Forge: Foundational Optimization Representations from Graph Embeddings

Laboratoire d'informatique de l'École polytechnique (LIX), 2026

Serdar Kadıoğlu

Adj. Assoc. Professor, Dept. of Computer Science, Brown

Group VP, AI Center of Excellence, Fidelity Investments

skadio.github.io

Learning & Reasoning

Data Science: ML/DL/NLP/LLMs/etc.

Focuses on **machine learning using historical data** to identify patterns and make predictions. Excels at pattern recognition, classification, and forecasting.

System I - Predictive Models

- Learning from historical data patterns
- Probabilistic predictions and insights
- Ideal for unstructured problems
- Applications include recommendation systems, image recognition, and natural language processing

Decision Science: OR/MIP/CP/SAT/LS/etc.

Focuses on **combinatorial satisfaction and optimization** using logical and mathematical models. Provides provable optimality and explicit reasoning.

System II - Prescriptive Models

- Mathematical and logical formulations
- Provably optimal for deterministic environments
- Perfect for structured problems
- Applications include verification, planning, scheduling, routing, and resource allocation

Learning & Reasoning

Data Scie **AT/LS/etc.**

Focuses on **nd optimization**
to identify p rovides provable
pattern reco

System I –

- Learning
- Probabili ironments
- Ideal for
- Applicati ning,
  image re cation

Evolution of AI Paradigms

Classical AI → **Modern AI** → **Generative AI**

expert systems (1950 – 2010)

supervised learning (2010 – 2020)

self-supervision (2020+)

If-Then Rules

Deep Learning

Large Language Models

Significant increase in investment, research, accessibility & visibility over the recent years

The Evolution of AI Paradigms: From Classical AI to Modern and Generative AI (AAAI YouTube)

Strategic Pillars of Enterprise AI @ Fidelity AI Center

AI Learning from Offline Data

Robust, scalable, reproducible features from structured, unstructured, and semi-structured datasets.

Selective, TextWiser, Seq2Pat

AI for Learning from Online Feedback

Adaptive, real-time, A/B testing systems that continuously learn from user interaction.

Mab2Rec, MABWiser

AI for Decision Making

Large-scale, integrated, (meta) solvers for resource management and optimization.

Forge, Balans, PathFinder

AI for Automated Assistants

Extraction and translation of natural language into downstream tasks and intents for human-computer interaction.

Text2Model, Learn2Zinc, Gala, Ner4Opt, Text2Zinc, iCBS

Responsible AI

Horizontal capabilities for explainability, evaluation, fairness, and bias mitigation across all systems.

Jurity, BoolXAI, BoolXLLM

Strategic Pillars of Enterprise AI @ Fidelity AI Center

AI Learning fro

Robust, scalable, reproc
from structured, unstru
semi-structured datase

Selective, TextWise

AI for Decision Making

e-scale, integrated, (meta)
ers for resource
agement and optimization.

orge, Balans, PathFinder

AI for Automate

Extraction and translatio
tasks and intents for hu

Text2Model, Learn2Zi

inability, evaluation, fairness,
stems.

, BoolXLMM

Received: 26 July 2025 | Revised: 27 August 2025 | Accepted: 2 September 2025

DOI: 10.1002/aaai.70032

SPECIAL TOPIC ARTICLE

Open-source AI at scale: Establishing an enterprise AI strategy through modular frameworks

Serdar Kadıoğlu[1,2]

[1]AI Center of Excellence, Fidelity Investments, Boston, Massachusetts, USA

[2]Department of Computer Science, Brown University, Providence, Rhode Island, USA

Correspondence
Serdar Kadıoğlu
Email: serdark@cs.brown.edu

Abstract

We present a comprehensive enterprise AI strategy developed within the AI Center of Excellence at Fidelity Investments, emphasizing the strategic integration of open-source AI frameworks into scalable, modular, and reproducible enterprise-grade solutions. Our approach is structured around five key pillars: learning from offline data, learning from online feedback, intelligent decision-making, automated assistants, and responsible AI practices. Through a suite of 12 open-source libraries, we demonstrate how modular and interoperable tools can collectively enhance scalability, fairness, and explainability in real-world AI deployments. We further illustrate the impact of this strategy through three enterprise case studies. Finally, we distill a set of best deployment practices to guide organizations in implementing modular, open-source AI strategies at scale.

