GRAPH LEARNING-BASED DIRECTED ABSTRACTION OF COMBINATORIAL SPACES FOR ORDER-PRESERVING SEARCH IN MIXED-COMBINATORIAL NON-LINEAR OPTIMIZATION

by

Gishnu Madhu

May 2026

A dissertation submitted to the
Faculty of the Graduate School of
the University at Buffalo, State University of New York
in partial fulfillment of the requirements for the
degree of

Master of Science in Computer Science and Engineering

Department of Computer Science and Engineering

Acknowledgments

It was during the second semester of my Master's coursework that I applied for the opportunity to work with Dr. Souma Chowdhury at the Adaptive Design Algorithms and Modeling Systems (ADAMS) Lab at the University at Buffalo. The breadth and complexity of the research topics available at the lab deeply excited me, and after discussions, Prof. Chowdhury agreed to serve as my advisor.

During my initial semesters, I explored a variety of research topics, and by my third semester, I was introduced to the field of optimization in mixed combinatorial spaces. Although it initially took time for me to fully grasp the underlying concepts, I quickly realized the importance and impact of this field. It has been a privilege to contribute, even in a small way, to research in such a challenging and evolving domain. I am sincerely grateful to Prof. Chowdhury for introducing me to this field and for providing me with the opportunity to work on problems that expanded my understanding of abstract mathematics, optimization, algorithms and deep learning. Through my work at ADAMS Lab, I was also introduced to the concepts of co-design strategies and optimization in engineering design spaces, which significantly shaped my academic growth and research perspective.

I would also like to express my sincere gratitude to Feng Liu, Senior Ph.D. candidate in Mechanical and Aerospace Engineering at the University at Buffalo, who has been a constant source of inspiration, guidance, and mentorship throughout my journey toward completing this thesis. I greatly benefited from the frameworks and code he developed, which served as the foundation for my thesis work and further research in this field. His

support in preparing figures and refining the writing of this thesis has been invaluable. I would also like to thank all the members of the ADAMS Lab for their insightful comments, constructive feedback, and encouragement during the preparation of this thesis.

I am equally grateful to Prof. Karthik Dantu and Prof. Ryan St. Pierre for their valuable reviews, suggestions, and comments, all of which helped shape and strengthen this thesis.

Finally, I would like to thank my father, Madhu R. Nair, my mother, Vinodini Madhu, my sister, Sandra Madhu, and my girlfriend, Shweta Sapaliga, for their unwavering support, encouragement, and concern for my well-being throughout the entire duration of my work toward completing this thesis.

Table of Contents

List of Tables

List of Figures

Abstract

Mixed combinatorial nonlinear programming (MCNLP) problems are a challenging class of optimization problems that arise in engineering design, optimal planning, and robot codesign applications. Traditional approaches to represent the combinatorial space in conjunction with the continuous space include integer deification or binary representations, which however introduce spurious (unintended relations) between combinations, leads to substantial increase in the search space dimensionality, and/or require imposing additional compatibility constraints. Instead, our approach draws motivation from recent developments in robot codesign and network routing domains that aim to learn search heuristics over combinatorial spaces using Graph Neural Networks (GNN). We present a first-of-its-kind structured abstraction of the combinatorial space by learning how to map a set of combinatorial choices given as a fully connected undirected graph into a directed graph where edges express direct improvement. The mapping is performed by an architecture we propose, namely, Edge Flow Graph Network (EFGN) trained with pairwise difference regression over a loss function that uses a special cyclical regularizer. The presented direction-aware abstraction model provides a potentially more scalable and interpretable retrieval of combinations compared to the original recommendation system in that framework. For evaluation purposes, the optimization framework is implemented using the Mixed discrete particle swarm optimization algorithm and is used to solve analytical benchmark problems that vary in the numbers of combinations and design variables. This approach is also applied to an engineering codesign problem, namely design of the morphology and control

parameters of an active tether-net system used for space debris capture. Compared to a baseline using indexification of the allowed combinations, the optimization using the GCN based recommender consistently provides significantly better mean value of the optimum and greater robustness across multiple runs of PSO in these benchmark and engineering design problems.

Chapter 1

Introduction

Optimization is the process of selecting the best solution from a set of feasible alternatives based on a defined objective function and subject to constraints. Formally, it involves determining values of decision variables that minimize or maximize an objective while satisfying system constraints.

Optimization problems can be classified based on the structure of their objective function, variables, and constraints. A *linear optimization* problem is where both the objective function and constraints are linear, whereas *nonlinear optimization* involves at least one nonlinear component. Based on the domain of the decision variables, problems may be continuous, where variables take real values, or discrete, where variables are restricted to integer or binary values, as in *combinatorial optimization*. Additionally, optimization problems may be *constrained*, requiring the solution to satisfy a set of equality or inequality constraints, or *unconstrained*, where no such restrictions are imposed. These distinctions are fundamental in determining the appropriate modeling and solution techniques for a given problem. In addition to the basic classification of optimization problems, an important subclass arises based on the structure of the variables and the geometry of the feasible region. One such class is the Mixed Integer Programming (MIP) problem, where the domain is mixed, that is, some variables take integer values while others are continuous. Depending

on the nature of the objective function and constraints, optimization problems may also be classified as *convex* or *non-convex*. Convex optimization problems are characterized by convex objective functions and feasible regions, ensuring that any local optimum is globally optimal and enabling efficient solution methods. In contrast, non-convex problems may contain multiple local optima and are generally more difficult to solve. A special case of MIP is Mixed Integer Linear Programming (MILP), where both the objective function and constraints are linear, whereas in a Mixed Integer Non Linear Programming (MINLP), at least one component is nonlinear. While MILP benefit from well-developed and efficient solution techniques, MINLP are generally more challenging due to the combination of combinatorial structure and nonlinear system behavior. We have another class called Mixed Combinatorial Non Linear Programming (MCNLP) problems, in which the design space combines combinatorial choices and continuous domains. This thesis focuses on designing and optimizing the MCNLP problem, which we discuss in detail. But before that, we will discuss and go through Combinatorial optimization (CO) and MINLP, since the literature review in these two domains will give us the necessary insight to proceed with the formulations and challenges of operating in the MCNLP domain.

1.1 Combinatorial optimization

CO problems objective remains in finding an optimal solution from given finite possibilities. Unlike continuous optimization, where variables can take any real value within a given range, classical CO operates strictly within integer or categorical domains. These problems tend to be difficult because, although the search space is a finite set of possibilities, the underlying number of possible combinations or discrete choices increases exponentially with problem size. Examples of these problems are network management [15], the traveling salesman problem (TSP) [16], and graph coloring [15]. For example in TSP, n nodes (representing cities) and the pairwise distance between them is given and the objective is to

find the shortest Hamiltonian cycle between them, which is basically the path which goes to each of nodes exactly once and returns to the starting point. The decision variables are binary, indicating whether a connecting edge is traversed or not. The key challenge is the combinatorial explosion of the solution space, that is, the number of feasible paths grows factorially with the number of nodes, specifically on the order of $(n-1)!$. To put it to perspective, a 5 nodes problem has 24 paths to check , while a 10 nodes one has 362,880 and a 20 node problem has 1.21×10^{17}, Thus even a brute force technique becomes intractable quickly with increase in nodes.

But there exists a much more complex set of optimization problems, which are generally seen in many complex real-world engineering and scientific problems and are characterized by the simultaneous optimization of discrete choices along with continuous parameters. Examples of such types of problems commonly exist in the field of robot planning [7], spacecraft design [18], and task planning [21]. The presence of continuous variables distinguishes these problems from classical CO problems.

1.2 Mixed-Integer Nonlinear Optimization

This class is important because it unifies discrete design choices, logical decisions, and nonlinear physical or economic relationships in a single model. Here, the integer variables represent choices such as unit selection, on/off decisions, network topology, or sequencing, while the continuous variables capture operating levels, flows, temperatures, timings, or other physical quantities.

MINLP problems are typically solved using a combination of techniques drawn from nonlinear programming and integer programming, reflecting their hybrid structure. Broadly, the literature classifies solution methods into two main categories: decomposition-based methods and branching-based methods.

In decomposition-based methods, like Outer Approximation (OA) and Generalized

Benders Decomposition (GBD) , the problem is solved by splitting the problem into two parts, namely, a continuous nonlinear part and a discrete decision part, and then the optimization proceeds by alternating between them. OA works by decomposing the problem into a nonlinear subproblem and a linear master problem. At each iteration, the discrete variables are fixed, and the resulting Nonlinear Programming (NLP) problem is solved to obtain optimal continuous variables and the local information about nonlinear constraints. These nonlinear constraints are then linearized, thus producing linear constraints that approximate the feasible region. This becomes the new MILP problem, which is solved, and the resulting discrete solution is fed back in, and the process repeats until convergence to an optimal solution. On the other hand, GBD solves the problem by trying different discrete choices and then solving the corresponding continuous problem. This process provides a bound on the objective value and helps determine the cost associated with each choice. These cost estimates are then used to define cuts (or rules), which guide the optimizer toward better decisions.

In branching methods, Branch-and-bound (BnB) is a widely used technique for solving optimization problems. It works by systematically dividing the original problem into smaller subproblems, a process known as branching, and solving a relaxed version of each subproblem in which some constraints, such as integrality are temporarily ignored. This relaxation helps obtain a bound on the best possible solution within that region. Although the relaxed solution may not be feasible for the original problem, it provides an optimistic estimate, that is, a lower bound in the case of minimization problems. These bounds are then compared with the best feasible solution found so far. If a subproblem cannot yield a better solution, even in this relaxed form, it is discarded or pruned, and the algorithm focuses on more promising branches. If the bound indicates potential improvement, the branch is explored further. By repeatedly branching, solving relaxations, and pruning unpromising regions, the method efficiently explores only the most relevant parts of the solution space until the optimal solution is reached. Another branching method is Branch-and-Cut (BnC),

which extends the BnB method by adding extra constraints called cuts that eliminate unrealistic and suboptimal solutions. It begins by solving a relaxed version of the problem, then introduces cuts to remove fractional or infeasible solutions without excluding valid ones. After tightening the problem this way, it applies branching to explore smaller subproblems. By continuously refining the solution space and pruning poor options, BnC reaches the optimal solution faster than basic BnB.

In addition to these core approaches, modern MINLP solvers increasingly emphasize reformulation and structure exploitation. One such technique is Perspective reformulation. This technique is used in optimization problems where binary variables control continuous variables or decisions. It ensures that when a binary variable is inactive, the associated continuous variables behave consistently, that is ideally it would also be set to zero or related inactive states, this leads to a more accurate representation of the problem and helps in solver performance. Another method is convexification, which is generally applied to nonconvex functions, and it basically creates convex approximations of the same using linear bounds or convex envelops. These approximations do not exactly represent the problem, but it provides a tractable version that is easier to solve and yields useful bounds on the optimal solution. This method is generally used alongside iterative frameworks like branch and bound. General Disjunctive Programming (GDP) on the other hand, are used in problems where a decision involves choosing between different system configurations or sets of constraints, which are basically ”either-or” decisions. GDP converts these problems into logical statements, and they are converted into mathematical constraints, so that optimization solvers can work with them. Here GDP is not solving the problem; rather, it is a paradigm to formulate complex decision-making problems before they can be solved using standard optimization algorithms.

Another key format to represent MINLP is by converting the mathematical formulations to expression graphs. Expression graphs or factor graphs represent nonlinear functions as directed acyclic graphs composed of elementary operations, variables, and constants, en-

abling a factorable decomposition of complex expressions into simpler components. This representation allows optimization algorithms to systematically analyze structural properties such as convexity, separability, and differentiability. It supports efficient computation of derivatives through automatic differentiation and facilitates convex relaxation of global optimization. This way, expression graphs help transform complex mathematical models into a form that is easier to analyze and optimize.

MINLP has been extensively studied, with a wide range of solution techniques developed to handle problems involving continuous and discrete variables. However, there exists another important class of problems that extends beyond the traditional MINLP formulation, referred to as Mixed Combinatorial Nonlinear Programming (MCNLP) problems[18].

1.3 Mixed Combinatorial Nonlinear Optimization

In this thesis, we focus on the problems with the co-presence of both continuous and combinatorial variables, which are broadly categorized as MCNLP problems. In MCNLP engineering design problems, decision variables may include continuous parameters (e.g., geometry), integer variables (e.g., counts of features such as fins or holes), and selections from finite sets of combinations. Such combinations typically arise from component or material choices, where each option is characterized by multiple attributes, such as motor mass, voltage, and power ratings, or material density and elasticity. When these choices are independent, they can be treated as separate combinatorial sets, whereas dependencies or conflicts among components require defining a unified set of valid combinations.

We will discuss few of the problems to understand the mixed discrete-continuous aspect of the problem. For example, lets consider a mobile robot navigating an environment, in this case, it must decide the sequence of actions to execute, which is a discrete combinatorial problem, while also determining how to execute each action in terms of continuous variables like velocity, timing and control inputs. In the context of robotic planning,

Fernández-González et al. [7] formalize this as a hybrid planning problem. Similarly, consider the design of a spacecraft-based tether-net system for capturing space debris. Here, the system must select from a finite set of hardware and structural configurations, such as thruster types, net materials, and mesh resolutions which together form a discrete combinatorial design space. At the same time, it must determine how the system operates through continuous variables such as control inputs, geometric parameters, and trajectory-related quantities. Therefore, we can formulate a design optimization, where overall system performance metrics like fuel consumption jointly depend on these discrete configurations and continuous design variables. This naturally leads to a mixed-combinatorial optimization problem, where the objective is defined over both a finite set of valid design combinations and a continuous parameter space. Liu et al[18] abstract this as a graph problem and learn the hybrid space using a Graph Neural Network (GNN) , which is embedded within a heuristic algorithm and used to optimize the system performance metrics. The resulting optimization framework forms the core foundation of this thesis, upon which we build more generalized abstractions and advanced learning-based formulations to further enhance the efficiency and structure of the optimization process.

This key distinction on how discrete decisions are represented distinguishes it from MINLP, where discrete variables are typically modeled as independent integer or binary variables, allowing them to be indexed and manipulated within a structured numerical space. In contrast, MCNLP problems involve selecting from a finite set of valid combinations, where each combination represents a structured grouping of interdependent design choices. These combinations cannot be decomposed into independent variables without losing their inherent relationships. As a result, the combinatorial space in MCNLP does not admit a natural Euclidean structure. Unlike integer variables, combinations cannot be ordered, interpolated, or traversed through local neighborhood operations. Instead, the search over this space is fundamentally a selection problem over a discrete set, often with complex dependencies and constraints between elements.

We see that unlike continuous or integer variables, combinations do not naturally support vector operations or ordered neighborhood relationships, which makes them difficult to handle using conventional gradient-based or metaheuristic optimization methods. Consequently, most existing approaches treat the combinatorial space as a selection problem or transform it into alternative representations that are more amenable to optimization, such as one-hot or ordinal encoding with integers [20, 28, 24].

1.4 Abstractions and Limitations

MCNLP and MINLP problems are computationally NP-Hard [22], and the increased choices among discrete variables lead to combinatorial explosion as we had discussed earlier on[17]. Therefore, the mixed combinatorial search space is generally intractable, which necessitates abstractions to represent it. Thus, reformulation and structure exploitation become central perspectives in solving optimization problems, typically via subsets, projections, or latent representations. For example, Pattern Databases, a hierarchical method, abstracts combinatorial spaces by reducing them to simpler subproblems, providing tight lower bounds that guide and accelerate search cite patterndatabase. Combinatorial Optimization with Policy Adaptation using Latent Space Search (COMPASS) abstracts combinatorial space by mapping it to a low-dimensional continuous latent representation of policies, within which effective strategies are searched[3]. MAXQ Value Function Decomposition (MAXQ) abstracts by using only subtask-relevant variables under safety guarantees, reducing the complexity [6]. Combinatorial Bayesian Optimization (COMBO) on the other hand, abstracts the combinatorial space by representing configurations as nodes in a graph and models a surrogate function over this using Laplacian-based kernels, exploiting local transitions between configurations[23]. While these methods expose techniques to reduce the complexity of the search space, they lack a structured notion of directionality within the combinatorial space, and thus are limited in providing navigational guidance during

optimization.

Several existing research studies aim to improve navigation or provide gradient-like guidance for effective search in the combinatorial space. The Gumbel-Softmax relaxation [11] technique introduces gradient-based awareness into combinatorial exploration, but it suffers from a discretization gap and scalability bottlenecks. Generative Flow Ant Colony Sampler (GFACS) is a technique where combinatorial space is abstracted as a sequential decision process over a directed acyclic graph (DAG); here, Generative Flow Networks (GFlowNets) learns a reward distribution over solutions using Graph Neural Network (GNN) internally. This learned prior further guides ant-based search toward high-quality solutions [13]. Despite the significant progress, the existing methods operate at the level of state compression or scalar function approximation. Most existing methods assume a static combinatorial objective, whereas in MCNLP settings the landscape itself varies with the continuous context, requiring a representation that adapts across conditions. These limitations point to a critical gap: there is a lack of a unified, geometry-aware abstraction and learning methodology that can learn transitions in the objective landscape, especially in the mixed combinatorial domain.

1.5 Motivation

A recent approach for solving such MCNLP problems in design automation proposes a decomposition that separates the CO part from the problem and solves the CO problem with an GNN recommender for combination selection. This concept draws partly on work in multi-robot task allocation [25, 26] and network reconfiguration [9], where large combinatorial spaces are effectively represented as graphs and efficiently searched over by GNNs trained on data or experience. Building on this concept, our recent work [18] presents **GNN-ReCo**, an optimization framework that divides the problem into a combination learning stage—where configurations are represented as graph nodes and processed by a GNN-

based recommender—and a non-CO optimization stage, which is solved using standard NLP or MINLP methods.

Liu et al. [18] abstract the combinatorial search space as a graph and propose a GNN-based recommendation system, named as **GNN-ReCo**, to embed in the unconstrained optimization framework, which efficiently selects combinatorial variables corresponding to a given continuous variable vector.

This decomposition addresses a fundamental challenge in mixed combinatorial optimization: establishing meaningful relational structure among combinatorial selections. Classical MINLP approaches such as branch-and-bound often rely on indexification, where categorical or combinatorial variables are represented through indexed numerical encodings that implicitly assume ordinal or neighborhood relationships between choices. While this may be suitable for low-dimensional categorical variables, it becomes inadequate for MCNLP problems, where each combinatorial decision corresponds to an entire configuration vector rather than an isolated categorical label. In such settings, indexification introduces artificial relationships between combinations that do not reflect the underlying structure of the optimization landscape. In contrast, GNN-Recommender For Combinatorial Optimization (GNN-Reco) naturally formulates the combinatorial selection process as a recommendation problem conditioned on the continuous optimization context. By embedding combinatorial configurations into a learned latent graph space, the framework captures relational and structural dependencies among combinations that conventional index-based representations cannot. This results in a richer, more expressive representation space that supports ordering, retrieval, and recommendation across large combinatorial domains, thereby motivating the decomposition of the optimization problem and the use of graph neural recommendation models.

