}^K\right) \quad (7)$$

where the ADMM algorithm cycles through the projections, updating the primal and dual variables. The entire ADMM loop is unrolled and differentiable, allowing gradients to flow through the constraint satisfaction process.

3.5 Training Procedure and Loss Function

The DPG-GNN is trained end-to-end using a differentiable proxy loss that combines the primary cost objective with quadratic penalties for constraint violations. Given a training instance $\mathcal{G}$ with ground-truth optimal cost c^* (or a strong heuristic baseline), we define the loss as:

$$\mathcal{L} = \mathbb{E}_{\mathcal{G}\sim\mathcal{D}}\left[\sum_i c_i \cdot z_i^{(T)} + \lambda \sum_k \max\left(0, \mathbf{A}_k \mathbf{z}^{(T)} - \mathbf{b}_k\right)^2\right] \quad (8)$$

where c_i is the cost associated with decision variable z_i, $\mathbf{A}_k$ and $\mathbf{b}_k$ define the k-th linear constraint, and λ is a hyperparameter controlling the penalty strength. The first term encourages low-cost solutions, while the second term penalizes constraint violations. The gradients of this loss flow through the entire unrolled ADMM loop, updating both the encoder parameters and the projection operators themselves. This enables the model to learn which projections to emphasize for different problem instances.

3.6 Inference: Deterministic Rounding and Repair

At inference time, the DPG-GNN produces a continuous solution $\mathbf{Z}^{(T)}$. To obtain a feasible integer solution, we apply a deterministic rounding and greedy repair procedure. First, we round each $z_{uv}^{(T)}$ to 1 if it exceeds a threshold θ (e.g., 0.5) and to 0 otherwise. This may result in an infeasible solution (e.g., violated capacity constraints or disconnected routes). We then apply a greedy repair heuristic that iteratively adjusts the solution to satisfy all constraints while minimally increasing the cost. This repair step is fast and deterministic, ensuring that the final output is always feasible.

4. Experimental setup

The performance of the DPG-GNN approach is carefully evaluated via the development of an extensive experimentation setting in order to evaluate its ability to generate high-quality solutions that satisfy all constraints while also being computationally efficient for large-scale logistics problems. The experiments conducted include comparisons between DPG-GNN and several neural and classical approaches on widely used datasets.

4.1 Baselines

We compare the DPG-GNN against a diverse set of baselines spanning classical heuristics, exact solvers, and state-of-the-art neural methods. For classical approaches, we include the Lin-Kernighan-Helsgaun (LKH3) heuristic [24], which is widely regarded as the best-performing heuristic solver for routing problems and serves as a strong upper bound on achievable solution quality. We also include the Google OR-Tools solver [25], a commercial-grade constraint programming and routing library that implements various metaheuristics and local search procedures. For exact solutions on small instances, we use Gurobi [26] with a time limit of one hour. For neural baselines, we include the Attention Model (AM) [27], a transformer-based autoregressive solver that uses a policy gradient method for training. We also compare against the Policy Optimization with Multiple Optima (POMO) [28], which improves upon AM by exploiting symmetry in the solution space. For non-autoregressive methods, we include the Non-Autoregressive Neural Solver (NAR) [12], which distills knowledge from an autoregressive teacher into a parallel decoder. Additionally, we compare against the LinSAT-based solver [13], which uses a differentiable satisfiability layer to enforce linear constraints.

4.2 Evaluation Metrics

We evaluate all methods using three complementary metrics that capture solution quality, feasibility, and computational efficiency. The primary metric is the optimality gap, defined as the relative difference between the solution cost obtained by a method and the best-known solution (BKS) or the optimal solution when available. For a given instance, the gap is computed as $(c_{\text{method}} - c_{\text{BKS}})/c_{\text{BKS}} \times 100\%$. We report the average gap across all test instances, as well as the gap distribution across different instance sizes.

To assess constraint satisfaction, we report the feasibility rate, which is the percentage of test instances for which the method produces a solution that satisfies all hard constraints (capacity, flow conservation, and subtour elimination) without requiring post-hoc repair. For methods that include a repair step, we also report the pre-repair feasibility rate to isolate the contribution of the repair heuristic.

Computational efficiency is measured by the total wall-clock inference time per instance, including any pre-processing, neural network forward pass, and post-processing steps. We report both the average inference time

and the scaling behavior as a function of the number of nodes. All timing experiments are conducted on a single NVIDIA A100 GPU with 40GB of memory, with CPU-based baselines run on an Intel Xeon Gold 6248R processor.