Open-Source AI at Scale: Establishing an Enterprise AI Strategy [AI Magazine'25]

Mixed-Integer Programming (MIP)

MIP formulates combinatorial optimization problems with both continuous and integer variables.

$$f(x) = \min c^T x \mid Ax \leq b, x \in \mathrm{R}^n, x_j \in \mathrm{Z}\ \forall j \in I$$

LP Relaxation

Obtained by relaxing integer constraints to continuous. Standard bounding procedure.

Integrality Gap

The integrality gap measures the difference between LP relaxation and optimal MIP.

Learning & Reasoning Hybrids in Optimization

Existing ML-OR Integration

- Algorithm configuration procedures
- Variable and constraint selection
- Branching strategies
- Cut selection
- Node selection
- Tree-search configuration

Balans (IJCAI'25), Dash (EJOR'16)
3S (CP"11), ISAC (ECAI'10)

Emerging LLM-OR Integration

- Named entity recognition for optimization
- Natural language interfaces for solvers
- Automated model formulation
- Explanation generation
- Interactive modeling assistants
- Domain-specific optimization co-pilots

Text2Model (AAAI'25), Gala (NeurIPS'25)
Ner4Opt (Constraints'24) , iCBS (MAKE'24)

LLM Copilots for Text-to-Model Translation

A suite of LLM modeling copilots, datasets, fined-tuned models, demos, interactive editor, and online leaderboard for translating natural language text into formal combinatorial constraint models.

Serdar Kadıoğlu[1,2] Karthik Uppuluri[2]

[1] Department of Computer Science, Brown University [2] AI Center of Excellence, Fidelity Investments

Paper | Copilots | Slides | Leaderboard | Interactive Editor | Collections

https://skadio.github.io/text2model

Learning-Based Methods Face Practical Limitations

Heavy Training Dependency

Training is computationally costly and depends on **carefully curating datasets** with desired properties and distributions.

Limited Generalization

Adapting learning-based methods to **new distributions and domains** remains a significant challenge in the field.

Solver Dependency Paradox

Ironically, **training depends on optimization solvers** to create labeled datasets, defeating the purpose of improving solving for hard instances.

Vision: A Foundational Model for Optimization

Schloss Dagstuhl – Leibniz Center for Informatics - Seminar on Data-Driven Combinatorial Optimization (2022)

The Growing Success of ML-Based Approaches

Meta-Learning Frameworks

1

Zhou et al. (2023) propose methods that **generalize across vehicle routing problem variants of different sizes** but remain limited to routing.

Multi-Task Approaches

2

Cai et al. (2025a) introduce frameworks for **backdoor prediction and solver configuration**, trained for each problem.

LLM-Based Methods

3

Li et al. (2025) use evolutionary frameworks to generate **diverse MIP problems but require supervision** and many pre-solved instances.

The Gap: No General-Purpose Optimization Embeddings Exist Today

Problem-Specific

Generalize across tasks but **confined to one problem domain** (e.g., vehicle routing).

Task-Specific

Scale across problem sizes and variants but **limited to one optimization task**.

Supervised

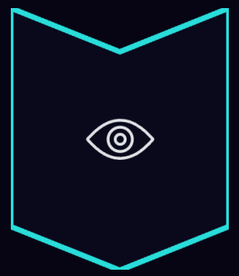
Rely on pre-solved instances and costly data labeling, hindering real-world adaptability.

Forge: Foundational Model for MIP Embeddings

Generate MIP embeddings through pre-training to learn **structural representations** at the **instance level** in an **unsupervised manner**, using a broad distribution of MIP instances **without access to their solutions**.

From Natural Language Processing

We adopt the **concept of a vocabulary to represent the latent space** of optimization problems enabling instance-level representations.



From Computer Vision

We leverage **vector quantization to preserve global information**, addressing the limitations of GNN-based approaches in prior work.

Our Contributions

Foundational Model

Forge captures **both local and global structures**. A single pre-trained model provides embeddings at multiple levels: instance, variable, constraint.

Unsupervised Generalization

Forge embeddings cluster **previously unseen instances** across diverse problem types with high accuracy without any supervision.

Supervised Adaptability

Pre-trained embeddings can be **fine-tuned on diverse downstream tasks** using minimal additional data and low-cost labeling strategies.