However, the edge construction in GNN-Reco primarily connects combinatorial configurations based on structural similarity measures, such as Euclidean distance, within the combinatorial embedding space. As a result, the learned relationships are largely restricted

to the combinatorial domain itself and do not explicitly encode how the objective function changes across configurations within a given continuous context. In the present work, we extend this formulation by augmenting graph edges with signed pairwise objective differences between configurations conditioned on the continuous variables. Consequently, the graph no longer represents only combinatorial proximity, but also captures directional information within the objective landscape itself. This induces a richer edge-flow representation over the graph that reflects optimization-relevant relationships between configurations. Furthermore, the HodgeRank interpretation provides a principled mathematical framework for analyzing these edge flows in terms of global consistency, local cyclic inconsistency, and reliability of the learned ranking structure.

Building on this optimization framework, we further draw inspiration from abstracting combinatorial spaces through directed graphs of pairwise relationships, where global structure is inferred from local edge-wise comparisons. This perspective, inspired by HodgeRank[12], serves as our novel formulation for building a learning-based model that reasons over combinatorial structures in a geometry-centric way and learns edge-wise differences in objectives between configurations indexed by continuous variables. We extend the GNN-Reco Optimization framework and learning edge-wise energy differences to the mixed combinatorial optimization setting.

Here, we explore an alternative recommendation-based approach to improve the scalability and interpretability of search across mixed combinatorial optimization spaces. By incorporating concepts from HodgeRank into the learning process, the proposed framework provides a structured mechanism for estimating global ranking structure from sparsely sampled regions of a much larger combinatorial domain. The central idea is to learn the underlying relationship between combinatorial configurations, continuous contextual variables, and the resulting objective landscape by analyzing pairwise differences in objectives. The main hypothesis behind this approach is the notion of explicit gradient awareness over combinatorial spaces: we posit that learning signed pairwise objective changes when transition-

ing between configurations enables more scalable retrieval while simultaneously making the recommendation process more structured, interpretable, and optimization-aware.

This new approach necessitates a new graph representation, in which edge weights encode relative differences in objective values between configurations. Through this construction, we define a structured mathematical object over the combinatorial graph that captures the local geometry of the optimization landscape. In practice, during each combinatorial choice selection step, the model is provided with a subset of candidate configurations, represented as nodes in an undirected graph, along with a continuous context proposed by the optimizer. The model then outputs a directed graph over the same nodes, where edges are assigned normalized objective differences. An illustrative visualization of this mapping concept is shown in Fig. 3.3. This enriched directed graph enables the optimizer to make an informed decision over the objective landscape, which is a mixed, combinatorial, hybrid search space. Consequently, this technique enables gradient-like navigation over this hybrid space while preserving the discrete structure and underlying geometry. This new GNN structure navigates the selection of combinations; we name this variation of the prior recommending system **GNN-NavCo** to distinguish it from prior work.

1) ***Abstraction of Combinatorial Spaces on Directed Graph:*** We represent the searchable combinatorial space as a directed graph, where edges encode *pairwise energy differences* between configurations. These values capture how the objective and penalty-augmented constraints evolve across transitions.

2) ***Theoretical Foundation using Combinatorial Hodge Theory:*** We develop a mathematical framework for the proposed graph-based representation by interpreting the abstraction in terms of combinatorial Hodge theory. This edge flow function encodes pairwise energy differences between configurations and induces a directional structure that provides gradient-like guidance for search. The formulation is grounded in combinatorial Hodge theory, enabling the analysis of consistency and recoverability of the underlying scalar energy.

3) ***Gradient-aware Learning of Combinatorial Navigation via GNN:*** We develop a learning framework called **GNN-NavCo** that computes the gradient-like edge weights of the graph and predicts the corresponding *energy* landscape for a given continuous variable vector of a candidate design. The predicted landscape is then used to guide the recommendation of promising nodes, enabling efficient exploration of the combinatorial space across different sampled subsets of configurations.

4) ***Evaluation of GNN-NavCo-aided optimization:*** We decompose the MCNLP solution process into a combination recommendation model generation stage and a non-combinatorial (in our problems, continuous) optimization stage as [18]. We then conduct an extensive evaluation of the proposed framework on two unconstrained benchmark problems to examine its scalability and performance across large combinatorial spaces. Furthermore, the framework is evaluated on constrained benchmark problems to demonstrate its applicability to constrained optimization settings.

Chapter 2

Theoretical Foundations of Approach

In this section, we discuss the theoretical foundation for the thesis. We will first define the Mixed combinatorial non-linear programming (MCNLP), followed by our proposal on how to abstract the problem space using a graph-based representation of pairwise data. Then, combinatorial Hodge theory is briefly discussed, followed by a Hodge-theoretic interpretation of the proposed abstraction. These ideas would be used towards implementing the model architecture and learning process in Chapter 3.

2.1 MCNLP Optimization Problem

We consider a MCNLP with k_{comb} combinatorial variables z and k_{cont} continuous variables x with p inequality constraints and q equality constraints in the following form:

$$
\begin{aligned}
&\min_{\mathbf{Z}_{\text{comb}}, \mathbf{X}_{\text{cont}}} f(\mathbf{Z}_{\text{comb}}, \mathbf{X}_{\text{cont}}) \\
\text{subject to} \quad & g_i(\mathbf{Z}_{\text{comb}}, \mathbf{X}_{\text{cont}}) \leq 0, \quad i = 1, 2, \ldots, p \\
& h_j(\mathbf{Z}_{\text{comb}}, \mathbf{X}_{\text{cont}}) = 0, \quad j = 1, 2, \ldots, q \\
\text{where} \quad & \mathbf{Z}_{\text{comb}} = \begin{bmatrix} z_1 & z_2 & \ldots & z_{k_{\text{comb}}} \end{bmatrix} \\
& \mathbf{X}_{\text{cont}} = \begin{bmatrix} x_1 & x_2 & \ldots & x_{k_{\text{cont}}} \end{bmatrix} \\
& x_n^L \leq x_n \leq x_n^U, \quad \forall\, n = 1, 2, \ldots, k_{\text{cont}} \\
& \mathbf{Z}_{\text{comb}} \in \mathbb{Z}
\end{aligned}
\tag{2.1}
$$

Where $\mathbf{Z}_{\text{comb}}$ and $\mathbf{X}_{\text{cont}}$ are the vectors of the combinatorial and continuous variables, respectively. x^{L} and x^{U} are the lower and upper bounds of the continuous variables, and $\mathbb{Z}$ is the valid combination set.

The prior work of [18] decomposes the MCNLP into two parts: the learning of combinations which are abstracted as graph nodes, and solving a Non-CO as just NLP or MINLP. We will briefly discuss the framework to provide background on the problem. In this thesis, our focus is on the former part of the decomposition technique, that is, to define a new abstraction of the mixed domain and to formulate a new method to learn this abstracted space , which will be grounded in Discrete Exterior Calculus [5] and Combinatorial Hodge Theory [12]. We will show how our abstracted model can be explained using discrete geometry and how we exploit tools in this field to define the neural network architecture and a custom regularizer to stabilize learning.

2.2 Decomposing with GNN-Reco Framework

If the valid combination set is defined as $\mathbb{Z}$, then each combination can be represented by a feature vector $\mathbf{Z} = [z_1, z_2, \ldots, z_{k_{\text{cont}}}]$, where $\mathbf{Z} \in \mathbb{Z}$ and k_{cont} denotes the number of features. These features may correspond to the physical attributes associated with the selected components. For example, in robot design, such features may include motor mass, voltage, and power ratings, as well as sensor detection range and mass.

The work of [18] represents the combinatorial space as a fully connected undirected graph $\mathcal{G} = (\mathbf{Z}_{\text{comb}}, E)$, where each valid combination is abstracted as a node positioned according to its feature vector. The edge weight between any two nodes is defined as the Euclidean distance between the corresponding feature vectors of the two combinations. This graph representation enables the learning of relational structure over the discrete design space.

The key idea is to decompose the problem defined in (4.1) and rewrite it as following Non-CO form:

$$
\begin{aligned}
&\min_{\mathbf{X}_{\text{cont}}} f(\mathbf{Z}_i^*, \mathbf{X}_{\text{cont}}) \\
\text{where} \quad &\mathbf{y} = \gamma(\mathcal{G}, \mathbf{X}_{\text{cont}}) \\
&i^* = \underset{i=1,2,\ldots,n_{\text{tot}}}{\operatorname{argmax}} \; y_i, \quad y_i \in \mathbf{y} \\
&\mathbf{X}_{\text{cont}} = \begin{bmatrix} x_1 & x_2 & \ldots & x_{k_{\text{cont}}} \end{bmatrix} \\
&x_n^L \leq x_n \leq x_n^U, \quad \forall\, n = 1, 2, \ldots, k_{\text{cont}} \\
&\mathbf{Z}_i^* \in \mathcal{G}, \text{ and } \mathcal{G} \in \mathbb{Z}
\end{aligned}
\tag{2.2}
$$

Here $\mathbf{y}$ is the score vector indicating the fitness value of each candidate combination in the abstracted graph $\mathcal{G}$, which is calculated by a function $\gamma(\cdot)$ involving the prediction of **GNN-ReCo**. i^* is the index of the node (candidate combination in $\mathcal{G}$) with the highest fitness score, denoted by $\mathbf{Z}_i^*$.

Trained by a specialized list-wise loss function, **GNN-ReCo** efficiently learns to predict

the *score* of each combination and then recommends the best-suited combination in the valid set for the given non-combinatorial variables. Here, the *score* can be a scalar value defined by the objective function value. By embedding it into the function evaluation, **GNN-ReCo** can work with sampling-based heuristic optimization algorithms as well as classic optimization solvers. In this thesis, we will be replacing **GNN-ReCo** with **GNN-NavCo**. This new, pluggable, and enhanced version will be embedded in the MCNLP decomposition framework.

In our work, GNN-NavCo handles both unconstrained and constrained problems. We adopt a common approach in constrained optimization, converting the constrained problem into an unconstrained one by introducing penalty functions that incorporate constraint violations into the objective [18]. Following this approach, we define a penalized energy function. We model the objective function landscape as an energy function [14], where both feasible and infeasible objective values are unified into a single objective or energy landscape, yielding a mixed combinatorial energy landscape on which optimization is performed. The energy landscape is shaped by objectives and constraints in equation (4.1).

$$\phi(\mathbf{Z}, \mathbf{X}) = f(\mathbf{Z}, \mathbf{X}) + \rho \left(\sum_{i=1}^{p} P_\tau(g_i(\mathbf{Z}, \mathbf{X})) + \sum_{j=1}^{q} P_\tau(h_j(\mathbf{Z}, \mathbf{X})) \right), \tag{2.3}$$

where $\rho > 0$ is a penalty weight and $P_\tau(\cdot)$ denotes penalty. It can be chosen based on the objective landscape.

2.3 Graph-Based Representation of Pairwise Data

The energy defined in (2.3) depends jointly on the discrete configuration variable z and the continuous parameter x. To model this interaction, we construct a graph-based abstraction over the mixed combinatorial space. Let us first consider $\mathbf{Z}_{comb}$, which defines our combinatorial space. We first create k number of nodes and connect each of them, constructing a fully connected or complete graph $\mathcal{G} = (\mathbf{Z}_{\text{comb}}, \mathcal{E})$, where $\mathcal{E} = \{(i, j) : i \neq j\}$. Now,

let's assume that there is a continuous $\mathbf{X}_{\text{cont}}$ variable, which can induce an energy at these nodes based on equation (2.3). Now, for a given $\mathbf{X}_{\text{cont}}$, we can then take the difference in energy between each of these nodes and move that to the edge; this is what we call pairwise differences in energy. Now, if we say there exists a function of energy at the nodes which is parametrized by $\mathbf{X}_{\text{cont}}$, then surely the pairwise differences value on the edge will also be parametrized for the same $\mathbf{X}_{\text{cont}}$. Specifically, for each pair of configurations $(\mathbf{Z}_i, \mathcal{Z}_j)$, we define an edge function $\omega(i, j, \mathbf{X}_{\text{cont}})$, which captures the relative change in energy between configurations under context x.

$$\omega(i, j, \mathbf{X}_{\text{cont}}) = \phi(z_j, \mathbf{X}_{\text{cont}}) - \phi(z_i, \mathbf{X}_{\text{cont}}) \tag{2.4}$$

This construction induces a directed edge structure over the graph, where edge weights encode transition-wise energy differences. Importantly, while the graph topology remains fixed, the edge weights vary continuously, thereby altering the relative ordering and direction of improvement between configurations. Furthermore, this edge quantity satisfies the antisymmetry property, $\omega(i, j, \mathbf{X}_{\text{cont}}) = -\omega(j, i, \mathbf{X}_{\text{cont}})$, reflecting that reversing a transition negates the corresponding energy difference. Therefore, the abstraction can be thought of as having fixed nodes representing the combinatorial variables, and edges between those nodes representing a function that encodes the pairwise energy difference between the respective nodes for the indexed continuous variable X_{cont}. This, when seen from the matrix perspective, is having n-number of matrices of graph $\mathcal{G}_{\text{II}} = (Z_{comb}, \mathcal{E}_{\text{II}})$, with Z_{comb} fixed and $q \in 1, 2, 3....n$. In this case, E_q represents the adjacency matrix, which encodes the pairwise differences between the respective nodes. Therefore, we use this central abstraction as a target for a neural network to learn. This edge function, which represents the pairwise energy differences, can be thought of as a dynamic edge flow on graphs and thus is similar to the formulation of HodgeRank pioneered by Jiang et al[12]. In this thesis, the proposed abstraction will be analyzed within this context.

2.4 Combinatorial Hodge Theory and HodgeRank

HodgeRank, proposed by Jiang et al. [12], is a pioneering work in the field of statistical ranking on graphs and the abstraction of discrete data. It specifically handles cardinal scores to rank preferences, unlike the ordinal scores approach seen before. This is of particular interest to this thesis because the pairwise differences in the energy space can be interpreted as the cardinal score of preference. HodgeRank is a framework or set of methods for analyzing pairwise data represented as edge flows on graphs using discrete or combinatorial Hodge theory.

HodgeRank is a graph-based ranking framework derived from combinatorial Hodge theory that converts pairwise comparisons into a globally consistent ranking while also measuring data inconsistencies. Instead of assuming that all preferences can be perfectly ordered, HodgeRank models rankings as flows on a graph, where items are vertices and pairwise preferences are edges. Using a Hodge decomposition, the method separates ranking information into three components: a gradient flow representing the optimal global ranking, a curl flow capturing local inconsistencies, and a harmonic flow representing global cyclic inconsistencies. This allows HodgeRank not only to produce a ranking but also to quantify its reliability. One of its major advantages is that the global ranking can be computed efficiently through linear least-squares optimization rather than NP-hard combinatorial optimization methods such as Kemeny ranking. HodgeRank is particularly useful for modern datasets that are incomplete, imbalanced, noisy, or based on pairwise comparisons, such as those used in recommendation systems, voting systems, web ranking, and preference aggregation.

A pairwise comparison graph is a weighted directed graph $G = (V, E)$ in which each vertex $i \in V$ represents an alternative and each edge $(i, j) \in E$ encodes a preference comparison between alternatives i and j. The observed comparisons are represented by a skew-symmetric matrix $Y \in \mathbb{R}^{n \times n}$ where $Y_{ij} = -Y_{ji}$ and Y_{ij} measures the preference

intensity of alternative j over i.

A pairwise ranking flow is then defined as a skew-symmetric edge flow on this graph, assigning a numerical value to every oriented edge that captures the direction and magnitude of preference. The global ranking model assumes the existence of a score function.

$$s : V \to \mathbb{R} \tag{2.5}$$

such that the induced ranking flow satisfies

$$X_{ij} = s_j - s_i \tag{2.6}$$

where $X = [X_{ij}]$ is called a gradient flow or globally consistent ranking flow. Thus, X belongs to the model space

$$M_G = \left\{ X \in \mathbb{R}^{n \times n} \;\middle|\; X_{ij} = s_j - s_i,\ s : V \to \mathbb{R} \right\} \tag{2.7}$$

which consists of all rank-2 skew-symmetric matrices generated by score differences. According to the Hodge-theoretic framework, an edge flow X is globally consistent if for every triple of alternatives (i, j, k), the additive consistency condition

$$X_{ij} + X_{jk} + X_{ki} = 0 \tag{2.8}$$

holds, implying that preferences can be explained entirely by a single global potential function s. This characterization provides the mathematical foundation for distinguishing coherent rankings from cyclic inconsistencies in pairwise comparison data.

HodgeRank introduces the combinatorial gradient operator, which maps a scalar potential or score function $s : V \to \mathbb{R}$ defined on the vertices of a graph to an edge flow

representing pairwise rankings. Mathematically, the gradient flow is defined as

$$(\operatorname{grad} s)(i,j) = s_j - s_i \tag{2.9}$$

meaning that the preference between alternatives i and j is determined entirely by the difference in their global scores. Any edge flow X that can be expressed in this form is called a gradient flow and represents a globally consistent ranking because all pairwise comparisons arise from a single underlying potential function.

The collection of all such globally consistent flows forms the model space

$$M_G = \{\operatorname{grad}(s) \mid s : V \to \mathbb{R}\} = \operatorname{im}(\operatorname{grad}) \tag{2.10}$$

which is the image of the gradient operator. The statistical ranking problem is then formulated as a weighted least-squares optimization that projects the observed pairwise comparison flow Y onto the subspace M_G of gradient flows:

$$\min_{X \in M_G} \|X - Y\|_{2,w}^2 = \min_{X \in M_G} \sum_{(i,j) \in E} w_{ij}(X_{ij} - Y_{ij})^2 \tag{2.11}$$

Here, the weight matrix $W = [w_{ij}]$ assigns importance to each comparison edge and may encode factors such as confidence, frequency, or reliability of voter judgments. The minimizer therefore corresponds to the closest globally consistent ranking flow to the observed data, providing a principled graph-theoretic interpretation of ranking as an ℓ_2-projection onto the space of gradient flows.

After obtaining a global ranking through the combinatorial gradient framework, the next step is to quantify the degree to which the observed pairwise rankings deviate from global consistency. This is achieved through the notion of combinatorial curl, which serves as a measure of local inconsistency on triangular loops within the comparison graph. Consider a graph $(G = (V, E)$ whose vertices represent alternatives and whose edges encode

pairwise comparisons through an edge flow X. In a perfectly consistent ranking system, traversing a closed loop of comparisons should yield zero net preference change, meaning that the accumulated preference differences around the loop cancel out. Since the smallest nontrivial cycle in a graph is a triangle, local inconsistency is naturally measured over triples of alternatives.