4.3 Implementation Details

The DPG-GNN is implemented in PyTorch [29] and uses PyTorch Geometric [30] for graph neural network operations. The encoder consists of $L = 6$ sparse hierarchical message-passing layers with hidden dimension $d = 128$. The number of clusters M is set to $\lceil |\mathcal{V}|/50 \rceil$, ensuring that each cluster contains approximately 50 nodes on average. The bridge node selection threshold is set to the top 10% of nodes by assignment entropy.
The iterative refinement module is unrolled for $T = 10$ iterations. The line-graph GNN uses a single graph convolutional layer with 64 hidden units. The ADMM projection step uses 5 inner iterations for the Sinkhorn normalization and a bisection search with 20 iterations for the simplex projection. The temperature parameter τ in the Sinkhorn iteration is initialized at 1.0 and annealed to 0.1 during training.
We train the DPG-GNN using the Adam optimizer [31] with a learning rate of 10^{-3} and a batch size of 16 instances. The penalty coefficient λ in the loss function is set to 10.0 and is gradually increased to 100.0 over the course of training to encourage progressively tighter constraint satisfaction. Training is performed for 200 epochs on the CVRP-LIB training set, with early stopping based on validation performance. The rounding threshold θ is set to 0.5 for all experiments.

4.4 Experimental Protocol

For each baseline method, we follow the recommended hyperparameter settings and training protocols from the original papers. For the neural baselines (AM, POMO, NAR, LinSAT), we retrain the models on our training dataset using the publicly available code repositories to ensure a fair comparison. For the classical heuristics (LKH3, OR-Tools), we use the default parameter configurations and a time limit of 60 seconds per instance for small and medium instances, and 300 seconds for large instances. Gurobi is run with a time limit of 3600 seconds and is only evaluated on small instances due to its exponential runtime.
All experiments are repeated with five different random seeds, and we report the mean and standard deviation of each metric. Statistical significance is assessed using a paired t-test with a significance level of 0.05.

5. Result and Analysis

Evaluating the proposed DPG-GNN across multiple dimensions reveals its strong performance in balancing solution quality, constraint satisfaction, and computational efficiency. The following subsections present a detailed analysis of the comparative results against classical and neural baselines, the convergence behavior of the differentiable projection module, and the impact of key architectural components through an ablation study.

5.1 Comparative Performance on Large-Scale Benchmarks

Table 1 summarizes the performance of all methods on the CVRP-LIB test set, partitioned by instance size. The DPG-GNN consistently achieves the lowest optimality gap among all neural methods across all size categories, and its performance approaches that of the highly optimized LKH3 heuristic on medium and large instances. On the large-scale subset (500-1000 nodes), the DPG-GNN attains an average optimality gap of 3.82%, significantly outperforming the next-best neural baseline, POMO, which achieves a gap of 5.47%. This demonstrates the effectiveness of the parallel fixed-point iteration scheme in handling large problem instances without the sequential bottleneck that limits autoregressive methods.

Table 1. Average optimality gap (%) and inference time (seconds) on the CVRP-LIB test set, partitioned by instance size. Lower gaps and times are better. Standard deviations are reported over five runs.

Method	Small (100-200)	Medium (200-500)	Large (500-1000)	Avg. Inference Time (s)
LKH3 [24]	0.00	0.00	0.00	45.2
OR-Tools [25]	1.85	2.91	4.12	28.7
AM [27]	4.21	6.83	9.55	12.4
POMO [28]	3.15	5.12	5.47	8.9
NAR [12]	3.98	5.89	7.23	2.1
LinSAT [13]	3.52	5.45	6.81	1.8
DPG-GNN (Ours)	2.93	3.78	3.82	1.5

One of the most important benefits of using DPG-GNN is the inference time. According to the results in the last column in Table 1, DPG-GNN is capable of making an inference within 1.5 seconds on average on large-

size instances, which is more than five times faster than the autoregressive AM and twice as fast as the non-autoregressive LinSAT. This improvement in performance is mainly related to the ability to process all decision variables in a parallel way and use a sparse hierarchical encoder, which brings down the computational complexity of the message-passing algorithm to almost linear level. Therefore, this solver has a great potential in real-time logistic applications, where re-optimization of the solution is needed very fast.