Solver Integration

Forge predictions **integrate into Gurobi** demonstrating consistently lower primal gap across tasks, domains, and sizes.

ML Augmentation

Evaluate against state-of-the-art methods, **improving their performance** on large sets of instances they were trained on, **yet unseen by Forge.**

Our Contributions

Foundational Model

Forge captures **both local and global structures**. A single pre-trained model provides embeddings at multiple levels: instance, variable, constraint.

Unsupervised Generalization

Forge embeddings cluster **previously unseen instances** across diverse problem types with high accuracy without any supervision.

Supervised Adaptability

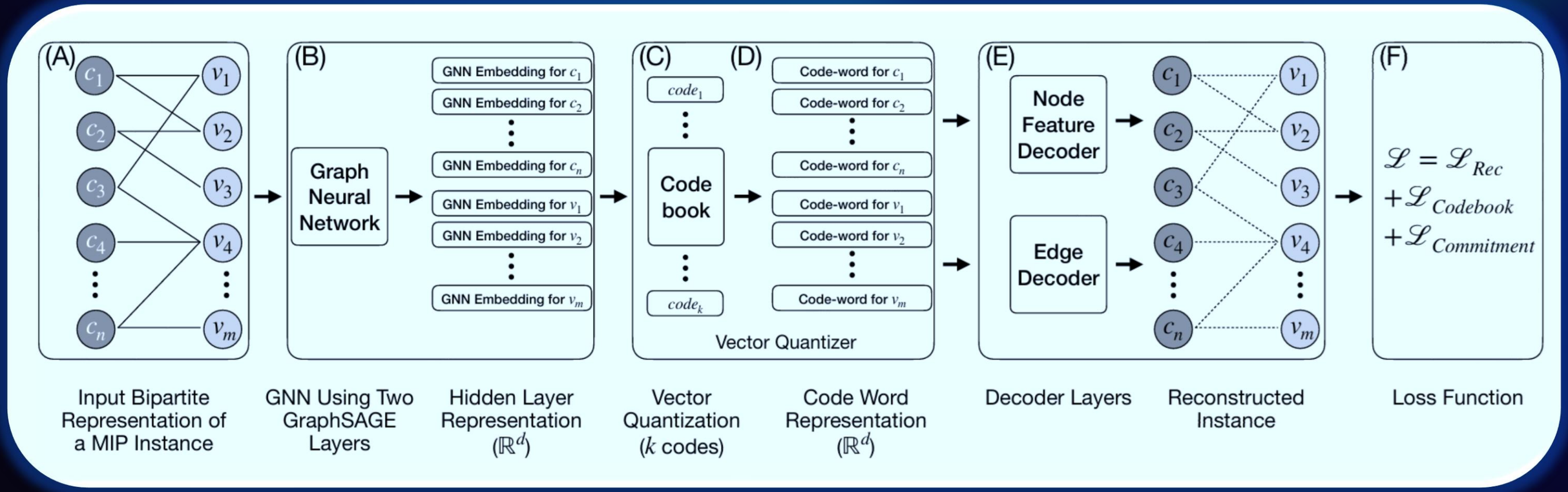
Pre-trained embeddings can be **fine-tuned on diverse downstream tasks** using minimal additional data and low-cost labeling strategies.

Solver Integration

Forge predictions **integrate into Gurobi** demonstrating consistently lower primal gap across tasks, domains, and sizes.

ML Augmentation

Evaluate against state-of-the-art methods, **improving their performance** on large sets of instances they were trained on, **yet unseen by Forge.**



1

High-Level Idea

Forge uses a **vector quantized graph autoencoder** to reconstruct node features and edges. It is pretrained across diverse problems and sizes to learn generic MIP representations without dependency on optimal solutions.

2

Architecture

The architecture combines **bipartite graph**, **GNN embeddings**, **vector quantization with a codebook**, and **reconstruction** objectives to learn structural patterns in an unsupervised manner.