To formalize this, the framework introduces a triangular flow, a function

$$\Phi : V \times V \times V \to \mathbb{R} \tag{2.12}$$

that behaves as an alternating 3-tensor, satisfying antisymmetry properties such as

$$\Phi(i,j,k) = \Phi(j,k,i) = \Phi(k,i,j) = -\Phi(j,i,k) = -\Phi(i,k,j) = -\Phi(k,j,i) \tag{2.13}$$

A triangular flow therefore captures triplewise ranking interactions in the same way an edge flow captures pairwise preferences. The combinatorial curl operator maps an edge flow X to a triangular flow and is defined on every triangle (i,j,k) in the graph by

$$(\operatorname{curl} X)(i,j,k) = X_{ij} + X_{jk} + X_{ki} \tag{2.14}$$

whenever all three edges belong to the graph, and zero otherwise. Intuitively, this quantity measures the failure of pairwise preferences to satisfy transitive consistency on the triangle. For example, if

$$i \succ j, \qquad j \succ k, \qquad k \succ i$$

then the sum around the cycle is nonzero, indicating a cyclic contradiction in preferences. The skew-symmetry condition

$$X_{ij} = -X_{ji}$$

guarantees that the curl itself forms a valid triangular flow with the required antisymmetric structure. Hence, the combinatorial curl provides a rigorous mathematical statistic for detecting and quantifying local inconsistencies in ranking data. While the gradient component represents the globally consistent portion of the ranking, the curl component isolates cyclic preference structures that obstruct the existence of a coherent global ordering. This decomposition is central to Hodge-theoretic ranking, as it distinguishes reliable ranking information from locally contradictory voting behavior.

HodgeRank also provides another important tool that can help in situations where ranking data is sparse or incomplete. Hodge-theoretic ranking aggregates individual voter preferences into a more complete average pairwise ranking matrix Y. This averaging process transforms partial rankings into statistically meaningful pairwise comparisons between alternatives. Several averaging schemes can be used, depending on the nature of the data and the desired invariance properties. The arithmetic mean of score differences computes the average score difference and is translation-invariant. Alternatively, when scores are positive, the geometric mean of score ratios uses logarithmic differences, providing scale invariance. In ordinal ranking settings, binary comparisons define pairwise preferences using the sign function, producing a probability difference that is invariant under monotone transformations. Together, these averaging methods provide flexible mechanisms for converting incomplete voter data into robust pairwise ranking flows suitable for global ranking and inconsistency analysis.

2.5 Hodge Theoretic Interpretation

In the general HodgeRank framework, the ranking problem is modeled as a graph in which the alternatives to be ranked are represented as vertices, and the comparisons between them form the graph's edges. The notion of "voters" arises naturally in recommendation systems and preference aggregation settings, where different users independently assign car-

dinal scores or ratings to items such as movies, products, or websites. For example, in a movie recommendation system like Netflix, each viewer acts as a voter by rating a subset of movies, and these ratings are transformed into pairwise comparisons that express how strongly one movie is preferred over another. Since different users rate different subsets of items, the resulting data is often incomplete and imbalanced, motivating a graph-based formulation. HodgeRank aggregates all these pairwise preferences into an edge flow on the graph and then uses a least squares approximation to recover a global ranking. Specifically, the method finds a global score function whose induced pairwise differences best match the observed voter comparisons while minimizing the total squared error. In this way, the least squares formulation provides the optimal global ranking that most consistently explains the collective preferences expressed by the voters.

In the present work, the combinatorial state space naturally admits an interpretation within the HodgeRank framework. The combinatorial configurations $\mathbf{Z}_{\mathrm{comb}}$ corresponds to the alternatives or items to be ranked, while each continuous context vector $\mathbf{X}_{\mathrm{cont}}$ induces a context-dependent ranking flow over these configurations akin to a "voter". For a fixed context $\mathbf{X}_q$, the energy function $\phi(Z, \mathbf{X}_q)$ assigns a scalar score to each combinatorial configuration Z. The induced pairwise interaction $\omega(i, j, \mathbf{X}_q)$ therefore defines a context-conditioned edge flow that measures the relative preference or energy difference between configurations $\mathbf{Z}_i$ and $\mathbf{Z}_j$. In particular, when $\omega(i, j, \mathbf{X}_q) = \phi(\mathbf{Z}_j, \mathbf{X}_q) - \phi(\mathbf{Z}_i, \mathbf{X}_q)$ the edge flow lies in the image of the combinatorial gradient operator and is therefore globally consistent. This establishes a direct correspondence between the proposed energy-based formulation in Eq.(2.3) and the gradient flow structure of HodgeRank in Eq. (2.9). Consequently, the ranking problem may be viewed as recovering a globally consistent discrete energy over the combinatorial graph conditioned on the continuous context space. The framework also naturally accommodates sparsity. In practice, it is computationally infeasible to exhaustively sample the entire continuous context space $\mathbf{X}_{\mathrm{cont}}$ or enumerate all combinatorial configurations and their pairwise relations, especially if we want to use a

neural network or an approximator to learn this space. As a result, the observed ranking data consists only of sparse subgraph samples and partial pairwise comparisons. This aligns closely with the statistical setting of HodgeRank, where global rankings are estimated from incomplete edge observations. Let $G_q = (Z_q, \mathcal{E}_q)$ denote a sampled subgraph containing a subset of combinatorial states together with the associated adjacency structure and pairwise comparisons induced by context $\mathbf{X}_q$. The learning problem then becomes one of estimating the underlying globally consistent ranking flow from these sampled local observations. This interpretation also provides theoretical justification for subgraph sampling strategies such as those employed in GNN-Reco [18], since Hodge-theoretic recovery guarantees that consistent global structure can be estimated from sparse local comparisons. Within this formulation, the neural architecture may be interpreted as a learned operator that maps sampled combinatorial subgraphs and continuous contextual information to a skew-symmetric edge flow representing pairwise ranking relations:

$$(G_q, \mathbf{X}_q) \longrightarrow \omega_q$$

The network therefore learns the latent relationship between combinatorial structure, contextual conditioning, and pairwise energy differences. Therefore, the learning operator takes a subgraph or an undirected graph of combinatorial combinations and a continuous context X_{cont} and outputs a prediction as a skew-symmetric matrix representing the edge-wise differences between all the nodes for that continuous context. It needs to be noted that although the learning space is sparse in nature, the learning operator learns to estimate all the edges for the nodes it gets during the inference operation. In cases where data is sparse, we have earlier seen in HodgeRank that it uses the averaging method to estimate these missing values. Similarly, it can be seen that the learning operator does a similar function of estimating the edge values, but from the intricate latent structure over the mixed combinatorial space. Once the edge flow is predicted, a least-squares Hodge projection may

be used to recover the globally consistent gradient component corresponding to the discrete potential function over the graph. Furthermore, the combinatorial curl operator provides a principled measure of local inconsistency:

$$(\operatorname{curl}\omega)(i,j,k) = \omega_{ij} + \omega_{jk} + \omega_{ki}$$

If the curl vanishes on all triangular loops, the learned flow is globally integrable and corresponds exactly to a discrete potential field. Nonzero curl values instead quantify cyclic inconsistencies and may therefore be interpreted as a certificate of unreliability or local contradiction within the learned ranking structure.

This suggests a natural regularization strategy in which the training objective penalizes the magnitude of the combinatorial curl through a loss function of the form

$$\mathcal{L} = \mathcal{L}_{\text{data}} + \lambda \|\operatorname{curl}(\omega)\|^2$$

The curl regularization term suppresses cyclic inconsistencies in the predicted pairwise ranking flow, thereby encouraging the learned edge flow to remain close to the gradient subspace of globally consistent rankings. Consequently, the network is biased toward learning integrable discrete potential structures whose pairwise differences admit a coherent global ranking interpretation. Consequently, the proposed framework extends HodgeRank from static statistical ranking to a context-conditioned learning setting in which globally consistent ranking flows are learned directly from sparse combinatorial observations and continuous contextual variables.

Chapter 3

GNN NavCo: Learning and Optimization Framework

3.1 Introduction

This chapter will define the learning framework we have developed. We will go through the architecture, the loss functions, and the regularization functions in detail. Then we will explain the optimization framework and how we embed the learned model into it. Post that, we will work on benchmark problems and discuss the results of the problem.

3.2 Learning Pairwise Data on Configuration Graph

Given the configuration graph $\mathcal{G}$ constructed from sampled configurations $\mathbf{Z}_{\text{comb}}$, we introduce an Edge Flow Graph Network (EFGN) Architecture, a model that learns a parametric approximation of the edge flow introduced in Chapter 2. For a given context X_{cont}, the model predicts $\hat{\omega}_\theta(i, j; x)$ as an approximation of the true energy differences Eq(2.4). The proposed model adopts an encoder–decoder architecture.A graph neural network (GNN) encoder first processes the configuration graph to produce context-conditioned node embeddings that capture the relational structure among configurations. These embeddings are

then passed to a pairwise edge decoder, implemented as a multilayer perceptron (MLP), which maps pairs of node embeddings to predicted energy differentials. This design enables the model to learn a context-dependent differential representation of the energy landscape, where node-level representations encode global structure, and the decoder recovers directional relationships between configurations. The following subsections describe the encoder and decoder components in detail.

3.3 GNN Encoder

Before the raw context is passed onto the GNN, each configuration is augmented with a shared continuous context vector. This ensures that all node representations are explicitly conditioned on the continuous variables. This step creates the $v_i = [z_i \parallel x]$ for each configuration. These concatenated features are first projected into a latent space $h_i^{(0)} = \sigma(W_{in} v_i + b_{in})$, where W_{in} is a learnable projection matrix and $\sigma(\cdot)$ denotes a nonlinear activation function and b_{in} is the bias vector. A stack of graph message-passing layers is applied, allowing each node to aggregate information from all other configurations and encode its relative position within the candidate set. This can be expressed as :

$$H^{(k+1)} = \sigma\left(\hat{A} H^{(k)} W^{(k)} + b^{(k)}\right) + H^{(k)}$$

where $H^{(k+1)}$ is node embeddings at layer k+1 , $\hat{A}$ is the row normalized adjacency matrix and $+H^{(k)}$ denotes the residual connection. After L layers, the resulting node embeddings are denoted by h_i. The node embeddings parameterize the discrete differential field over the configuration graph.

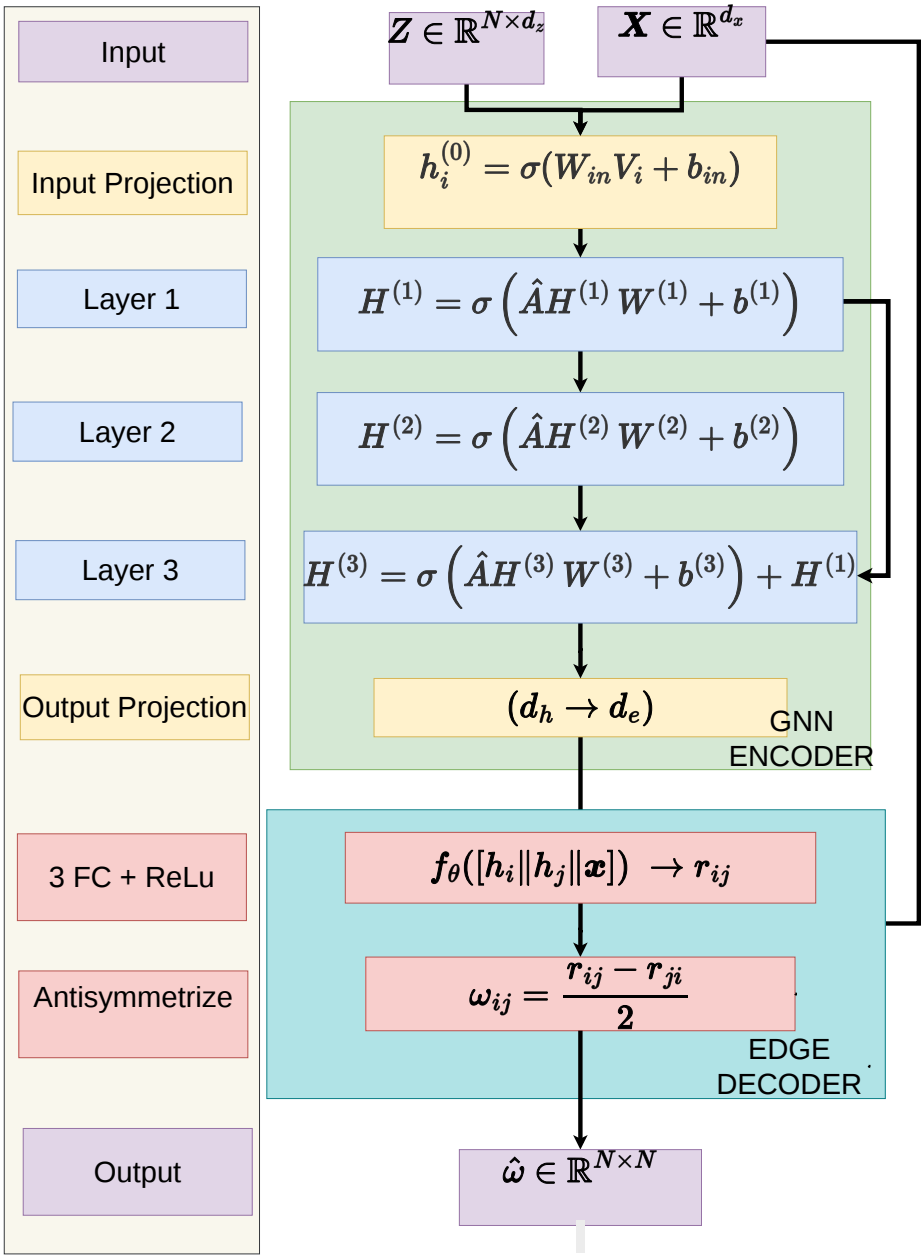

Figure 3.1: EFGN Model Architecture

3.4 Edge Decoder

The node embeddings produced by the GNN encoder are then passed to the edge decoder, a multilayer perceptron (MLP) that predicts the raw transition score$r_{ij} = f_\theta(h_i \parallel h_j \parallel x)$ where $f_\theta(\cdot)$ is a multilayer perceptron. Although the continuous context x is already encoded implicitly within each node embedding h_i through the GNN encoder. However, it is explicitly provided as input to the MLP, giving the edge decoder a direct, unattenuated copy of the context; this is analogous to a residual connection, compensating for any dilution of context information that may occur during message passing. To enforce antisymmetry of pairwise differences, the final edge prediction is defined as the average of the difference between ω_{ij} and ω_{ji} The resulting matrix $\Omega = [\omega_{ij}]$ is skew-symmetric, ensuring that predicted edge values form a valid discrete 1-form over the configuration graph[5].

3.5 Pairwise Differential Regression

The primary supervision signal is derived from pairwise energy differences between combinatorial choices[27]. For a fixed context X_{cont}, the true edge flow function associates each edge (i, j) of the configuration graph with $\omega_x(i, j)$ Eq. (2.4), which represents the change in the penalized objective when transitioning from configuration Z_i to Z_j. These values provide direct observations of the local variation of the energy landscape and therefore serve as the training targets.

Let $\hat{\omega}_\theta(i, j; x)$ denote the predicted edge difference produced by the model. Learning proceeds by minimizing the discrepancy between predicted and observed differences over the edges of the graph:

$$L_{\text{edge}} = \mathbb{E}_x \left[\frac{1}{|E(G)|} \sum_{(i,j)\in E(G)} \ell\big(\hat{\omega}_\theta(i, j; x) - \omega_x(i, j)\big) \right],$$

where $\ell(\cdot)$ denotes a regression loss such as the mean squared error or Huber loss.

Each objective evaluation contributes supervision for multiple configuration pairs[27]. For a sampled configuration set, pairwise differences generate a dense set of training signals that capture both the ordering and magnitude of energy changes between configurations. As a result, the model learns a structured approximation of how the objective varies across the configuration space.

3.6 HodgeRank-Based Consistency Regularization

While the pairwise differential regression loss supervises the learned pairwise ranking quantities using observed edge differences, it does not guarantee that the predicted edge flow corresponds to a globally consistent ranking in the HodgeRank sense. In the graph-based HodgeRank framework, a globally consistent ranking is represented by a gradient flow induced by a scalar score function $s : V \to \mathbb{R}$, where the pairwise ranking between two nodes is given by

$$X_{ij} = s_j - s_i.$$

Such flows belong to the gradient subspace and correspond to rankings that can be explained by a single global potential over the graph. However, a learned pairwise ranking flow $\hat{\omega}_\theta$ may contain cyclic inconsistencies that prevent it from being represented by a globally consistent ranking function.

HodgeRank characterizes these local inconsistencies using the combinatorial curl operator. For any triangular cycle (i, j, k) in the comparison graph, the curl of the learned edge flow is defined as

$$(\operatorname{curl} \hat{\omega}_\theta)(i, j, k) = \hat{\omega}_\theta(i, j) + \hat{\omega}_\theta(j, k) + \hat{\omega}_\theta(k, i).$$

If the edge flow is globally consistent, then the accumulated preference difference around

every closed triangular loop must vanish:

$$\hat{\omega}_\theta(i,j) + \hat{\omega}_\theta(j,k) + \hat{\omega}_\theta(k,i) = 0.$$

Therefore, nonzero curl values indicate the presence of local cyclic inconsistencies in the predicted ranking structure.

Overall Objective

The final training objective combines the differential regression loss with the structural regularization terms introduced above. The overall training objective is therefore defined as

$$L = L_{\text{edge}} + \lambda_{\text{cycle}} L_{\text{cycle}} \tag{3.1}$$

where λ_{cycle} control the contribution of the cycle consistency term.

3.7 Optimization Framework

Overview of the GNN-ReCo Optimization Framework

We adopt the GNN-ReCo optimization framework. This framework decomposes the design problem into continuous variables ($\mathbf{X}$) and discrete configuration choices ($\mathbf{Z}$), with objective $E(\mathbf{Z}, \mathbf{X})$. It alternates between exploring the continuous design space and selecting optimal configurations conditioned on each candidate design. At each iteration, a population-based optimizer—specifically Mixed Discrete Particle Swarm Optimization (MDPSO) [4]—generates candidate continuous vectors ($\mathbf{X}^{(t)}$). For each candidate, a neural model predicts the optimal configuration:

$$\mathbf{Z}^* = \arg\min_{\mathbf{Z}\in\mathbb{Z}} E(\mathbf{Z}, \mathbf{X}^{(t)}) \tag{3.2}$$

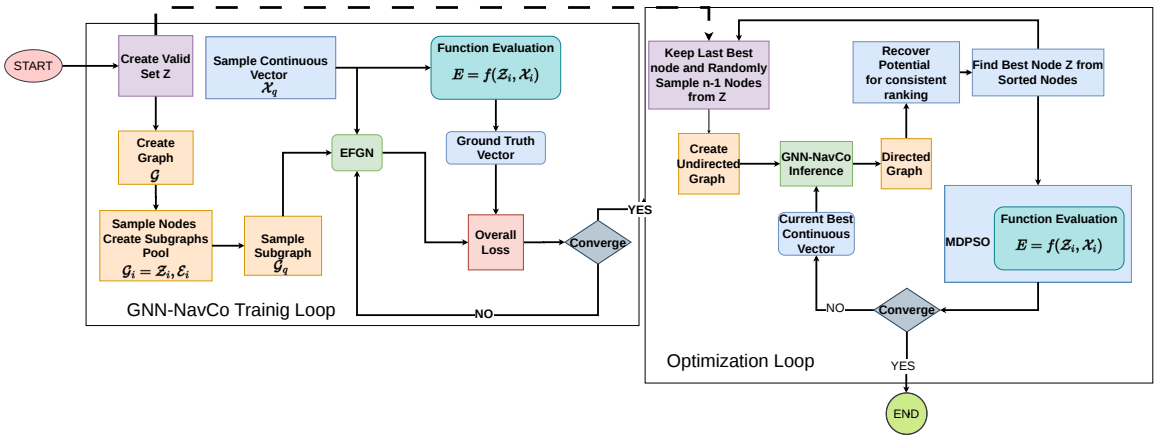

Figure 3.2: Overall Framework of GNN-NavCo Optimization

This approach lets the optimizer focus on continuous search while the model handles combinatorial selection. In the present work, we replace the GNN with the trained EFGN model, **GNN-NavCo**. This model predicts pairwise energy differences between configurations and provides a structured representation of the local energy landscape. The following subsection describes how this differential model applies to candidate configuration subgraphs during optimization.