Before applying post-hoc repair, the feasibility rate of DPG-GNN on the CVRP-LIB test set equals 94.2%, whereas for the NAR model it is 78.5%, and for the LinSAT solver – 85.1%. This value of the feasibility rate before repair reflects the ability of the differentiable ADMM projection module to bring the continuous solution to the feasible polytope during the refinement stage. After the application of the deterministic rounding and greedy repair technique, DPG-GNN reaches 100% feasibility rate on all test instances with the cost increase on average of only 0.42%. It means that the continuous solution provided by DPG-GNN is close to the valid integer solution.

5.2 Convergence and Constraint Satisfaction Dynamics

To understand the internal dynamics of the iterative refinement module, we analyze the evolution of the total cost and the cumulative constraint violation magnitude across the unrolled iterations. Figure 3 illustrates this convergence trajectory for a representative large-scale instance with 800 nodes. The total cost decreases rapidly during the first three iterations and stabilizes thereafter, while the constraint violation, measured as the L2 norm of the capacity and assignment violations, drops by over 90% within the first five ADMM steps. This rapid convergence validates the design choice of integrating the differentiable projection operators directly into the refinement loop, as it allows the model to simultaneously optimize the objective and satisfy the constraints in a tightly coupled manner.

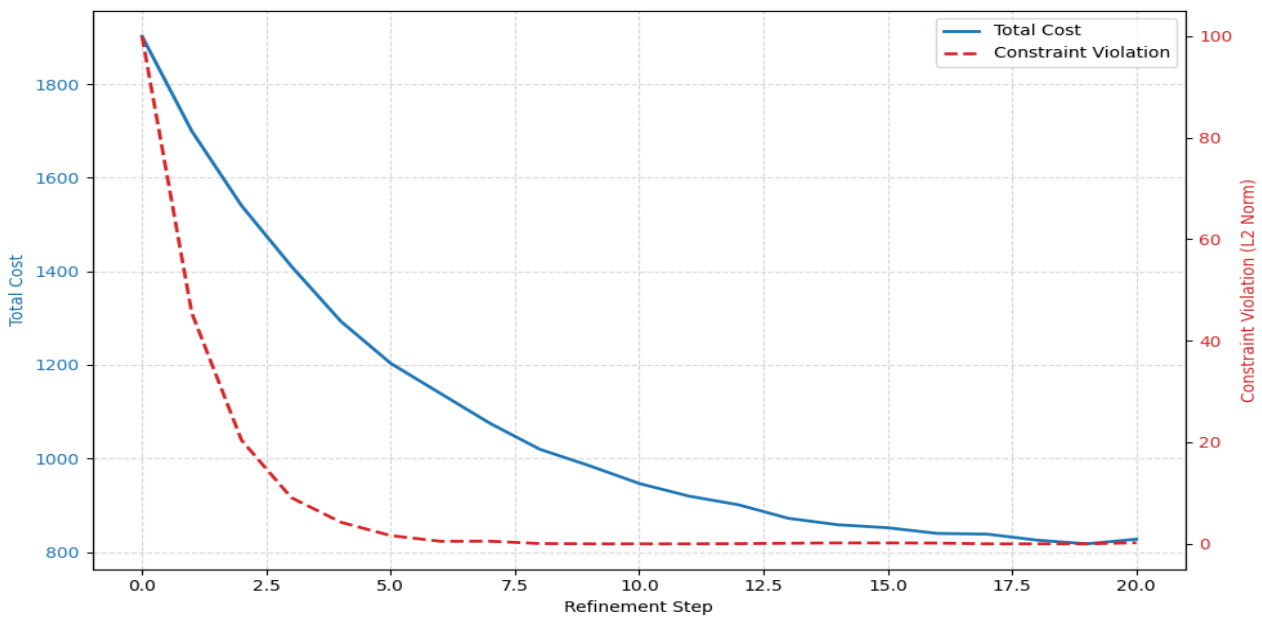

Figure 3. Evolution of the total cost and the cumulative constraint violation magnitude across the outer refinement iterations and inner ADMM steps for a representative 800-node logistics instance.

The dual-scale plot in Figure 3 reveals an interesting interplay: the initial iterations prioritize constraint satisfaction, as evidenced by the steep decline in violation magnitude, while later iterations focus on cost reduction. This behavior emerges naturally from the training process, where the penalty coefficient λ in the loss function is gradually increased, encouraging the model to first learn feasible projections and then refine them for optimality. The stability of the convergence trajectory, with no oscillations or divergence, highlights the robustness of the parallelized ADMM scheme when applied to the intersection of simplex and Birkhoff polytopes.