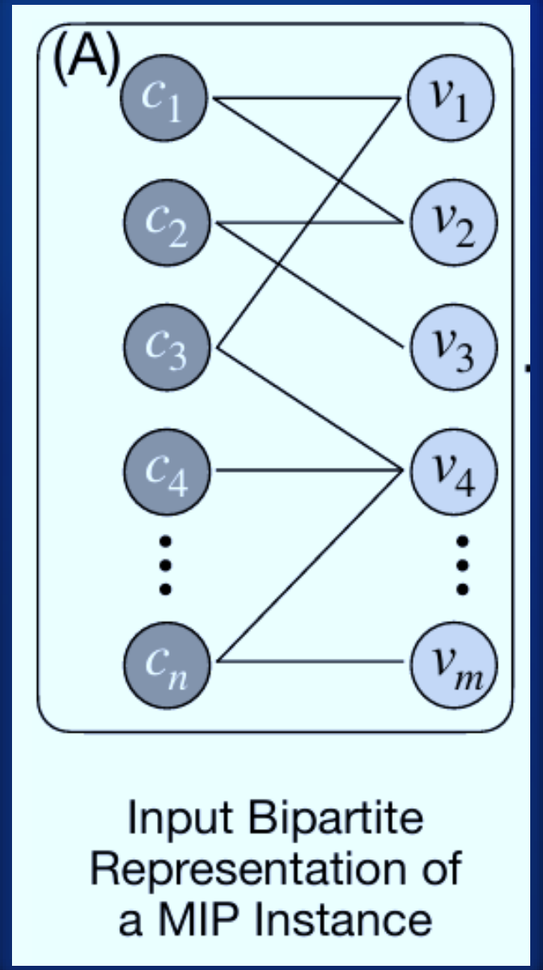
A) MIP-to-Bipartite Representation

Given a MIP instance, we start with its **bipartite representation** (Gasse et. al. 2019) and **node features**. Each node represents a constraint or variable, with edges indicating which variables are part of which constraints.

Node Features Forge uses **only basic properties** of the input instance.

- **Constraint nodes:** 4 features (sense: <,=,> and RHS value)
- **Variable nodes:** 6 features (type: bin, int, cont, ub/lb, objective coeff)
- **Total:** 10-dimensional vector per node

Key Advantage: **No dependency on solving the instance** or accessing internal solver information.



Input Bipartite Representation of a MIP Instance

B) Bipartite to GNN Embeddings

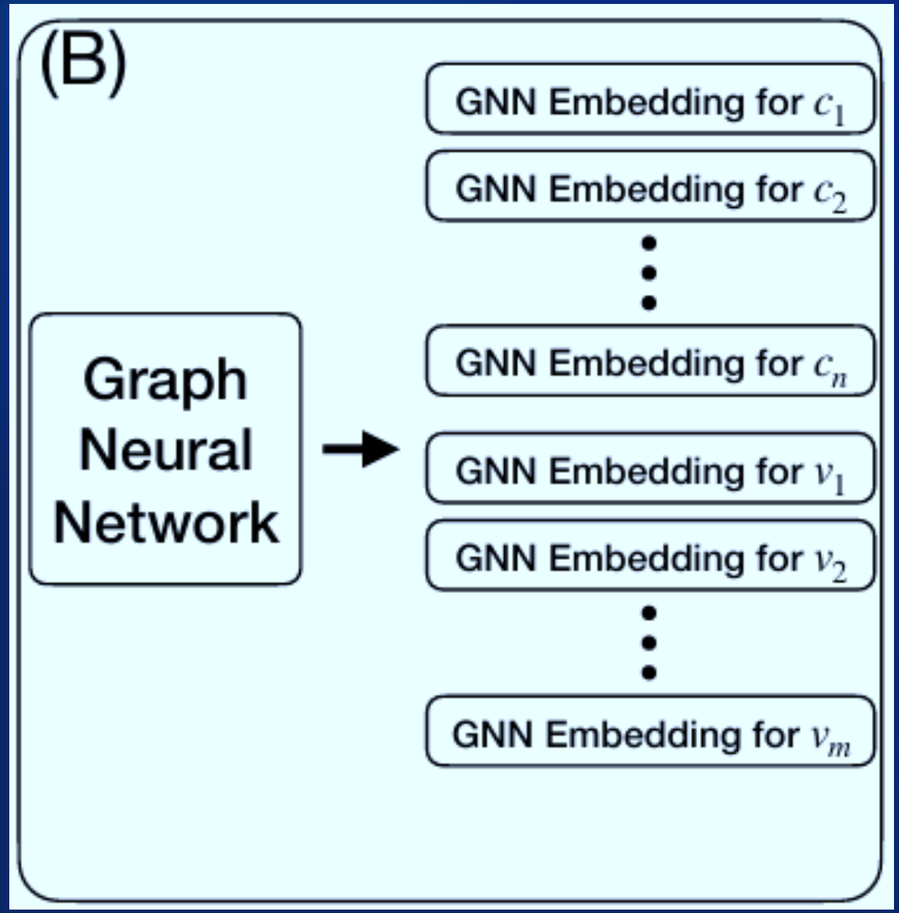
The bipartite graph with 10-dimensional input features is passed into a **Graph Neural Network** to generate embeddings for each constraint and variable node.