Inference on Candidate Configuration Subgraphs

At each iteration, the continuous optimizer proposes a design vector $\mathbf{X}^{(t)}$, which serves as the continuous context $\mathbf{X}$ for GNN-NavCo.Then a candidate subgraph is constructed by sampling configurations from $\mathbf{Z}_{\text{comb}}$, with current best configuration Z^* included. The remaining nodes are sampled stochastically to ensure diversity and reduce sampling bias, consistent with the subgraph-based learning paradigm used in GNN-ReCo [18]. The resulting input is an *undirected fully connected graph* over sampled configurations and the continuous vector on $\mathbf{X}^{(t)}$. GNN-NavCo transforms this into a *directed fully connected graph*, where predictions form an antisymmetric matrix $Y \in \mathbb{R}^{N\times N}$ representing a learned 1-cochain (edge flow) over the candidate graph that encodes pairwise preferences between configurations. A globally consistent discrete gradient is subsequently recovered by pro-

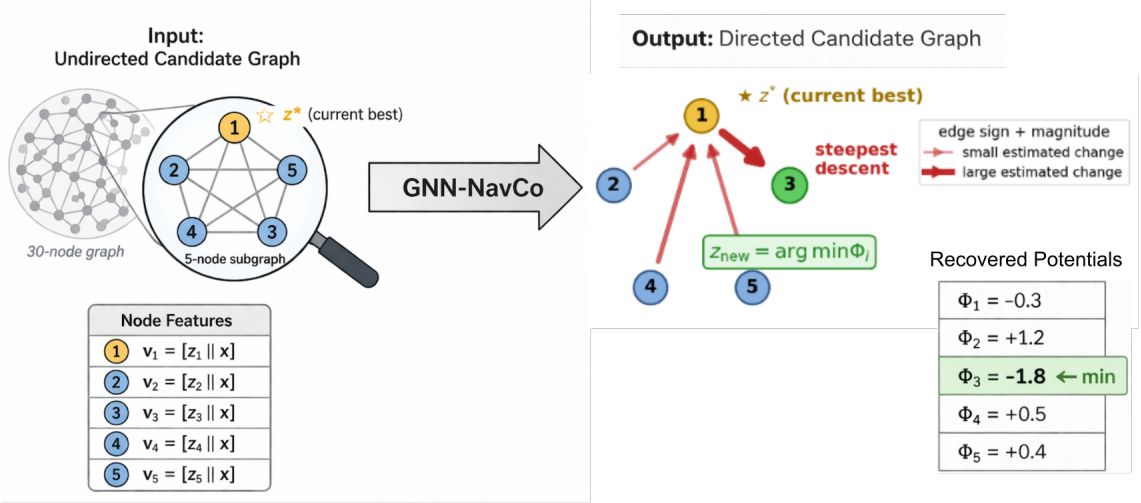

Figure 3.3: GNN NavCo Inference Process: Takes as input a subset of allowed combinations expressed as graph along with a candidate design vector, and produces as output a directed graph encoded with improvement direction and potential across the combinations.

jecting Y onto $\mathrm{im}(\partial_1^\top)$. This can be interpreted as assigning a directional "flow" over the graph, indicating which transitions are favorable under the current context.

From an intuitive standpoint, if we anchor ourselves at the current configuration z^*, the outgoing edges describe local directional preferences—i.e., which neighboring configurations appear better. However, this local view is inherently myopic: it does not account for how each candidate compares globally with all other configurations. To resolve this, we aggregate the predicted pairwise differences across the entire subgraph and recover scalar potentials for each node. This recovery step integrates the distributed edge information into a globally consistent ranking, enabling selection of the next configuration. Importantly, because the graph is fully connected and includes $\mathbf{Z}^*$, this global aggregation remains consistent with the local decision perspective at $\mathbf{Z}^*$, while correcting for inconsistencies in the predicted field. As a result, the method effectively provides a *globally informed discrete update direction*, giving the optimizer a gradient-like signal over the combinatorial space without violating its discrete structure. For a fully connected candidate graph, this reduces to Φ_i. The recovered potentials provide a scalar ranking of candidate configurations, and the configuration with the minimum potential is selected. In practice, potential recovery

reduces to a simple averaging operation.

$$\Phi_i = \frac{1}{N} \sum_{j=1}^{N} Y_{ij}$$

$$\mathbf{Z}^* = \arg\min_i \Phi_i$$

This procedure enables efficient evaluation of candidate configuration subsets while preserving the relative structure of the energy landscape predicted by the differential model. In PyTorch, this can be implemented as $\Phi = \text{mean}(Y, \text{dim} = 1)$ which aggregates the predicted pairwise energy differences for each configuration.

3.8 GNN-aided MDPSO Algorithm

Initialize the particle population $\mathbf{X}_1$ to $\mathbf{X}_{N_{\text{pop}}}$
Initialize the optimal node $\mathbf{Z}_1^*$ to $\mathbf{Z}_{N_{\text{pop}}}^*$
Function evaluation $f(\mathbf{Z}_i^*, \mathbf{X}_i)$, $i = 1$ to N_{pop}
for $t = 1$ to $max\_iteration$ **do**
    Update particles $\mathbf{X}_1$ to $\mathbf{X}_{N_{\text{pop}}}$ with MDPSO
    for $q = 1$ to N_{pop} **do**
        Select $n_{\text{sn}} - 1$ nodes randomly around each $\mathbf{Z}_q^*$ from $\mathcal{G}$
        Create $\mathcal{G}_q$
        run $\mathcal{G}_q^* = \text{GNN-NavCo}(\mathcal{G}_q, \mathbf{X}_q)$
        # *recover node score from adjacency matrix*
        $\Phi_i = \frac{1}{n_{\text{sn}}} \sum_{j=1}^{n_{\text{sn}}} \mathcal{E}_{ij}^*$
        Set $\mathbf{Z}_q^* = \arg\min_i \Phi_i$
    end for
    Function evaluation $f(\mathbf{Z}_i^*, \mathbf{X}_i)$, $i = 1$ to N_{pop}

if Converge **then**

Return the result

end if

end for

Return the result

3.9 Evaluation on Benchmark Problems

We design a set of case studies to evaluate the GNN-NavCo framework on benchmark problems derived from MINLPLib [10]. The studies aim to: **1)** assess scalability with respect to the size of the combinatorial set; **2)** evaluate scalability with increasing problem dimensionality; and **3)** examine applicability to both unconstrained and constrained problems.

Three minimization problems were selected from MINLPLib, including two unconstrained problems, which are *cvxnonsep_psig40* and *cvxnonsep_psig40*, and the other problem, which is *cvxnonsep_normcon40*, is a constrained problem. These are convex mixed-integer nonlinear programming problems. We construct a finite combination set by fixing the integer variables to predefined values, thereby transforming the original MINLP into an MCNLP formulation, and solve them with the **GNN-NavCo**-aided optimizer to compare with the original optimizer on performance metrics. The optimizer used here is **MDPSO**, a state-of-the-art heuristic optimization algorithm [4]. Both the MDPSO and GNN-MDPSO have a population size of 101, and the total iteration number is set to 100. The optimization is set to automatically terminate if no improvement in feasibility or objective value is observed over 15 consecutive iterations. All the other parameters are set to the default.

Unconstrained Problem

We first use an unconstrained problem as the experimental setup, using the **cvxnonsep_psig20** benchmark, which consists of 10 discrete and 10 continuous variables and involves no con-

straints. The discrete configuration space is represented by a candidate pool Z comprising 101 configurations, each corresponding to a specific assignment of the discrete variables. To evaluate scalability, we further extend the experiments to larger configuration pools containing 501 and 1001 candidates. Increasing the size of Z enables us to systematically assess the robustness and stability of the proposed approach as the combinatorial search space expands, while also evaluating its ability to learn a coherent representation of the underlying energy landscape as complexity increases. We further extend the study to the **cvxnonsep_psig40** benchmark, which increases the problem dimensionality to 20 discrete and 20 continuous variables. This setting introduces a significantly larger and more complex configuration space. For consistency, the candidate pool is fixed to 101 configurations, and performance is evaluated under the same sampling and training protocol. This experiment isolates the effect of increased problem dimensionality while keeping the sampling budget constant.

Constrained problem

We then consider **cvxnonsep_normcon40**, which consists of 20 discrete and 20 continuous variables with one constraint. The candidate pool is fixed to 101 configurations, similar to the unconstrained setting. In this case, the presence of constraints introduces additional structure into the energy landscape through penalty term.

Data Sampling

Based on the GNN-Reco study, we sample continuous variables using Latin Hypercube Sampling (LHS) within their prescribed bounds, providing a space-filling design that ensures uniform coverage of the continuous domain. For each sampled continuous context $\mathbf{X}_q$, a candidate subgraph $\mathcal{G}_q$ is constructed by randomly selecting nodes from the global configuration pool $\mathbb{Z}$. This stochastic sampling ensures broad and unbiased coverage of the combinatorial space while maintaining computational efficiency. Given a sampled pair

$(\mathcal{G}_q, \mathbf{X}_q)$, all configurations within the subgraph are evaluated under the same continuous context to obtain their corresponding scalar energy values. From these node-wise evaluations, pairwise energy differences are computed to define directed edge attributes, yielding a fully connected graph representation in which each edge encodes the relative energy difference between two configurations.

3.10 Results and Discussions

GNN Training and Evaluation

Figure 3.5 and Figure 3.4 shows the training and validation loss of all the GNN-based models we used for the upcoming optimization case studies. Training loss converges in all cases, however the validation loss does keep oscillating in some of the cases. Hence, based on the validation loss history, we use the model before overfitting is observed to occur in each case.

Table 3.1: Mean Optimized Objective of the Case Studies over 5 Runs

Problem	Method	Error w.r.t. True Optimum
cvxnonsep_psig20 (101 Combination)	MDPSO	1.930
	GNN-MDPSO-30SN	**0.044**
cvxnonsep_psig20 (501 Combination)	MDPSO	3.029
	GNN-MDPSO-30SN	**0.446**
cvxnonsep_psig20 (1001 Combination)	MDPSO	0.595
	GNN-MDPSO-30SN	0.984
	GNN-MDPSO-300SN	0.777
cvxnonsep_psig40 (101 Combination)	MDPSO	48.398
	GNN-MDPSO-30SN	**9.918**
normcon40 (101 Combination)	MDPSO	6.75
	MDPSO-cons*	6.631
	GNN-MDPSO-30SN	**5.651**

* This MDPSO is run with the standard constraint formulation without utilizing the penalty function. Error w.r.t. true optimum is reported as the difference in objective function value.

Unconstrained benchmark problem

Figure 3.6 shows the objective convergence histories of the unconstrained optimization case studies and the energy convergence history of the constrained optimization case studies. Table **??** shows the mean optimized objectives of each case study, and the optimal method has been bolded. In most cases, the **GNN-NavCo**-aided MDPSO (denoted as GNN-MDPSO) leads to better results compared with the original MDPSO.

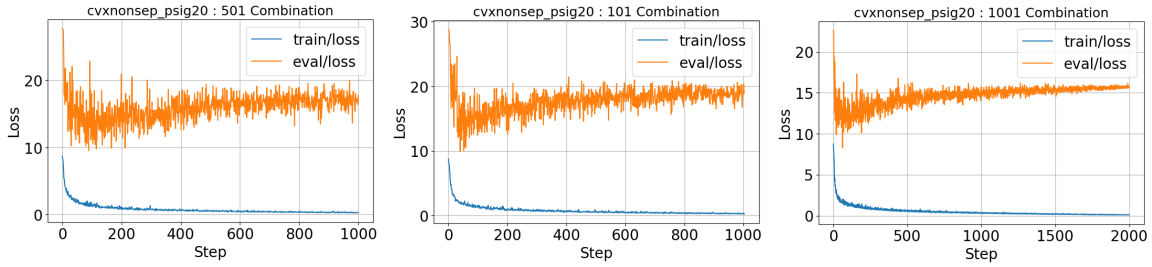

Figure 3.4: Training and validation loss of 3 EFGN Models for the benchmark problems. cvxnonsep_psig20 has been trained 3 times with 101, 501 and 1001 Combinations

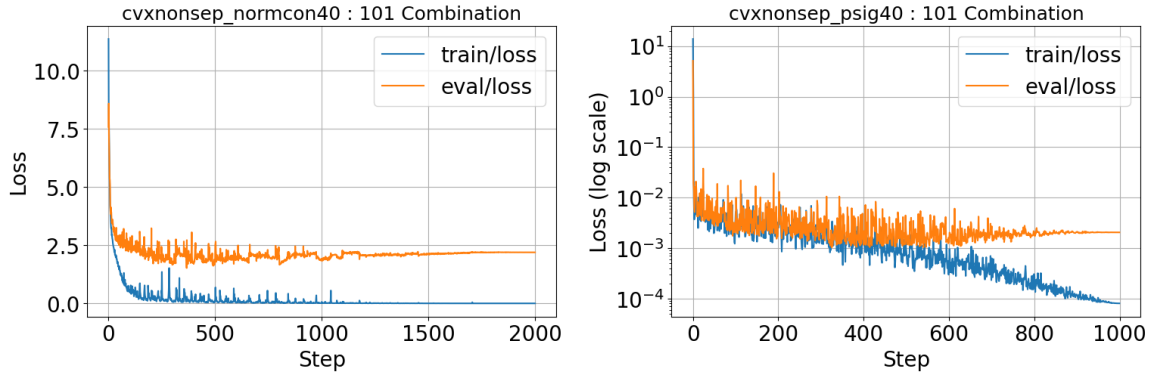

Figure 3.5: Training and validation loss of 2 EFGN Models for the benchmark problems cvxnonsep_normcon40 and cvxnonsep_psig40 , the loss values were between 0-1, therefore plotted against log loss for resolution

cvxnonsep_psig20: The convergence history of the optimization on *cvxnonsep_psig20* is shown in Figure 3.6. The shaded region represents the range (minimum to maximum) of the objective values over five independent runs with different initial populations and random seeds. Within each run, both the original MDPSO and the GNN-MDPSO share identical initial populations. However, their convergence histories begin from different starting

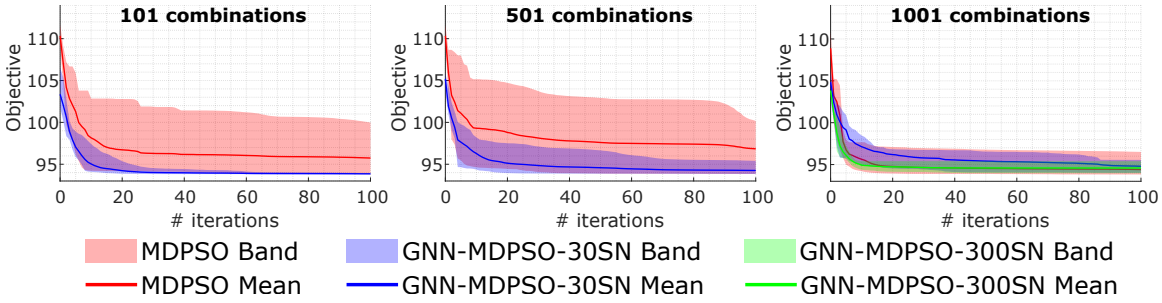

Figure 3.6: Objective (Energy) convergence history of MDPSO and GNN-MDPSO on the *cvxnonsep_psig20* with different number of valid combinations considered in the optimization. For the 1001 combinations optimization, a GNN-NavCo trained with 300-node subgraphs is applied, while the other two optimizations used GNN-NavCo trained with 30-node subgraphs.

points. This discrepancy arises because, in GNN-MDPSO, the initial population is first processed by **GNN-NavCo**, which recommends alternative configurations for function evaluation, thereby altering the effective starting point of the optimization process as expected. When the size of the combination set is 101, the GNN-MDPSO optimizes the objective to 93.855, which is only 0.044 worse than the true optimal (93.811). The GNN-MDPSO converges 12.93% more than the original MDPSO. Furthermore, the GNN-MDPSO is much more robust, leading to convergence histories with much less variation than the original MDPSO. When the size of the combination set increases to 501, both methods exhibit higher variances due to the larger combinatorial search space, but the GNN-MDPSO still shows a better-converged objective and lower variance across the five runs. As the size further increases to 1,001, the variance of GNN-MDPSO remains relatively stable; however, its convergence rate becomes slower than that of the original MDPSO. A plausible explanation is that the number of nodes sampled in the subgraph is insufficient to effectively explore the much larger combinatorial space. To investigate this, five additional experiments are conducted with a larger subgraph size of 300 nodes (denoted GNN-MDPSO-300SN). As shown in Fig. 3.6, both convergence performance and variance are significantly improved with more nodes in the subgraphs.

cvxnonsep_psig40: After doubling the number of continuous and combinatorial vari-

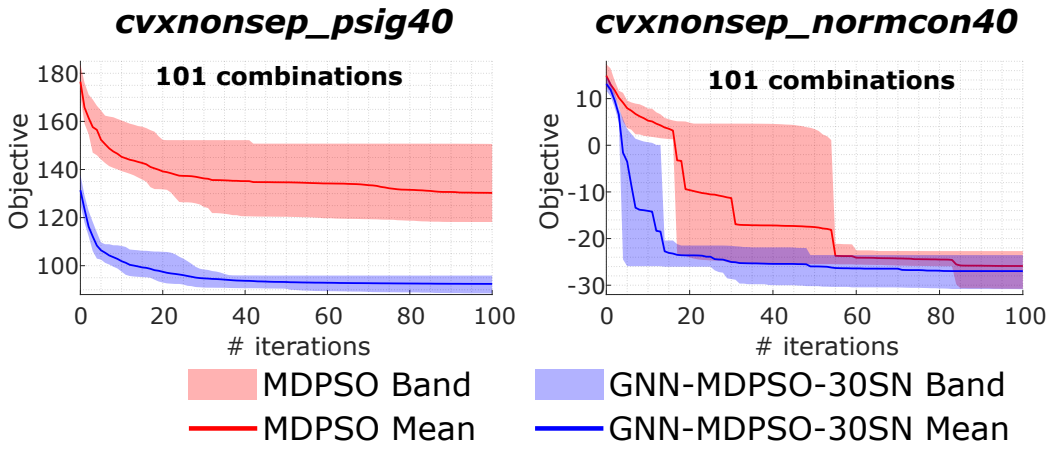

Figure 3.7: Objective (Energy) convergence history of MDPSO and GNN-MDPSO on the unconstrained problem *cvxnonsep_psig40* and constrained problem *cvxnonsep_normcon40*.