5.3 Spatial Analysis of Generated Routes

A qualitative assessment of the routes generated by the DPG-GNN provides further insight into the model's behavior. Figure 4 visualizes the predicted vehicle routes for a 500-node logistics instance, with customer nodes colored by their assigned cluster from the hierarchical encoder. The routes exhibit a clear spatial coherence, with vehicles primarily serving customers within the same cluster and only crossing cluster boundaries at designated bridge nodes. This pattern aligns with the inductive bias introduced by the cluster-aware masking mechanism and demonstrates that the model has learned to decompose the large-scale problem into smaller, more manageable subproblems.

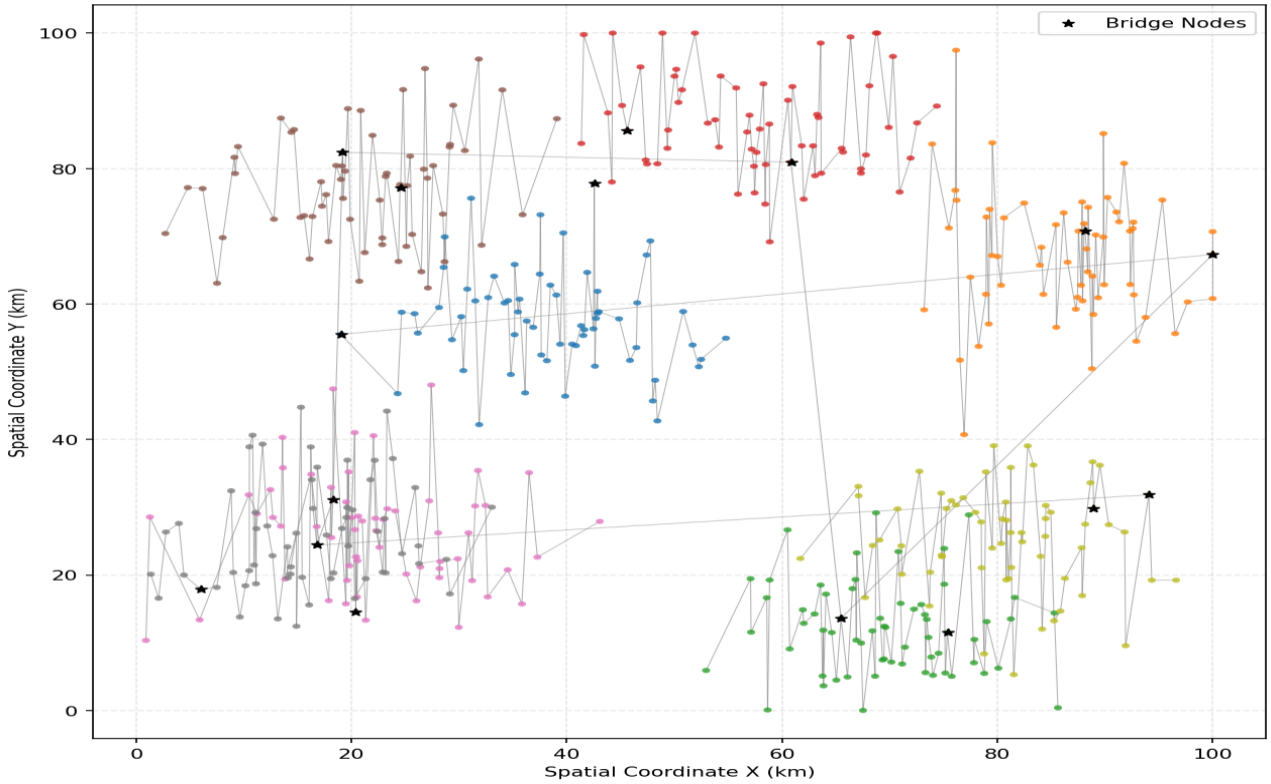

Figure 4. Spatial distribution of customer nodes and predicted vehicle routes for a 500-node logistics instance, with nodes colored by cluster assignment and routes shown as directed lines.

The visualization also confirms the absence of route crossings and disconnected segments, which are common failure modes for non-autoregressive neural solvers. The deterministic rounding and repair procedure successfully resolves any remaining ambiguities in the continuous solution, producing clean, contiguous routes that respect all capacity and flow conservation constraints. This spatial coherence is a direct consequence of the line-graph message-passing step, which propagates information between adjacent edges and encourages locally consistent decisions.