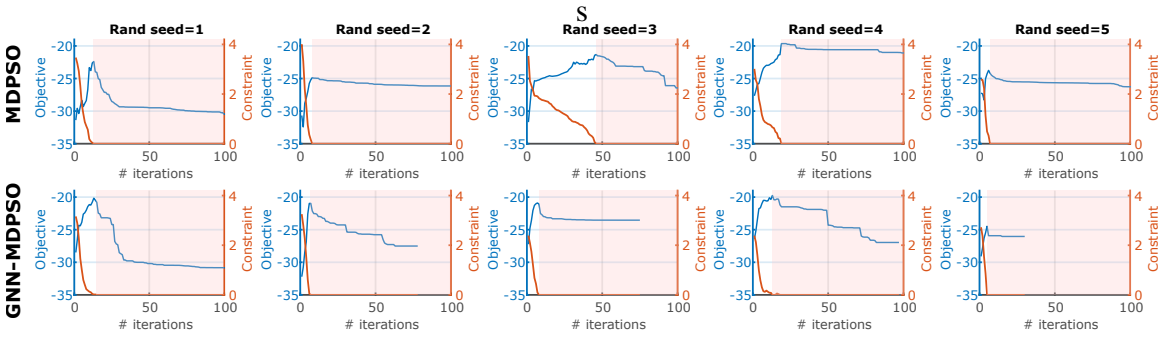

Figure 3.8: Objective and Constraint Convergence History of MDPSO with standard constraint formulation and GNN-MDPSO with penalty function (with constraint and objective recalculated) on the constrained problem *cvxnonsep_normcon40*. The pink shaded area represent the feasible region.

ables, GNN-MDPSO maintains strong convergence performance, as the left plot in Figure 3.7 shows. The final optimized objective remains significantly lower than that obtained by the original MDPSO, while exhibiting reduced variation across runs. These results indicate that GNN-MDPSO maintains high robustness as the design space dimensionality increases. This suggests that the proposed framework is scalable with respect to problem complexity and can effectively handle higher-dimensional mixed combinatorial optimization problems without significant performance degradation.

Constrained benchmark problem

cvxnonsep_normcon40: After switching the problem to constrained optimization problem, Figure 3.7 shows the convergence history of the two methods. It can be observed that the GNN-MDPSO is still robust and converging much faster, and leads to the mean objective being more optimal than the original MDPSO. Note that in this figure, the MDPSO also runs with the penalty function.

Figure 3.8 shows the constraint violations and the objective convergence history of the MDPSO with standard constraint formulation and the GNN-MDPSO with penalty functions, which can provide a perspective on whether the order-preserving search combined with the penalty function can efficiently lead the optimizer to the feasible region. It shows that the GNN-MDPSO enters the feasible region (the shaded section) faster than the original MDPSO, proving that the order-preserving search works well in the constrained problem as well. Also, 3 out of 5 runs of the GNN-MDPSO achieved more optimal objectives, indicating the potential of using this GNN-aided optimization concept in an an unconstrained solver in the future as well.

Hyperparameter Tuning of Cyclical Regularizer

We experimented with three cyclical regularization weights of 0,0.003 and 1 on the *cvxnonsep_psig40* problem with a linear transformation function of constant C = 1000 on a dataset

we created where the discrete Z structure is identical , but they differ sharply in the continuous variable distribution, which is the main source of skew. We have done this to exhibit a probable condition that can come during real-world simulation applications, where X_{cont} could be skewed due to various factors, including the preferred value range in the current operating systems or even a technical limit. Thus, we experiment in this setup to provide direction on the nature of λ_{cycle}. In table (3.10) $\lambda_{cycle} = 1$ yields the highest validation sign accuracy (0.99517) and the lowest validation edge loss (0.96394), indicating that a strong cycle-consistency constraint improves both predictive correctness and structural learning quality. The $\lambda_{cycle} = 0.003$ setting performs competitively, while $\lambda_{cycle} = 0$ yields the weakest performance, with reduced sign accuracy and increased edge loss. When we observe table (3.10), we observe that the $\lambda_{cycle} = 0.003$ gives the best metric of both sign accuracy as well as validation edge loss. Although $\lambda_{cycle} = 1$ still performs reasonably well, its slight decline relative to cycle = 0.003 suggests that, under skewed distributions, excessively strong cycle enforcement may overconstrain the model. Key takeaways from this are that cycle regularization consistently improves performance compared with omitting it altogether, demonstrating its value as an inductive bias in the learning process. Second, the optimal magnitude of this regularization is data-dependent and is a hyperparameter to tune. This gives us a rule of thumb: in highly complex or skewed data distributions, we can apply a soft regularization rather than a strong one.

Hyperparameter	**Sign Accuracy**	**Edge Loss**	**Cycle Loss**
$\lambda_{cycle} = 1$	**0.99517**	**0.96394**	0.000007
$\lambda_{cycle} = 0.003$	0.99172	1.01889	**0.00001**
$\lambda_{cycle} = 0$	0.99080	1.09397	16.22876

Table 3.2: Experiment Results for Uniform data regime

Hyperparameter	Sign Accuracy	Edge Loss	Cycle Loss
$\lambda_{cycle} = 1$	0.97333	125.38064	**0.000001**
$\lambda_{cycle} = 0.003$	**0.98011**	**98.25798**	0.000005
$\lambda_{cycle} = 0$	0.96701	132.70535	118460.29

Table 3.3: Experiment Results for Skewed data regime

Computing Cost Comparison

It is observed that in most of the above cases, the number of iterations that GNN-MDPSO spent to reach convergence is $\tilde{2}0$ iterations (except in *cvxnonsep_psig20* with 1001 combinations), fewer than the original MDPSO. Since both methods have a population size of 101, this results in approximately 2020 fewer function evaluations for the GNN-MDPSO to converge. Considering all the above case studies, the GNN training set is generated with 2,400 function evaluations, and the GNN-MDPSO converges at a cost similar to that of the original MDPSO. However, the above results also show that, although the MDPSO incurred a similar cost for function evaluation, it was not able to find a better optimum on average or robustly, and it is more easily affected by randomized initial populations. Therefore, based on the observation so far, GNN-MDPSO proves to be a more reliable and promising approach to solving the mixed-combinatorial optimization problem.

Chapter 4

Application on Robot Co-design

Robot Codesign is an optimization problem that simultaneously selects and configures interdependent robot components (hardware, software, and computation) to maximize task performance under constraints. Carlone et al. [2] frame co-design as a computational alternative to traditional human-led robot design, in which experts typically choose sensors, actuators, processors, algorithms, batteries, and structures through a slow, often sequential process. This traditional approach can be expensive and suboptimal, especially for low-cost, task-specific robots such as disposable drones for hurricane monitoring or search-and-rescue robots tailored to a particular environment.

The paper defines the robot co-design problem as the joint selection of robotic modules—for example, motors, cameras, embedded computers, perception algorithms, control algorithms, and planning algorithms—from available catalogs. Each possible module has features such as weight, cost, power consumption, torque, computational load, or sensing capability. The goal is to choose one option for each module so that the overall robot maximizes task-dependent performance metrics, such as endurance, agility, payload, or speed, while respecting constraints such as cost, compatibility, power, and resource limits. Carlone et al formulated the problem as a binary optimization problem, while Liu et al[18] formulated a similar robot codesign problem of optimizing the design of a tether net for

space debris capture as a more generalized model and formally classified it as a mixed combinatorial non-linear optimization problem, which is the foundational framework that this thesis works upon. However, in this section, we will be applying GNN-NavCo to the problem of optimizing the Active Tether-Net System to Capture Large-Scale Space Debris based on the work of Liu et al [19] and Boonrath et al [1].

4.1 Active Debris Removal Using Thether Net System

Space debris is becoming a serious problem because unused satellites, rocket fragments, and collision debris remain in Earth's orbit and can threaten active spacecraft. Even relatively small debris can cause major damage because objects in orbit travel at very high speeds. This underscores the importance of active debris removal (ADR). ADR, instead of only tracking debris or waiting for it to decay naturally, aims to physically remove dangerous objects from orbit. A tether-net system is one promising ADR method because it can capture debris without requiring direct contact between the chaser satellite and the target. This is especially useful for large, tumbling, or uncooperative debris, meaning debris that is not designed to be captured and may be rotating unpredictably. In this system, a net is deployed from a chaser spacecraft toward the debris. Once the net surrounds the object, a closing mechanism tightens the net, and a tether connects the captured debris back to the chaser so it can be stabilized or deorbited. The advantage of an active robotic tether-net system is that it provides the net with greater control during deployment. Instead of passively throwing the net and relying mostly on initial launch conditions, the system attaches multiple maneuvering units (MUs) to the corners or edges of the net. These MUs have small thrusters that guide the net toward the debris and adjust its shape and trajectory during flight. This improves the chance of successful capture, reduces the risk of collision between the debris and the chaser satellite, and can make the mission more fuel-efficient and reliable. This system, which uses an active tethernet instead of a passive one, was

proposed by Huang et al. [8].

Optimizing an active robotic tether-net system naturally falls within the mixed-combinatorial nonlinear programming (MCNLP) framework, as it requires the simultaneous handling of continuous design and control variables along with discrete hardware selections. The continuous space governs quantities such as maneuvering unit (MU) trajectories, timing, and velocity profiles, whereas the combinatorial space arises from component-level choices, such as thruster selection. These thrusters are not parameterized independently; rather, each option corresponds to a fixed bundle of properties, including maximum thrust, specific impulse, and dry mass. This induces a heterogeneous search space in which smooth optimization over continuous variables must be performed jointly with selection over a finite set of discrete configurations.

Within this setting, the proposed framework jointly co-designs the tether-net system by simultaneously optimizing its physical configuration and control strategy. The system consists of a chaser satellite, a deployable net connected via a main tether, multiple maneuvering units attached through corner threads, and a closing mechanism activated during capture. The debris target is modeled geometrically, and both structural parameters (e.g., net material properties and thread characteristics) and control inputs, formulated through a PID-based scheme, are treated as optimization variables. This integrated formulation enables coordinated tuning of morphology and control, thereby improving capture performance while accounting for realistic system constraints and dynamic interactions.

Traditional mixed-combinatorial nonlinear programming methods can handle this type of problem, but they become inefficient when each candidate design must be evaluated through a high-fidelity simulation. In tether-net debris capture, evaluating whether a design succeeds requires simulating net deployment, maneuvering-unit motion, debris interaction, and fuel consumption. If the optimizer has to directly simulate every possible combination of design variables and hardware choices, the computational cost becomes very high. This makes conventional brute-force or direct-optimization approaches impractical for large de-

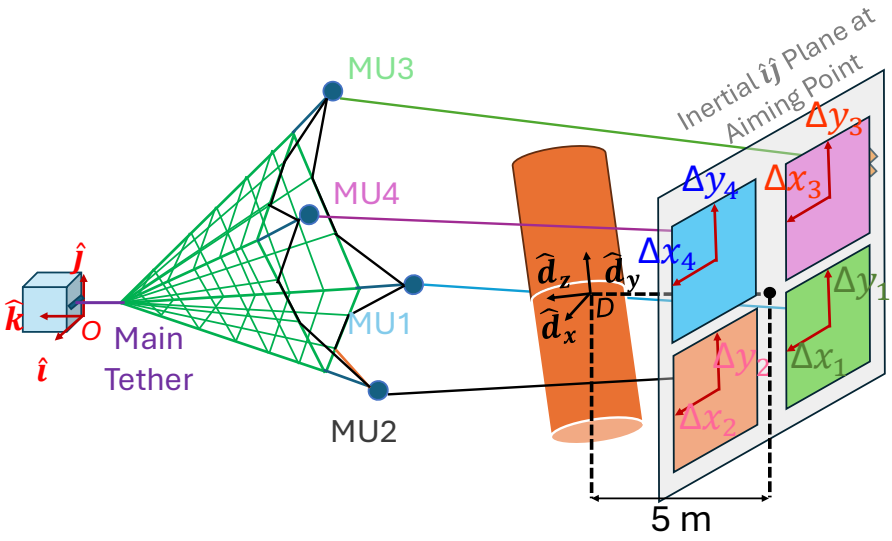

Figure 4.1: Diagram Tether-net System. The cross marks are examples of aiming points of MUs

sign spaces; thus, we will apply the GNN-NavCo Framework to address this problem.

4.2 Debris Capture Process

The capture process is governed by both the controlled dynamics of the maneuverable units (MUs) and the geometric quality of the net enclosure around the debris. Each MU is guided toward a parameterized aiming point defined relative to the debris center of mass (CoM), with offsets $(\Delta x_i, \Delta y_i)$ serving as continuous design variables that shape the approach trajectory. After a fixed maneuvering duration, the closing mechanism is activated, whereby the perimeter nodes of the net are pulled together through winch-driven threads, forming a constrained enclosure. This process results in a discrete closure state quantified by the number of locked node pairs N_L, with a maximum achievable value corresponding to full closure. Thus, the capture outcome depends jointly on the continuous control trajectory and the discrete closure configuration induced by the net dynamics.

To evaluate capture quality, a scalar metric termed the Capture Quality Index (CQI) is

defined to measure geometric conformity between the net and the debris. Specifically,

$$J = 0.1\frac{|V_n - V_D|}{V_D} + 0.1\frac{|S_n - S_D|}{S_D} + 0.8\frac{|q_n|}{L_c},$$

where V_n and S_n denote the convex hull volume and surface area of the net, V_D and S_D are the corresponding debris properties, q_n is the distance between the CoM of the net and the debris, and L_c is a characteristic length scale of the debris. This formulation emphasizes spatial alignment and enclosure fidelity, with a higher weight assigned to CoM alignment. A capture is deemed successful if the final CQI satisfies $J^* < \epsilon$ with $\epsilon = 2.5$ and the closure constraint $N_L = N_{\max}$ is met. This dual criterion enforces both geometric containment and structural integrity, enabling the optimization framework to rigorously balance trajectory design, material configuration, and capture performance.

4.3 MCNLP Optimization Problem

For tether-net systems, the physical design of the net and the controls for the Maneuverable Unit (MU) greatly impact the probability of capture success. Here, the goal of the optimization is to minimize the cost of fuel, m_{prop}, while achieving successful capture during the mission to capture the space debris.

The cost of fuel is calculated by the type of MUs and the thrust that they deliver. The thrust is controlled by the PID controller, and the maximum thrust is specified by the manufacturer. This problem, like all MCNLP problems, has a continuous and combinatorial part, which we will discuss briefly.

The continuous variables have the control parameters of the MUs denoted by $\Delta x_{i,i=1,2,3,4}$, $\Delta y_{i,i=1,2,3,4}$ and $v_{i,i=1,2,3,4}$, m_{MU}, mass of each MU and the net morphology parameters which include the net thread radius r_{net}, corner thread radius r_{corner}, length of corner thread L_{net} and the length of net thread L_{net}. These together constitute 17 continuous variables.

The primary combinatorial components include selecting one of five candidate thrusters,

ranging from cold-gas to combustion-based chemical systems—each characterized by its maximum thrust magnitude F_T, specific impulse I_{sp}, and dry mass m_T. Additionally, the model considers three high-strength fibers for the net material properties, defined by their Young's Modulus E_{n} and density ρ_{n}, while assuming a uniform Poisson's ratio across all options. The net topology is governed by the integer variable N_k, which defines the number of knots along a single side of the net for a total grid of N_k^2 knots. Finally, the closing mechanism index K_{cls} identifies the specific node on the net's periphery where the closing mechanism is attached; a higher index indicates a position farther from the central node. In total, the number of features is 8, and the total number of combinatorial choices is 180. The optimization is then defined mathematically as follows.

$$
\begin{aligned}
&\min_{\mathbf{Z}_{\text{comb}},\mathbf{X}_{\text{cont}}} \quad f(\mathbf{Z}_{\text{comb}},\mathbf{X}_{\text{cont}}) = f_{\text{MU}} \\
&\text{subject to} \quad g_1(\mathbf{Z}_{\text{comb}},\mathbf{X}_{\text{cont}}) = I^*_{\text{CQI},j} \leq 2.5, \forall\, j = 1,2,...,10 \\
&\qquad\qquad\quad\; g_2(\mathbf{Z}_{\text{comb}},\mathbf{X}_{\text{cont}}) = \gamma_{\text{tension},j} \leq 1, \forall\, j = 1,2,...,10 \\
&\qquad\qquad\quad\; h_1(\mathbf{Z}_{\text{comb}},\mathbf{X}_{\text{cont}}) = N_{L,j} = 12, \forall\, j = 1,2,...,10 \\
&\text{where} \quad f_{\text{MU}} = \sum_{j=1}^{10} m_{\text{prop},j}, \quad \forall\, j = 1,2,...,10 \\
&\qquad\qquad\quad\; \mathbf{Z}_{\text{comb}} = \begin{bmatrix} F_T & I_{\text{sp}} & m_T & E_{\text{n}} & \rho_{\text{n}} & N_k & K_{\text{cls}} \end{bmatrix} \\
&\qquad\qquad\quad\; \mathbf{X}_{\text{cont}} = \begin{bmatrix} \Delta x_1 \ldots \Delta x_4 & \Delta y_1 \ldots \Delta y_4 & v_1 \ldots v_4 & m_{\text{MU}} & r_{\text{thread}} & r_{\text{corner}} & L_{\text{net}} & L_{\text{ct}} \end{bmatrix} \\
&\qquad\qquad\quad\; \mathbf{Z}_{\text{comb}} \in \mathbb{Z}
\end{aligned}
\tag{4.1}
$$

The fuel consumption is calculated based on the following equation f_{MU}.

$$f_{\text{MU}}\left(\mathbf{Z}_{\text{comb}}, \mathbf{X}_{\text{cont}}\right) = \begin{cases} m_{\text{prop}} \quad , & \text{if } I^*_{\text{CQI}} < 2.5 \text{ and } N_L = 12 \\ \ln\left(\left(I^*_{\text{CQI}} - 2.5\right)^2 + 1\right) + \ln\left(\left(N_{\text{L}} - 12\right)^2 + 1\right) + \beta \quad , & \text{otherwise} \end{cases} \tag{4.2}$$

Continuous Variables

Category	Sub-category	Variable	Bound
Control	MU Control	$\Delta x_{i,i=1,2,3,4}$ [m]	[-5, 5]
		$\Delta y_{i,i=1,2,3,4}$ [m]	[-5, 5]
		$\nu_{i,i=1,2,3,4}$ [m/s]	[-1, 4]
Physical Design	MU Mass	m_{MU} [kg]	[2, 3]
	Net Morphology	r_{thread} [m]	[0.0005, 0.0015]
		r_{corner} [m]	[0.0001, 0.0015]
		L_{net} [m]	[19, 25]
		L_{ct} [m]	[0.5, 2]

Combinatorial and Integer Variables

Category	Sub-category	Variable	#1	#2	#3	#4	#5
Combinatorial	Thruster	F_T [N]	8.9	3.6	6.1	5.5	6.0
		I_{sp} [s]	60.0	57.0	277.0	253.0	250.0
		m_T [kg]	0.37	0.023	0.6	0.48	0.25
	Net Material	E_{n} [GPa]	70.0	70.5	112.4	-	-
		ρ_{n} [kg/m^3]	1390.0	1440.0	1440.0	-	-
	Net Shape	N_k	9	11	13	-	-
		K_{cls}	-2 to 0	-2 to +1	-3 to +1	-	-

Table 4.1: Bounds and Choices of Continuous and Combinatorial and Integer Variables for Tether-Net Fuel Consumption Optimization

4.4 Experiment Setup

The experiment is set up based on the initial flow process as described in Fig. 3.2, where the function evaluation module is replaced by the Active tethernet simulator from Boonrath et al. [1], implemented in Google Jax. The simulator accepts continuous and combinatorial

parameters, which are transformed into the simulator-accepted object and then fed to the simulator to first create the scenario and conduct the experiment, followed by calculating the score or the fuel cost using equation (4.2).

We first collect data samples from the simulator for training the EFGN model. Here we have taken 3000 total evaluations for training the EFGN model. Since we take subgraphs of 30 nodes each, our total subgraph count comes to 100. That is, in each subgraph, we have sampled 30 distinct combinatorial node choices and a common continuous context. Therefore, the total continuous context or variable sampled is 100. After this, we train the model to achieve 75% Sign accuracy and stop early at 581 epochs, at which point the validation edge loss metric reaches its lowest value during training. It is observed that in earlier experiments with benchmark problems, the accuracy was in the higher range, around 90% to 95%, but it should be noted that those were convex problems. Here, the problem is likely highly nonconvex, and the EFGN model is very likely to learn a smoothed convex function on top of the highly nonconvex hyperplane. We leave this to future work to modify the EFGN so that it can handle non-convex functions natively.

Post this, the trained model was embedded in the MDPSO algorithm, as in the previous evaluation in Chapter 3, and the continuous vector was optimized using MDPSO, while the combinatorial vector was recommended by the GNN-NavCo Model.

4.5 GNN Training

The current GNN architecture utilizes a 3-layer Graph Neural Network (GNN) backbone with a hidden dimension of 128. The embeddings are scaled to 64 dimensions to provide richer feature representations. To map graph features to cost predictions, the model employs a deep, 5-layer residual Multi-Layer Perceptron (MLP). Finally, to prevent overfitting given the increased capacity, a moderate dropout rate of 30% is applied throughout the network. For the data collection part, we sample 3000 function evaluations from the

high-fidelity simulator and construct 100 subgraphs, each having 30 nodes representing the combinatorial features and one common continuous variable. We train the GNN-NavCo Model while tracking edge and cycle-consistency regularization losses. Model selection is performed using the minimum validation edge loss, and the corresponding checkpoint was saved at 581 Epochs. In fig (4.2), early on, we can see that the edge loss stays stagnant, and it can be attributed to early stage learning by the model of the hidden representation, and also to the act of stabilizing the cycle consistency regularization.

When we compare figs (4.2) and (4.3), we observe that the cycle loss decreases rapidly until 50 epochs. After this point, the edge loss drops sharply, whereas the cycle loss largely plateaus with occasional spikes,although the overall spike is negligible. A similar edge-loss pattern is also observed across experiments with different values of the cycle-loss weight. This suggests that the abrupt improvement in edge loss is not caused solely by the regularizer, but rather by the model learning the representation for the edge prediction. This is evident in Fig (4.4), where we see that before roughly epoch 50-60, the model is having fluctuating sign accuracy, and post the stabilization mark, when we see a drop in loss, the sign accuracy stabilizes to 75-76% mark. Post this, the edge loss decreases sharply, indicating that once the model has learned a stable ordering or sign pattern, it can more effectively refine edge predictions.

In Fig (4.5), we see that the dataset is globally concentrated in a relatively low energy region, but the individual subgraphs still exhibit a substantial internal variation, so the cycle consistency regularizer still plays a local role. In this setting, the regularizer does not seem to dominate optimization, but it does appear to improve generalization slightly, as seen in the improved validation performance for models trained with cycle regularization. This points out that in problems, when the sampling produces a widely incoherent data distribution, using a strong cyclical regularizer is likely to push the model towards having better validation edge loss and thus, better prediction on unseen scenarios, and can act as a counter mechanism when the training data is widely distributed, creating inconsistencies.

The training dataset contains the score, or basically the energy as per equation (2.3) of every valid configuration Z_{comb} and X_{cont}, which is basically the combination of each node in a subgraph and the continuous context. But as we know, we learn the edge differences, which is the ground truth. This value is not precomputed, but is calculated on the fly during the training. That is, for two valid nodes Z_i and Z_j, it will be computed before taking their difference. This creates a situation where, when the energies are very close to each other, the difference, which is the learning difference between them, can be minuscule, which might negatively affect the training. For every close score region, for example, like 0.003 and 0.002, the difference is 0.001, which, when in a dataset of differences like 12 and 13, might get recognized as noise and could get ignored, leading to lower sign accuracy for the model. In order to resolve this, we can always use a **monotonic** function to transform the energy space, much like using a magnifying glass to give the model a better view of the near-tie regions.

In Table (4.2), we conduct the experiment with different transformation functions. Since our aim is to get a model that can predict the best sign accuracy along with magnitude in the best way possible, we considered a rank-normalized version to run with MDPSO. When we observe table (4.5), we see that for rank normalization transformation, the sign accuracy is best at cyclical regularization $\lambda_{cycle} = 0.003$ and the best edge loss with $\lambda_{cycle} = 1$. We see exactly the same pattern with $1000 \times \log$ transformation function as well, as seen in table (4.5). In Chapter 3, we discussed that the λ_{cycle} value is a hyperparameter to tune, and we proposed a rule of thumb: use soft regularization for highly complex or skewed data. We verified that assumption here empirically and chose to use $\lambda_{cycle} = 0.003$ for the model when embedded in MDPSO training. This indicates that using a monotonic function can help in the resolution of near-tie spaces. Based on the data distribution and the operation of choice, we can choose the method. The training and validation chart for each of the settings for $1000 \times \log$ is attached in the appendix for the reader to view.

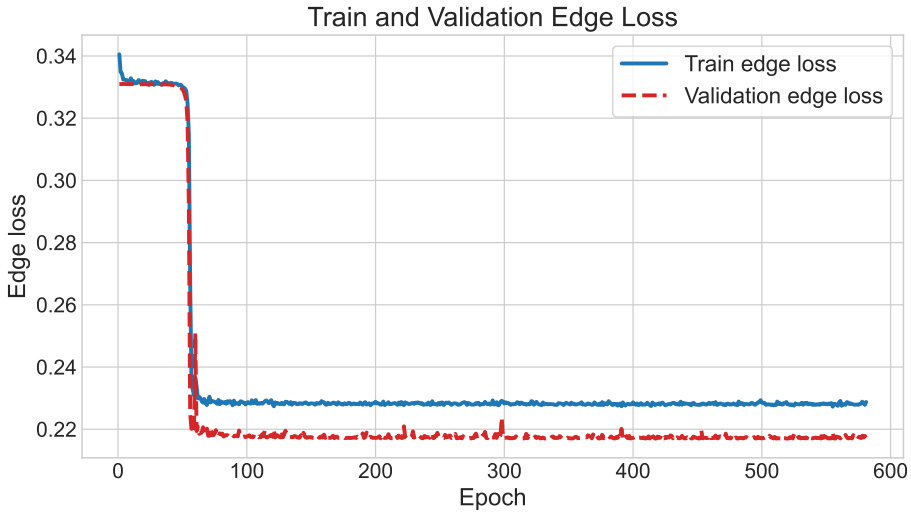

Figure 4.2: Training and validation loss of edge loss function

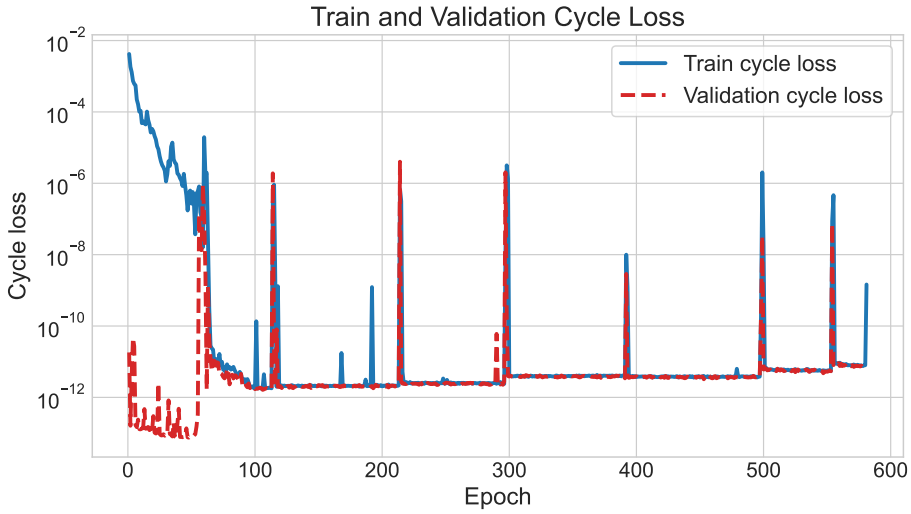

Figure 4.3: Training and validation loss of cyclical loss regularizer

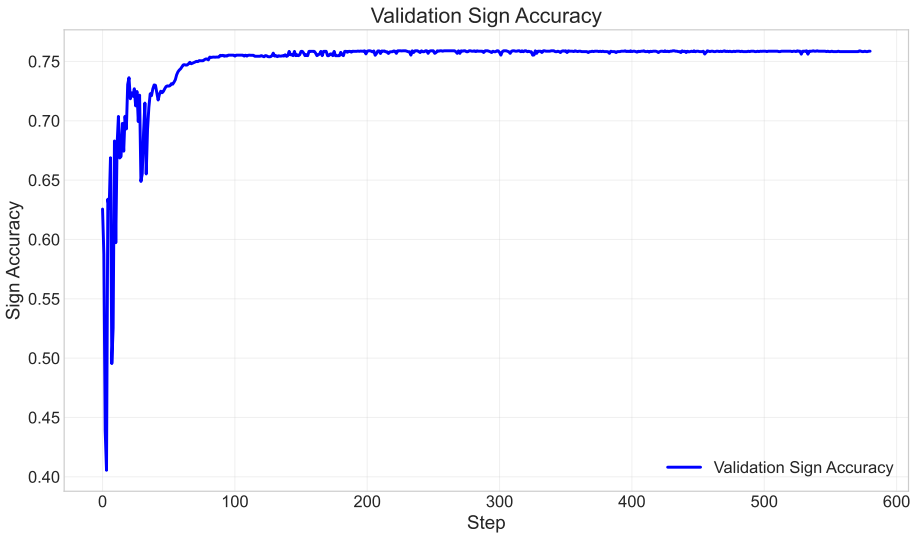

Figure 4.4: Validation Sign Accuracy of GNN-NavCo Model

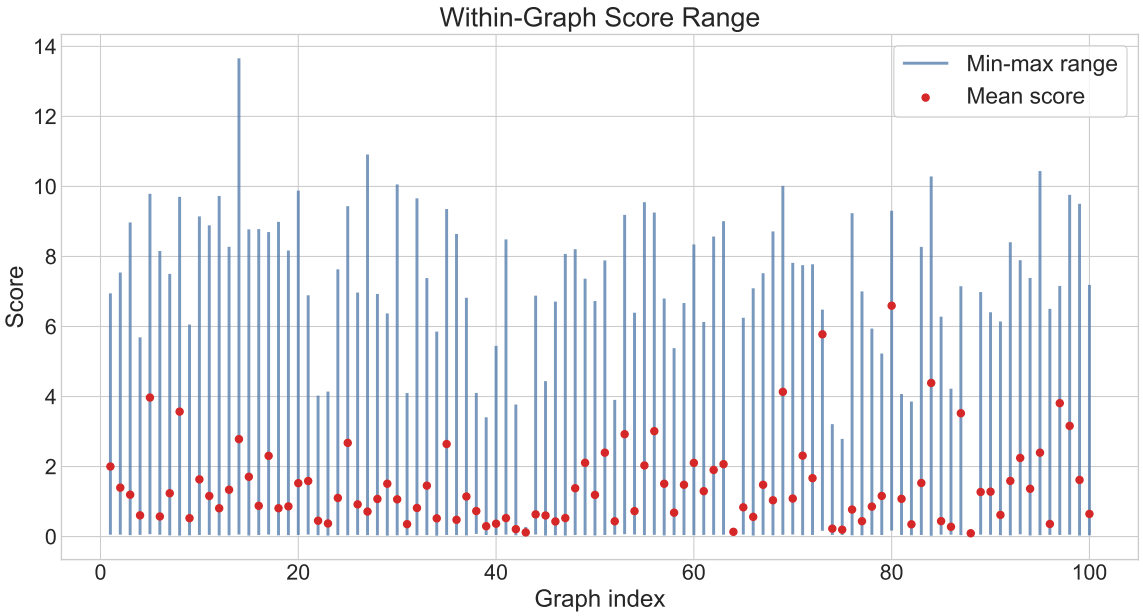

Figure 4.5: Score difference with individual graphs in training data

Transformation Function	Sign Accuracy (Step 200)
Rank Normalization	0.76414
$10 \times$ Log	0.74586
$1000 \times$ Log	0.74241
Linear(C=1000)	0.74161
Log	0.73782
Exponential	0.68655
Raw Energy	0.68563

Table 4.2: Sign Accuracy by Transformation Function Type at Step 200

Hyperparameter	Best Sign Accuracy	Edge Loss	Cycle Loss
$\lambda_{cycle} = 1$	0.76333	**0.04321**	1.16×10^{-12}
$\lambda_{cycle} = 0.003$	**0.76356**	0.04323	2.32×10^{-12}
$\lambda_{cycle} = 0$	0.7623	0.043224	1.85×10^{-12}

Table 4.3: Experiment Results with Rank Normalization Energy Transformation

4.6 Results and Discussion

The training, evaluation, and optimization of the GNN-ReCo framework were performed on a dual-socket Intel Xeon Gold 6148 machine (40 physical cores, 80 threads), equipped with 187 GB of RAM and an NVIDIA GeForce RTX 4060 Ti GPU with 16 GB of VRAM. Completing 53 iterations of the GNN-NavCo-assisted MDPSO required a total runtime of 26.5 hours.

The convergence behavior of the objective function is presented in Fig. 4.6. To assess whether further improvements were possible, a sequential quadratic programming (SQP) method (MATLAB's `fmincon()`) was applied to the continuous variables obtained from

Hyperparameter	Best Sign Accuracy	Edge Loss	Cycle Loss
$\lambda_{cycle} = 1$	0.74701	**128.21904**	2.23×10^{-7}
$\lambda_{cycle} = 0.003$	**0.74747**	128.22474	0.068097
$\lambda_{cycle} = 0$	0.74586	128.23426	15.1177

Table 4.4: Experiment Results with $1000 \times Log$ Energy Transformation

the GNN-NavCo-assisted MDPSO. However, no better solution was identified, indicating that the obtained solution is already locally optimal. In comparison, the baseline MDPSO converged after 27 iterations, whereas the GNN-NavCo-assisted variant reached convergence in only 8 iterations, resulting in 1,900 fewer function evaluations during the optimization phase.

It is important to note that the overall number of function evaluations for the GNN-NavCo-assisted MDPSO exceeds that of the standalone MDPSO by approximately 1,100 evaluations. Nevertheless, this does not diminish the appeal of the proposed approach. First, the additional evaluations mainly arise from the data generation and training stages of the GNN surrogate model, which can be executed offline. These stages are highly parallelizable and well-suited for high-performance computing (HPC) environments, enabling substantial reductions in wall-clock time despite the increased evaluation count. In contrast, the conventional MDPSO relies on sequential, online evaluations between iterations, limiting its time efficiency in time-sensitive settings.

Second, the scalability benefits of the GNN-NavCo framework are expected to become increasingly significant as the combinatorial search space grows in size and complexity. By leveraging learned structural representations and reducing reliance on costly data generation, the method is better positioned to handle large-scale problems, where purely optimization-driven techniques may struggle to efficiently explore the design space. Ongoing work aims to further validate this scalability advantage.

Table 4.6 summarizes the optimized design variables and objective values. Both methods converged to the same discrete configuration: a thruster with the highest specific impulse (I_{sp}) and second-highest thrust, combined with a net featuring the fewest meshes—collectively minimizing fuel consumption. Despite identical combinatorial selections, notable differences are observed in several continuous variables, including aiming points and net side length.

This discrepancy indicates that the optimization landscape is highly nonlinear, with

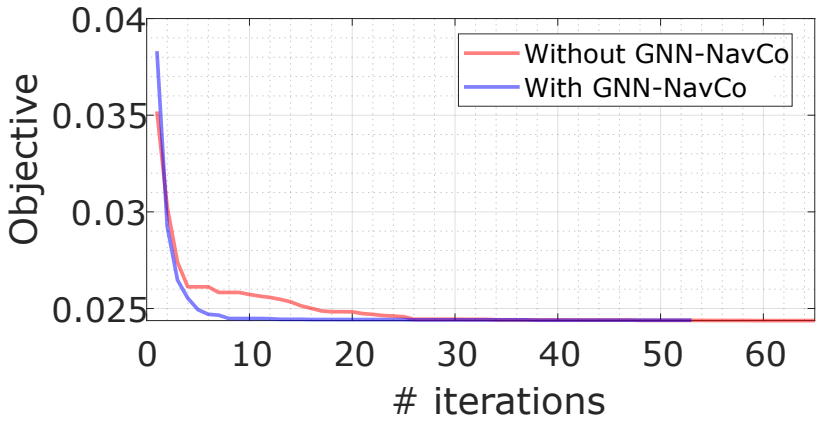

Figure 4.6: Comparison of objective convergence histories with and without GNN-NavCo assistance

multiple feasible solutions of similar performance. In other words, even under a fixed discrete configuration, the continuous design space contains multiple local optima or near-optimal regions that can achieve comparable objective values. The variation in continuous parameters suggests that the two methods converge to different solutions within this landscape rather than a single unique optimum. This reflects the problem's intrinsic multimodality, where trade-offs among continuous variables yield equivalent system performance. Consequently, the final solution is not unique, and different optimization trajectories may yield distinct yet equally valid designs, even when initialized identically.

Figure 4.7 presents rendered simulation results of the optimized design obtained using the GNN-NavCo-assisted approach. As all methods successfully achieved capture in this scenario, the metric f_{MU} corresponds directly to fuel consumption. Compared to the baseline design reported in [1], the optimized solution reduces fuel usage to just 11.1% of the baseline, underscoring the substantial impact of optimization on system efficiency.