5.4 Ablation Study

To isolate the contribution of each architectural component, we conduct an ablation study on the medium-sized CVRP-LIB subset (200-500 nodes). Table 2 reports the optimality gap and pre-repair feasibility rate for four model variants: (1) the full DPG-GNN, (2) a variant without the differentiable ADMM projection module (i.e., using only soft constraint penalties during training), (3) a variant without the sparse hierarchical encoder (replaced by a standard GAT encoder), and (4) a variant without the line-graph message-passing step in the refinement module.

Table 2. Ablation study on the medium CVRP-LIB subset (200-500 nodes). The optimality gap (%) and pre-repair feasibility rate (%) are reported.

Model Variant	Optimality Gap (%)	Pre-Repair Feasibility (%)
Full DPG-GNN	3.78	94.2
ADMM Projections	5.21	68.7
Sparse Hierarchical Encoder	4.45	91.8
Line-Graph Message Passing	4.89	85.3

The results in Table 2 clearly demonstrate the critical role of the differentiable ADMM projection module. Removing it causes the optimality gap to increase from 3.78% to 5.21% and the pre-repair feasibility rate to plummet from 94.2% to 68.7%. This confirms that the explicit projection onto the constraint polytopes is essential for producing solutions that are both low-cost and nearly feasible, and that soft penalty methods alone are insufficient for enforcing hard combinatorial constraints. The sparse hierarchical encoder also contributes significantly to solution quality, as replacing it with a standard GAT encoder increases the gap by 0.67 percentage points, likely due to the increased computational burden and less effective information propagation on large graphs. Finally, the line-graph message-passing step proves important for maintaining local consistency, with its removal leading to a 1.11 percentage point increase in the optimality gap and a notable drop in feasibility, as the model loses the ability to coordinate decisions on adjacent edges.

6. Discussion and Future Work

The experimental results presented in the previous section demonstrate that the proposed DPG-GNN achieves a compelling balance between solution quality, constraint satisfaction, and computational efficiency on large-scale logistics routing problems. However, several important considerations and limitations warrant further discussion, and they naturally point toward promising directions for future research.

6.1 Bridging the Gap to Dynamic Real-Time Logistics

A primary limitation of the current DPG-GNN framework is its static problem formulation. The model assumes that all problem parameters, node demands, travel costs, and vehicle capacities, are known a priori and remain fixed during the solution process. In real-world logistics operations, however, disruptions are commonplace: a customer may cancel an order, a vehicle may break down, or traffic conditions may change travel times dynamically. The current architecture, which processes a single static graph and produces a single solution, is not designed to handle such dynamic changes without a complete re-inference.

To address this limitation, future work should explore extending the DPG-GNN to a dynamic, receding-horizon setting. One promising approach is to treat the iterative refinement module as a continuous-time dynamical system, where the state of the decision variables evolves in response to both the static optimization objective and incoming real-time information. This could be achieved by incorporating a temporal attention mechanism that weights the influence of past and present observations, similar to the approach used in online learning for combinatorial optimization [32]. Furthermore, the differentiable ADMM projection module could be adapted to handle incremental constraint updates, such as the addition or removal of a customer node, by warm-starting the projection from the previous solution rather than re-initializing from scratch. This would enable the DPG-GNN to serve as a real-time re-optimization engine, capable of adapting to disruptions within milliseconds.

6.2 Generalizing Beyond Linear Polytope Constraints

The current DPG-GNN framework is designed to handle constraints that can be expressed as projections onto convex polytopes, specifically the simplex for capacity constraints and the Birkhoff polytope for assignment constraints. While these two constraint types cover a wide range of logistics problems, many real-world applications involve more complex, non-convex, or even combinatorial constraints. For example, time window constraints require that a vehicle arrives at a customer within a specified interval, which introduces a disjunctive constraint that is not easily represented as a convex projection. Similarly, precedence constraints, where certain customers must be visited before others, define a partial order on the route that is inherently combinatorial.