Continuous Variable	GNN-NavCo-aided MDPSO	MDPSO	Baseline
$\Delta x_1, \ldots \Delta x_4$ [m]	[2.843, -4.980, 0.041, -4.997]	[-1.132, -4.060, 4.286, -4.999]	[0,0,0,0]
$\Delta y_1, \ldots \Delta y_4$ [m]	[0.569, 4.103, -1.799, -0.880]	[1.794, 1.963, -4.144, -2.404]	[0,0,0,0]
$v_1, \ldots, v_4$ [m/s]	[0.527, 0.785, 0.837, 0.841]	[0.654, 0.733, 0.846, 0.767]	[0,0,0,0]
m_{MU} [kg]	2.1862	2.7569	2.0
r_{thread} [m]	0.0011	0.0011	0.0010
r_{corner} [m]	0.0012	0.0009	0.0010
L_{net} [m]	23.315	19.667	22.0
L_{ct} [m]	1.532	0.931	2.414
Combinatorial Variable			
$F_{T,max}$ [N]	6.1	6.1	8.9
I_{sp} [s]	277.0	277.0	60.0
m_T [kg]	0.6	0.6	0.37
E_{n} [Gpa]	70.0	70.0	70.0
ρ_{n} [kg/m^3]	1390.0	1390.0	1390.0
N_k	9	9	11
K_{cls}	0	0	1
Objective			
f_{MU}	0.0244	0.0244	0.220

Table 4.5: Comparison between optimized variables and objective with and without the aid of GNN-NavCo

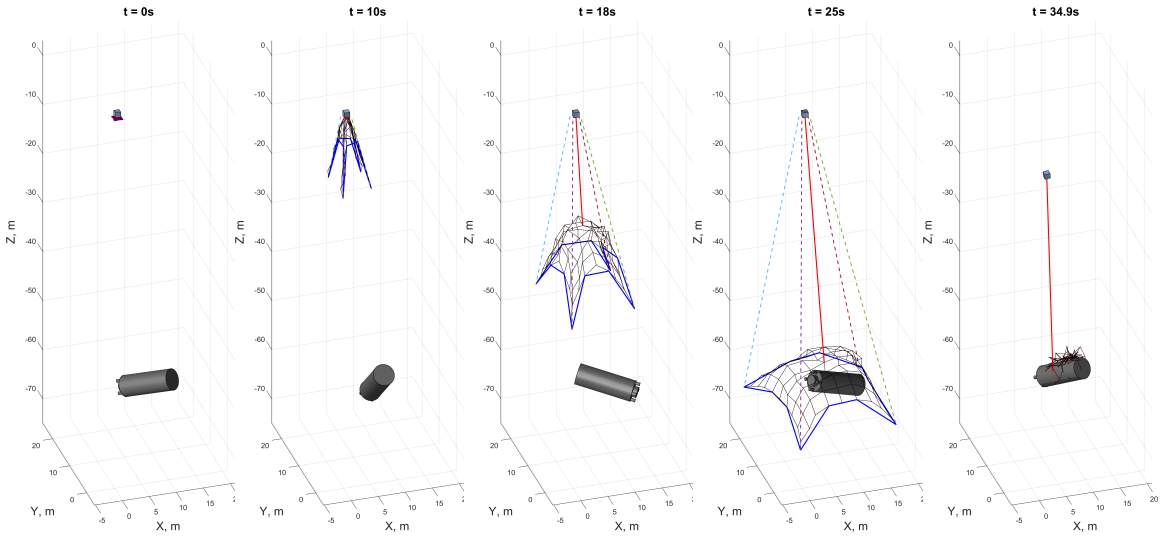

Figure 4.7: Rendered simulation screenshots of the GNN-NavCo-aided optimized design. The dashed lines are the minimum-energy reference trajectories for the MU. The following frames are shown in the above figure. **1) t=0s:** The beginning of the entire mission. **2) t=10s and 18s:** The MUs are maneuvered towards the optimized aiming points. **3) t=25s:** The thrusters are switched off, and the closing mechanism is switched on. **4) t=34.9s:** The capture is stabilized.

e

Chapter 5

Conclusion

5.1 Concluding Remarks

This paper developed an approach to abstract combinatorial spaces in mixed-combinatorial nonlinear programming (MCNLP) problems in a manner that enables scalable and interpretable recommendation of combinatorial configurations for candidate designs during optimization over the remaining continuous variables. In doing so, it addresses the lack of general-purpose optimization frameworks for MCNLP problems. The proposed formulation, termed GNN-NavCo, models the combinatorial search space as a graph, where combinatorial configurations are represented as nodes and pairwise relationships between configurations form the edges. For a fixed continuous context, the objective or energy function assigns a scalar value to each combinatorial configuration, thereby inducing a ranking over the graph. The pairwise differences of these energy values define a skew-symmetric edge flow on the graph, where each edge encodes the relative preference or directional improvement between two combinatorial states.

This formulation naturally aligns with the HodgeRank framework on graphs, where globally consistent rankings are represented as gradient flows induced by scalar potential functions on the nodes. In particular, the pairwise energy differences correspond to a dis-

crete gradient flow that provides a direction-aware representation of optimization progress over the combinatorial space. Consequently, the learning problem may be interpreted as learning a context-conditioned edge flow that approximates the globally consistent ranking structure of the underlying optimization landscape. The Hodge-theoretic interpretation further allows decomposition of the learned edge flow into globally consistent and locally inconsistent components through the combinatorial curl operator, enabling both interpretable ranking recovery and consistency analysis over the discrete search space. Consistency analysis following recovery of the node energy function remains an important direction for future work. The residual curl and non-gradient components of the learned edge flow may provide a certificate of reliability for the inferred ranking structure. Such measures could be incorporated during inference to detect uncertain or inconsistent optimization directions and suppress unreliable recommendations.

Thus, in this sense, GNN-NavCo learns a discrete, gradient-like ranking flow that guides optimization over mixed combinatorial spaces in a principled, graph-theoretic manner. Its use as an order-preserving combination recommender is demonstrated using a recently proposed decomposed solution framework for MCNLPs. GNN-NavCo is applied to a suite of analytical nonlinear benchmark problems with up to 20 continuous variables and 20 combinatorial features, and 100's of feasible combinations. The training process was found to be effective in providing stable learning of the GNN-based recommender. When used in optimization, it was found to usually yield better optimal solutions (except in a few cases) than a baseline optimization that uses indexification of the combinations, with both methods implemented with the same PSO solver. This is with similar overall function evaluations in mind, accounting for the additional investment required to train the GNN in our approach. Significantly lower variance in both the optimization history and final values is also observed compared to the baseline.

When applied to a robot codesign application problem, specifically, the tether net design problem, here, the co-design involved optimizing over the PID control parameters, net

morphology, and Maneuverable Unit (MU) choices, where the optimization objective was to capture the space debris successfully, while reducing the total fuel consumption. We can see that the GNN-NavCo model, trained on 3000 evaluations of the high-fidelity simulator, achieved a sign accuracy of 75%. Here, we also explored the application of different transformation functions in detail, showing that using the pairwise differences method gives us the ability to magnify the energy space, that is, increase resolution, in cases where the gradient signal would otherwise be minuscule due to near-tie situations. In a node-based approach, this can induce bias, but we show that, through the application of monotonic functions, the sign accuracy generally remains intact across these transformations, with rank normalization yielding the maximum value in this particular case, and linear magnification also giving a close second-best accuracy. We also tuned the cyclical regularization parameter and chose to move with a weight of 0.003, based on our earlier finding that using a cyclical weight as a soft regularizer is effective. We also experimented with cyclical regularization hyperparameter tuning along with energy transformations, and we got the same pattern of choices in both cases, showing that monotonic transformations only magnify the learning space and do not overly increase bias in the recommendation capacity of the model if both transformations are of equal standing. Post-training, we embedded the model along with the MDPSO algorithm, and it was seen to improve the performance of the algorithm. Even though the GNN-NavCo model was only trained with 3000 evaluations and 75% accuracy, we can see in the plot that the convergence to a feasible and acceptable optimized result is rather fast, helping us use a smaller number of evaluations to find an optimum. It was also found that increasing the size of the sampled sub-graph (used for abstraction and then recommendation) as the number of combinations in the problems increases is helpful.

5.2 Future Directions

More work is needed in the future to systematically identify sub-graph sizes (or a smarter sampling technique) to offset any comparative performance loss (as currently observed) compared to baselines for problems with a larger number of combinations. .While GNN-NavCo is currently used in conjunction with PSO, the underlying recommendation process and model formulation are not limited to PSO. An immediate direction for future work is to employ GNN-NavCo alongside other standard gradient-based algorithms (e.g., sequential quadratic programming) and gradient-free algorithms (e.g., genetic algorithms) that serve as general-purpose NLP solvers. Additionally, there is an opportunity to explore improved search mechanisms over the directed graph generated by GNN-NavCo, with probabilistic guarantees on order-preserving recommendations informed by model uncertainty. Applying GNN-NavCo to practical engineering design problems, supply chain optimization, and financial modeling could further illuminate its benefits under realistic computational resource constraints. Furthermore, expanding the design space to include sensor choices and a wider variety of net shapes would enable more meaningful exploration for autonomous tether-net systems. Application of the GNN-NavCo module with alternative optimization algorithms, especially gradient-based methods and Bayesian optimization, and the potential to update the GNN in situ during optimization, could provide additional insights into the generalized applicability of this approach for automated design guidance of mixed-combinatorial complex systems.

Appendix A

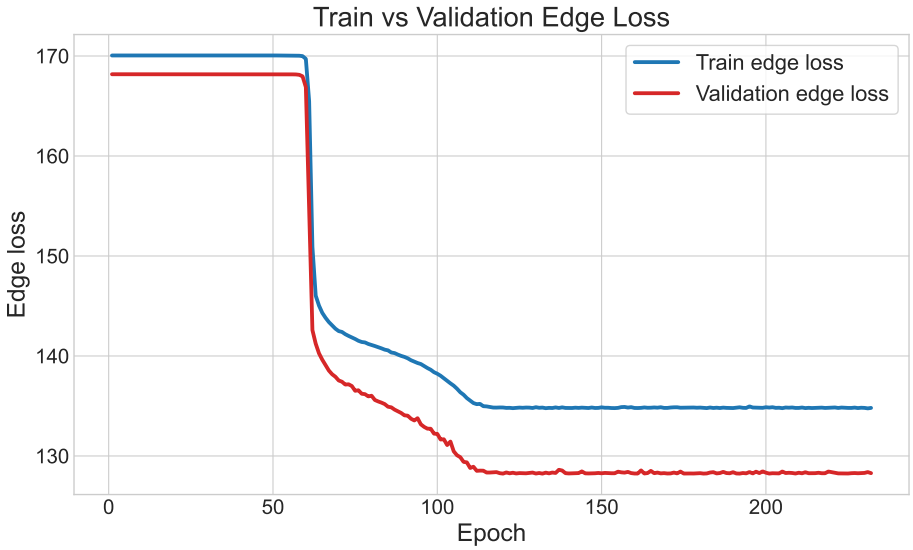

Figure A.1: Training and validation loss of edge loss function for cycle consistency weight $\lambda = 0$

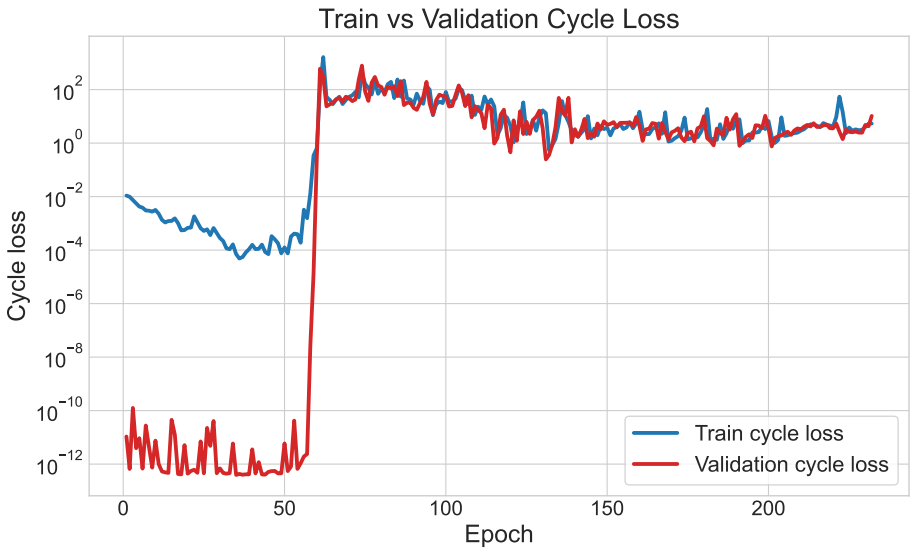

Figure A.2: Training and validation loss of cycle loss function for cycle consistency weight $\lambda = 0$

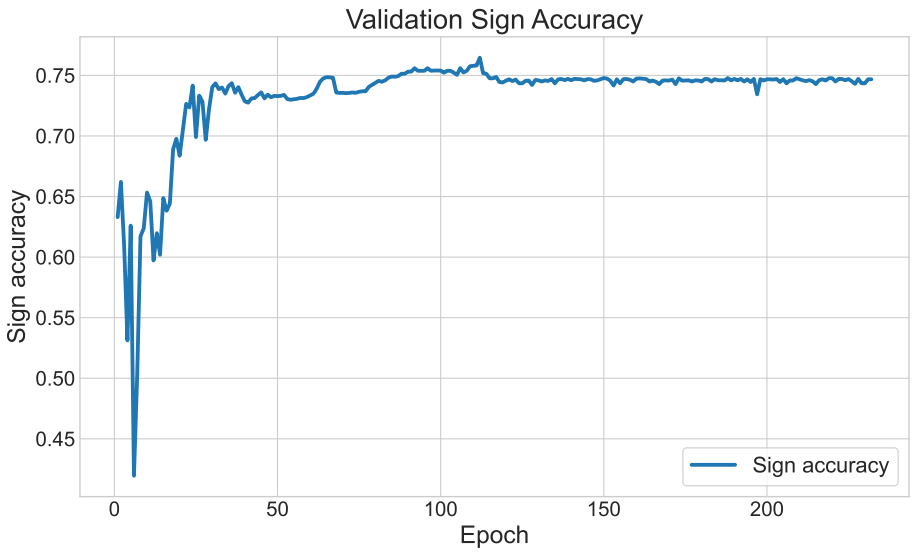

Figure A.3: Validation Sign Accuracy Curve for cycle consistency weight $\lambda = 0$

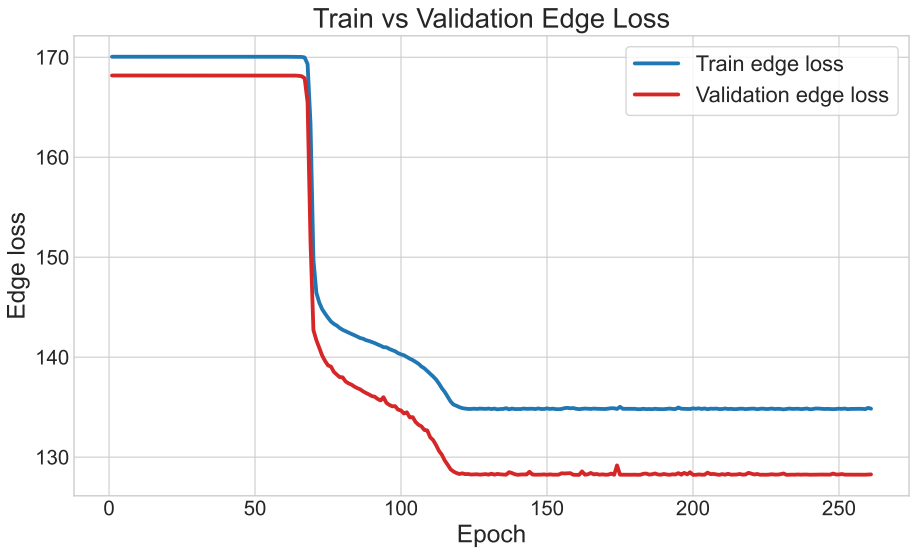

Figure A.4: Training and validation loss of edge loss function for cycle consistency weight $\lambda = 0.003$

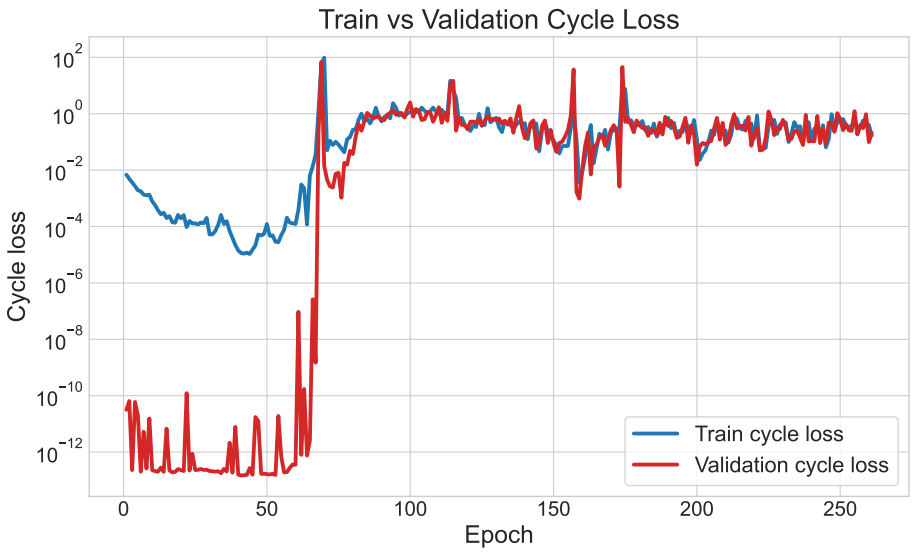

Figure A.5: Training and validation loss of cycle loss function for cycle consistency weight $\lambda = 0.003$

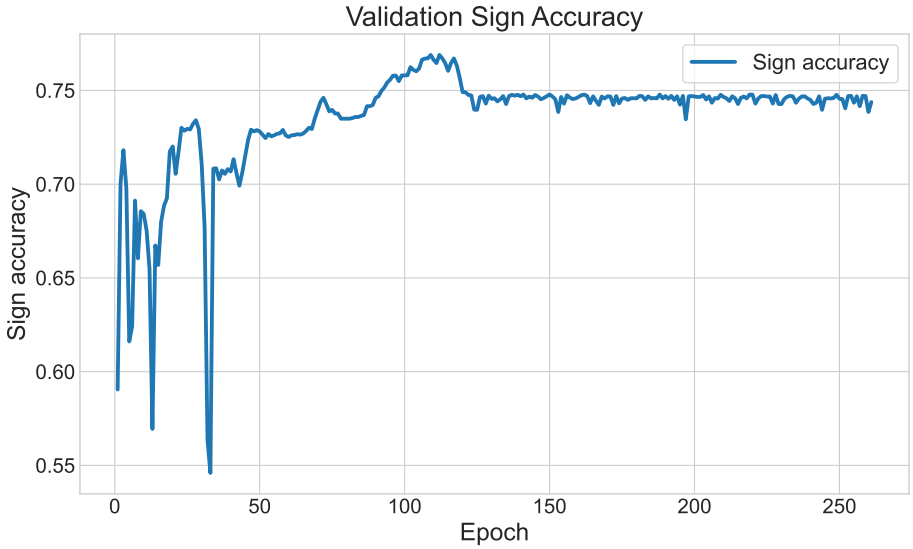

Figure A.6: Validation Sign Accuracy Curve for cycle consistency weight $\lambda = 0.003$