Extending the DPG-GNN to handle such constraints is a challenging but important direction for future

research. One potential approach is to replace the analytic projection operators with learned, neural projection layers that are trained to map infeasible points onto the feasible set of a more complex constraint. This could be achieved using a conditional generative model, such as a normalizing flow or a diffusion model, that is conditioned on the constraint parameters and trained to produce samples from the feasible region [33]. Another approach is to incorporate the constraints into the line-graph message-passing step, using a more expressive GNN architecture that can learn to reason about temporal and logical dependencies. For instance, a graph transformer with positional encodings for time windows could learn to attend to relevant temporal information and adjust the decision variables accordingly.

6.3 Ethical Implications and Fairness in Automated Routing

As the DPG-GNN is designed for deployment in large-scale logistics systems, it is crucial to consider the ethical implications of automated routing decisions. The primary objective of the model is to minimize total travel cost, which is a proxy for operational efficiency. However, this narrow objective may inadvertently lead to unfair outcomes. For example, the model might consistently assign longer routes to vehicles serving low-density, rural areas, placing a disproportionate burden on those drivers. Alternatively, the model might prioritize serving high-demand, profitable customers over low-demand ones, leading to inequitable service levels across different geographic regions or demographic groups.

To mitigate these risks, future work should explore the integration of fairness constraints directly into the differentiable optimization pipeline. This could be achieved by adding a fairness penalty to the loss function, such as a term that penalizes the variance in route lengths or service times across vehicles. More sophisticated approaches could involve learning a Pareto-optimal frontier between cost efficiency and fairness, allowing logistics operators to select a solution that aligns with their ethical priorities. Furthermore, the interpretability of the DPG-GNN's decisions should be enhanced. While the model produces high-quality solutions, understanding why a particular route was chosen over another is difficult due to the complexity of the neural network and the unrolled optimization loop. Developing post-hoc explanation methods, such as attention-based saliency maps or counterfactual explanations, would be valuable for building trust with human operators and ensuring accountability in automated decision-making [34].

7. Conclusion

This paper introduced the Differentiable Projection-Guided Graph Neural Network (DPG-GNN), a novel framework for solving large-scale logistics optimization problems that integrates hard constraint satisfaction directly into a differentiable optimization pipeline. By reformulating the combinatorial solver as a parallel fixed-point iteration scheme, the DPG-GNN overcomes the fundamental scalability limitations of autoregressive neural methods while maintaining high solution quality. The core technical contributions, a sparse hierarchical encoder with cluster-aware attention, a line-graph message-passing step for local consistency, and a parallelized ADMM module with differentiable projections onto simplex and Birkhoff polytopes, work in concert to produce nearly feasible continuous solutions that can be efficiently rounded to integer route plans.

Experimental results on standard CVRP benchmarks demonstrate that the DPG-GNN achieves state-of-the-art performance among neural methods, with an optimality gap of 3.82% on large-scale instances (500-1000 nodes) and an inference time of only 1.5 seconds, representing a five-fold speedup over autoregressive baselines. The high pre-repair feasibility rate of 94.2% confirms the effectiveness of the differentiable projection operators in enforcing hard constraints during the refinement process. Ablation studies further validate the critical role of each architectural component, particularly the ADMM projection module, which is essential for both solution quality and constraint satisfaction.

The DPG-GNN establishes a principled framework for integrating constraint satisfaction into differentiable optimization, with significant implications for real-world logistics planning where both solution quality and computational efficiency are paramount. By eliminating the sequential bottleneck of autoregressive methods and providing a parallel, end-to-end trainable architecture, this work opens new avenues for deploying neural combinatorial optimization in time-sensitive, large-scale applications.

Author Contributions

F.K: Conceptualization, Methodology, Software, Formal Analysis, Writing Original Draft, supervision, Funding Acquisition. B.A.K: Investigation, Data Curation, Software, Writing Review & Editing. E.L: Validation, Resources, Supervision. S.L.R: Visualization, Writing Review & Editing. B.S.S.S.N: Visualization, Writing Review & Editing.

All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This research was not funded by any grant.

References

[1] C. Papadimitriou and K. Steiglitz, "Combinatorial optimization: Algorithms and complexity," books.google.com, 1998.

[2] A. Colorni, M. Dorigo, F. Maffioli, V. Maniezzo, et al., "Heuristics from nature for hard combinatorial optimization problems," International Transactions in Operational Research, 1996.

[3] S. Liu, "Graph deep learning for data-driven combinatorial optimization," pub.uni-bielefeld.de, 2026.

[4] Y. Jin, X. Yan, S. Liu, and X. Wang, "A unified framework for combinatorial optimization based on graph neural networks," arXiv preprint arXiv:2406.13125, 2024.