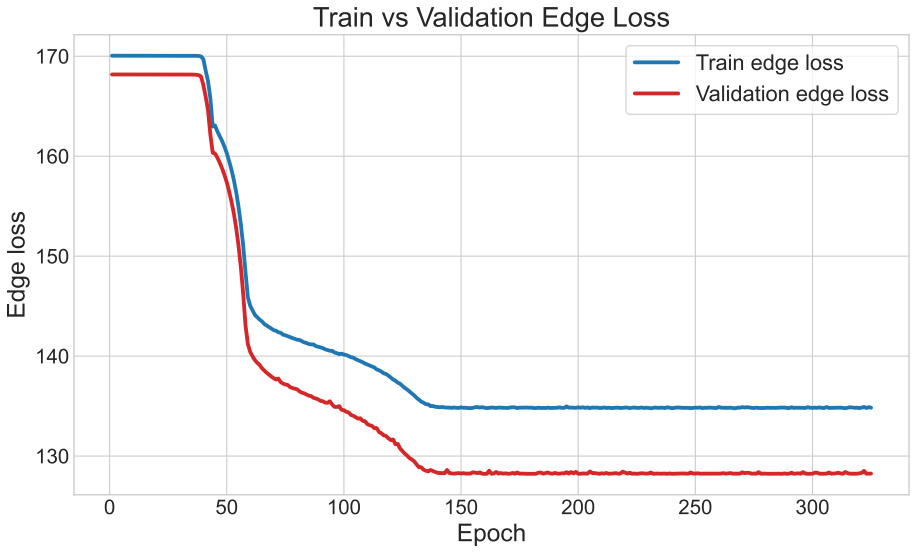

Figure A.7: Training and validation loss of edge loss function for cycle consistency weight $\lambda = 1$

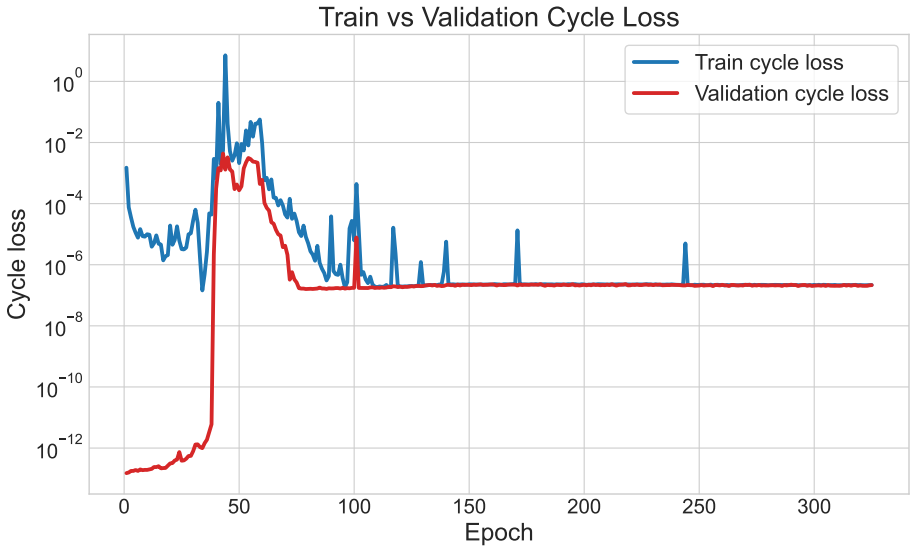

Figure A.8: Training and validation loss of cycle loss function for cycle consistency weight $\lambda = 1$

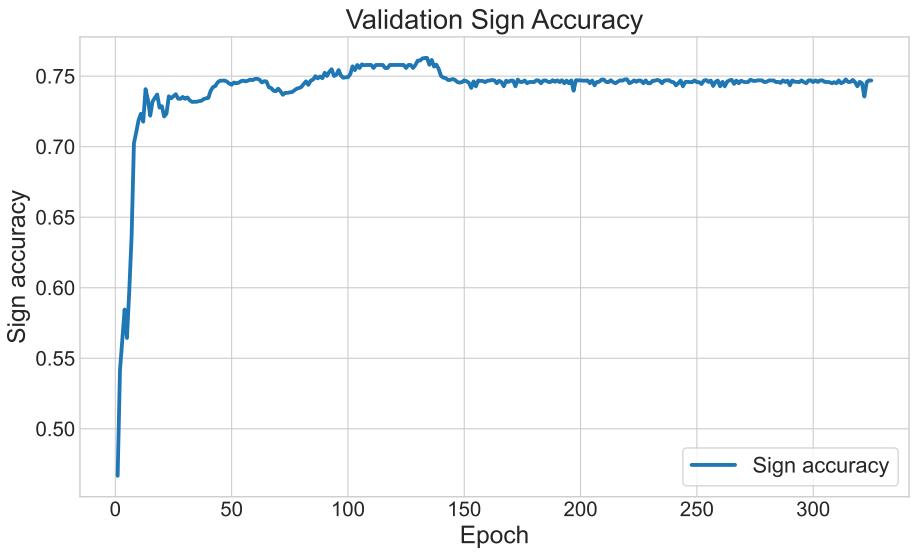

Figure A.9: Validation Sign Accuracy Curve for cycle consistency weight $\lambda = 1$

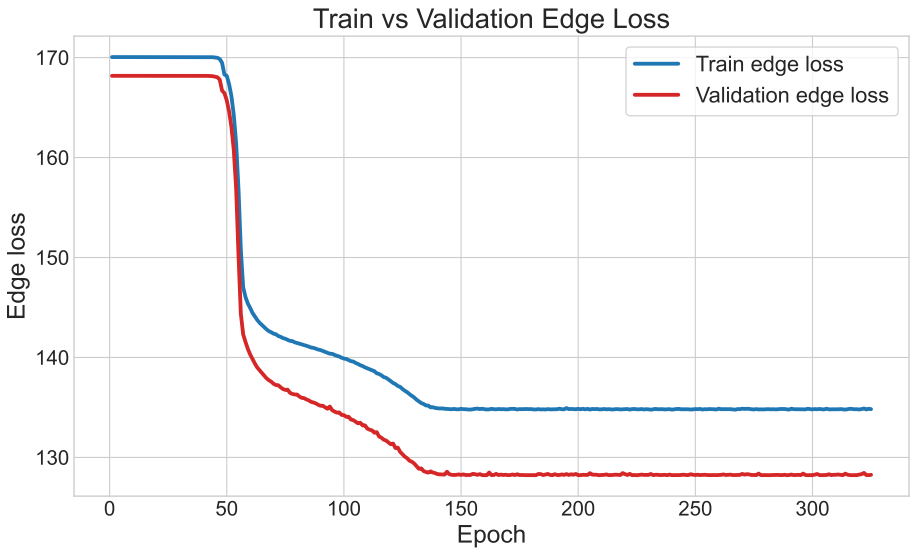

Figure A.10: Training and validation loss of edge loss function for cycle consistency weight $\lambda = 2$

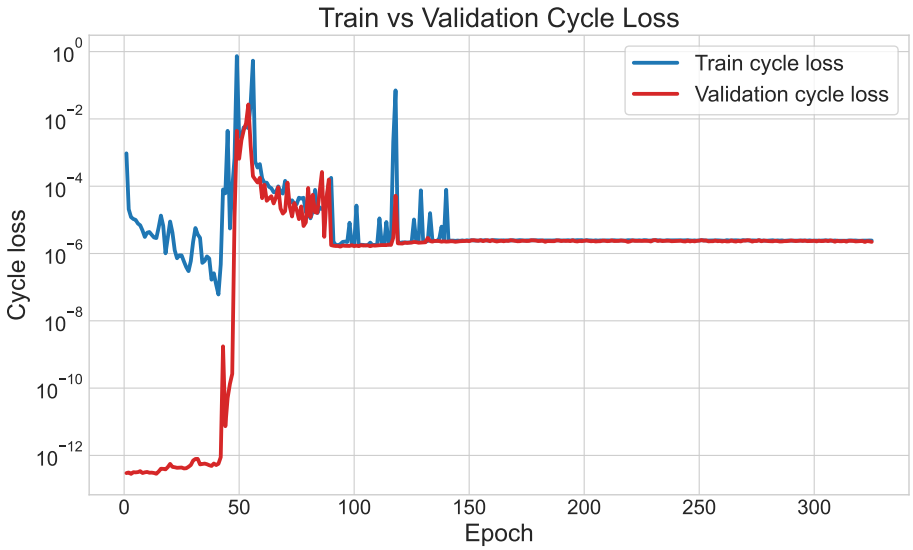

Figure A.11: Training and validation loss of cycle loss function for cycle consistency weight $\lambda = 2$

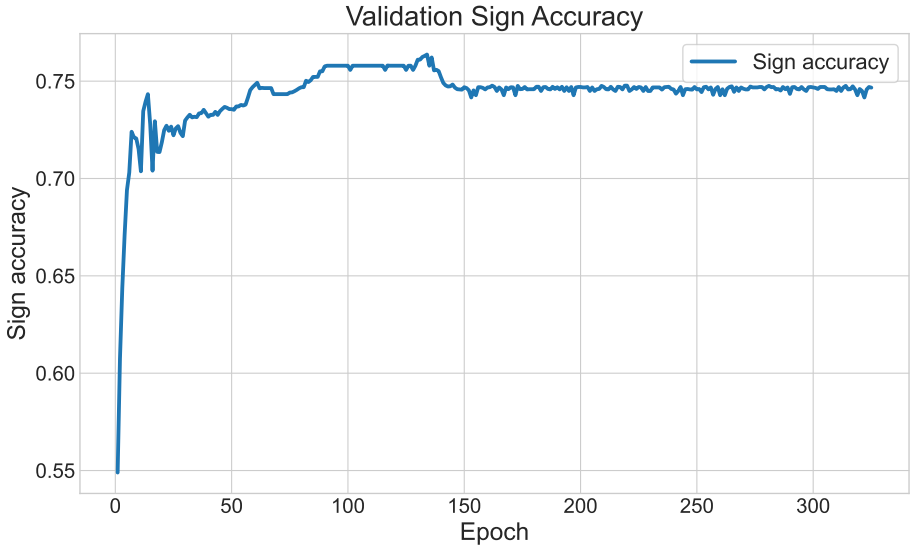

Figure A.12: Validation Sign Accuracy Curve for cycle consistency weight $\lambda = 2$

Reference

[1] Achira Boonrath et al. “Learning-Aided Control of Robotic Tether-Net with Maneuverable Nodes to Capture Large Space Debris”. In: *2024 IEEE International Conference on Robotics and Automation (ICRA)*. 2024, pp. 11737–11743. DOI: 10.1109/ICRA57147.2024.10610721.

[2] Luca Carlone and Carlo Pinciroli. “Robot Co-design: Beyond the Monotone Case”. In: *2019 International Conference on Robotics and Automation (ICRA)*. Montreal, QC, Canada: IEEE Press, 2019, pp. 3024–3030. DOI: 10.1109/ICRA.2019.8793926. URL: https://doi.org/10.1109/ICRA.2019.8793926.

[3] Felix Chalumeau et al. “Combinatorial optimization with policy adaptation using latent space search”. In: *Proceedings of the 37th International Conference on Neural Information Processing Systems*. NIPS ’23. New Orleans, LA, USA: Curran Associates Inc., 2023.

[4] Souma Chowdhury et al. “A mixed-discrete particle swarm optimization algorithm with explicit diversity-preservation”. In: *Structural and Multidisciplinary Optimization* 47 (2013), pp. 367–388.

[5] Mathieu Desbrun et al. *Discrete Exterior Calculus*. 2005. arXiv: math/0508341 [math.DG]. URL: https://arxiv.org/abs/math/0508341.

[6] Thomas G. Dietterich. “State abstraction in MAXQ hierarchical reinforcement learning”. In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*. NIPS’99. Denver, CO: MIT Press, 1999, pp. 994–1000.

[7] Enrique Fernández-González, Erez Karpas, and Brian C. Williams. “Mixed discrete-continuous heuristic generative planning based on flow tubes”. In: *Proceedings of the 24th International Conference on Artificial Intelligence*. IJCAI’15. Buenos Aires, Argentina: AAAI Press, 2015, pp. 1565–1572. ISBN: 9781577357384.

[8] Panfeng Huang et al. “Dynamics and configuration control of the Maneuvering-Net Space Robot System”. In: *Advances in Space Research* 55.4 (2015), pp. 1004–1014. ISSN: 0273-1177. DOI: https://doi.org/10.1016/j.asr.2014.11.

009. URL: https://www.sciencedirect.com/science/article/pii/S0273117714007121.

[9] Roshni Anna Jacob et al. "Real-time outage management in active distribution networks using reinforcement learning over graphs". In: *Nature Communications* 15.1 (June 2024), p. 4766. ISSN: 2041-1723.

[10] Tapio Westerlund Jan Kronqvist Andreas Lundell. *Convex MINLP test problems with non-separable nonlinear functions*. Accessed: August 15, 2017. URL: %7Bhttps://www.minlplib.org/docs/cvxnonsep.pdf%7D.

[11] Eric Jang, Shixiang Gu, and Ben Poole. *Categorical Reparameterization with Gumbel-Softmax*. 2017. arXiv: 1611.01144 [stat.ML]. URL: https://arxiv.org/abs/1611.01144.

[12] Xiaoye Jiang et al. "Statistical ranking and combinatorial Hodge theory". In: *Mathematical Programming* 127.1 (Mar. 2011), pp. 203–244. ISSN: 1436-4646. DOI: 10.1007/s10107-010-0419-x. URL: https://doi.org/10.1007/s10107-010-0419-x.

[13] Minsu Kim et al. *Ant Colony Sampling with GFlowNets for Combinatorial Optimization*. 2025. arXiv: 2403.07041 [cs.LG]. URL: https://arxiv.org/abs/2403.07041.

[14] Yann LeCun et al. "A Tutorial on Energy-Based Learning". In: *A Tutorial on Energy-Based Learning*. 2006. URL: https://api.semanticscholar.org/CorpusID:8531544.

[15] Henrique Lemos et al. "Graph Colouring Meets Deep Learning: Effective Graph Neural Network Models for Combinatorial Problems". In: *CoRR* abs/1903.04598 (2019). arXiv: 1903.04598. URL: http://arxiv.org/abs/1903.04598.

[16] Jiaxuan Liang. "Review of Combinatorial Optimization Methods in the Intelligent Era". In: *Proceedings of the 2025 International Conference on Software Engineering and Computer Applications*. SECA '25. Association for Computing Machinery, 2025, pp. 128–134. ISBN: 9798400715136. DOI: 10.1145/3747912.3747933. URL: https://doi.org/10.1145/3747912.3747933.

[17] Liberti, Leo. "Undecidability and hardness in mixed-integer nonlinear programming". In: *RAIRO-Oper. Res.* 53.1 (2019), pp. 81–109. DOI: 10.1051/ro/2018036. URL: https://doi.org/10.1051/ro/2018036.

[18] *Efficient Design Optimization Over Mixed-Combinatorial Spaces Enabled by Graph-Learning*. Vol. Volume 3A: 51st Design Automation Conference (DAC). International Design Engineering Technical Conferences and Computers and Information in

Engineering Conference. Aug. 2025, V03AT03A021. DOI: 10.1115/DETC2025-169719. eprint: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2025/89220/V03AT03A021/7557501/v03at03a021-detc2025-169719.pdf. URL: https://doi.org/10.1115/DETC2025-169719.

[19] Feng Liu et al. "Learning Constrained Corner Node Trajectories of a Tether Net System for Space Debris Capture". In: *AIAA AVIATION 2023 Forum*. AIAA, 2023. DOI: 10.2514/6.2023-3920. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2023-3920. URL: https://arc.aiaa.org/doi/abs/10.2514/6.2023-3920.

[20] Zhihao Liu et al. "Bayesian Optimization over Mixed Type Inputs with Encoding Methods". eng. In: *ADVANCES IN KNOWLEDGE DISCOVERY AND DATA MINING, PAKDD 2023, PT II*. Vol. 13936. Lecture Notes in Computer Science. Switzerland: Springer, 2023, pp. 203–215. ISBN: 3031333764.

[21] Viktoriya Nikitina, Sergejs Rogovs, and Matthias Gerdts. "Piecewise linear value function approximations in nonlinear dynamic scheduling problems with VTOLs". In: *Optimization* 74.5 (2025), pp. 1081–1103. DOI: 10.1080/02331934.2024.2444614. URL: https://doi.org/10.1080/02331934.2024.2444614.

[22] Ivo Nowak. *Relaxation and Decomposition Methods for Mixed Integer Nonlinear Programming (International Series of Numerical Mathematics)*. Birkhauser, 2005. ISBN: 3764372389.

[23] Changyong Oh et al. "Combinatorial Bayesian optimization using the graph cartesian product". In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[24] Shuntaro Okada, Masayuki Ohzeki, and Shinichiro Taguchi. "Efficient partition of integer optimization problems with one-hot encoding". eng. In: *Scientific reports* 9.1 (2019), pp. 13036–12. ISSN: 2045-2322.

[25] Steve Paul and Souma Chowdhury. "Learning to Allocate Time-Bound and Dynamic Tasks to Multiple Robots Using Covariant Attention Neural Networks". In: *Journal of Computing and Information Science in Engineering* 24.9 (Aug. 2024), p. 091005. ISSN: 1530-9827.

[26] Steve Paul, Payam Ghassemi, and Souma Chowdhury. "Learning Scalable Policies over Graphs for Multi-Robot Task Allocation using Capsule Attention Networks". In: *2022 International Conference on Robotics and Automation (ICRA)*. 2022, pp. 8815–8822.

[27] Michael Tynes et al. “Pairwise Difference Regression: A Machine Learning Meta-algorithm for Improved Prediction and Uncertainty Quantification in Chemical Search”. In: *Journal of Chemical Information and Modeling* 61.8 (2021). PMID: 34347460, pp. 3846–3857. DOI: 10.1021/acs.jcim.1c00670. eprint: https://doi.org/10.1021/acs.jcim.1c00670. URL: https://doi.org/10.1021/acs.jcim.1c00670.

[28] Yufeng Xia et al. “Mixed integer programming modeling for the satellite three-dimensional component assignment and layout optimization problem”. In: *Chinese Journal of Aeronautics* (2025), p. 103415. ISSN: 1000-9361.