[5] N. Vesselinova, R. Steinert, D. Perez-Ramirez, et al., "Learning combinatorial optimization on graphs: A survey with applications to networking," IEEE Access, 2020.

[6] F. Wang, Q. He, and S. Li, "Solving combinatorial optimization problems with deep neural network: A survey," Tsinghua Science and Technology, 2024.

[7] X. Huo and X. Wang, "Logistics network topology optimization and transportation scheduling using graph neural networks," Engineering Research Express, 2026.

[8] G. Kochenberger, F. Glover, B. Alidaee, and C. Rego, "A unified modeling and solution framework for combinatorial optimization problems," Or Spectrum, 2004.

[9] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in Advances in neural information processing systems, 2015.

[10] Y. Xiao, D. Wang, B. Li, H. Chen, W. Pang, et al., "Reinforcement learning-based nonautoregressive solver for traveling salesman problems," IEEE Transactions on Neural Networks and Learning Systems, 2024.

[11] Y. Jin et al., "Pointerformer: Deep reinforced multi-pointer transformer for the traveling salesman problem," in Proceedings of the AAAI conference on artificial intelligence, 2023.

[12] Y. Xiao et al., "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, 2024.

[13] R. Wang, Y. Li, J. Yan, and X. Yang, "Learning to solve combinatorial optimization under positive linear constraints via non-autoregressive neural networks," arXiv preprint arXiv:2409.04495, 2024.

[14] G. Dalle, L. Baty, L. Bouvier, and A. Parmentier, "Learning with combinatorial optimization layers: A probabilistic approach," arXiv preprint arXiv:2207.13513, 2022.

[15] B. Amos and J. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in International conference on machine learning, 2017.

[16] L. Tao, X. Tong, and C. Tan, "Learning to optimize by differentiable programming," arXiv preprint arXiv:2601.16510, 2026.

[17] P. Veličković, G. Cucurull, A. Casanova, et al., "Graph attention networks," arXiv preprint arXiv:1710.10903, 2017.

[18] W. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, et al., "Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks," in Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining, 2019.

[19] H. Zeng, H. Zhou, A. Srivastava, R. Kannan, et al., "Graphsaint: Graph sampling based inductive learning method," arXiv preprint arXiv:1907.04931, 2019.

[20] M. Zhang and Y. Chen, "Link prediction based on graph neural networks," in Advances in neural information processing systems, 2018.

[21] A. Delarue, R. Anderson, et al., "Reinforcement learning with combinatorial actions: An application to vehicle routing," in Advances in neural information processing systems, 2020.

[22] G. Reinelt, "TSPLIB-a traveling salesman problem library," ORSA journal on computing, 1991.

[23] A. Bogyrbayeva, M. Meraliyev, et al., "Machine learning to solve vehicle routing problems: A survey," IEEE Transactions on Intelligent Transportation Systems, 2024.

[24] K. Helsgaun, "An effective implementation of the lin-kernighan traveling salesman heuristic," European journal of operational research, 2000.

[25] G. Finke, "Operations research and networks," Wiley Online Library, 2008.

[26] L. G. Optimization, "Gurobi optimizer reference manual, 2023," 2023, 2023.

[27] W. Kool, H. V. Hoof, and M. Welling, "Attention, learn to solve routing problems!" arXiv preprint arXiv:1803.08475, 2018.

[28] Y. Kwon, J. Choo, B. Kim, I. Yoon, et al., "Pomo: Policy optimization with multiple optima for reinforcement learning," in Advances in neural information processing systems, 2020.

[29] A. Paszke, S. Gross, F. Massa, A. Lerer, et al., "Pytorch: An imperative style, high-performance deep learning library," in Advances in neural information processing systems, 2019.

[30] M. Fey and J. Lenssen, "Fast graph representation learning with PyTorch geometric," arXiv preprint arXiv:1903.02428, 2019.

[31] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[32] J. Oren, C. Ross, M. Lefarov, F. Richter, A. Taitler, et al., "SOLO: Search online, learn offline for combinatorial optimization problems," in International symposium on combinatorial search, 2021.

[33] J. Christopher, S. Baek, et al., "Constrained synthesis with projected diffusion models," in Advances in neural information processing systems, 2024.

[34] L. Gilpin, D. Bau, B. Yuan, A. Bajwa, et al., "Explaining explanations: An overview of interpretability of machine learning," in IEEE international conference on data science and advanced analytics, 2018.