GraphSage Layers

Forge uses **two GraphSage layers** that project each input node into a d-dimensional embedding space.

The Locality Challenge

GNNs capture **local variable and constraint-level information**, struggle with global information due to inherent locality bias (Feng et. al. 2025).



C) Vector Quantized Codebook

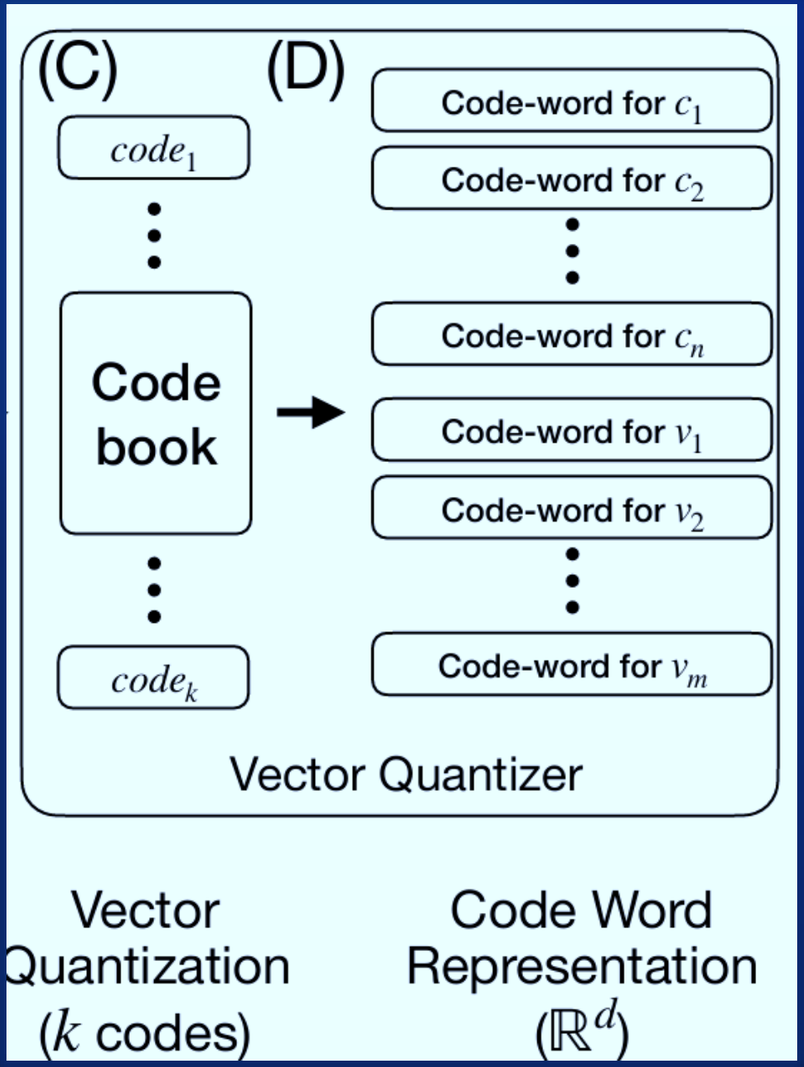
To preserve global structure, we introduce **a vector quantized codebook with k discrete codes**. These codes act as a **'vocabulary'**, akin to language models, across MIP instances of various domains and difficulties.

Inspiration from Computer Vision

The design follows approaches developed **in computer vision** and structure-aware graph tokenizer extensions (Yang et. al. 2024)

Preserving Global Structure

By utilizing discrete codes, Forge **circumvents the over-smoothing issue** and captures the global structure of MIP instances.



D) GNN-to-Codeword Mapping

GNN embeddings are passed into a vector quantizer which consists of a codebook with k codes. The codebook maps each node to a discrete code.

Code Assignment

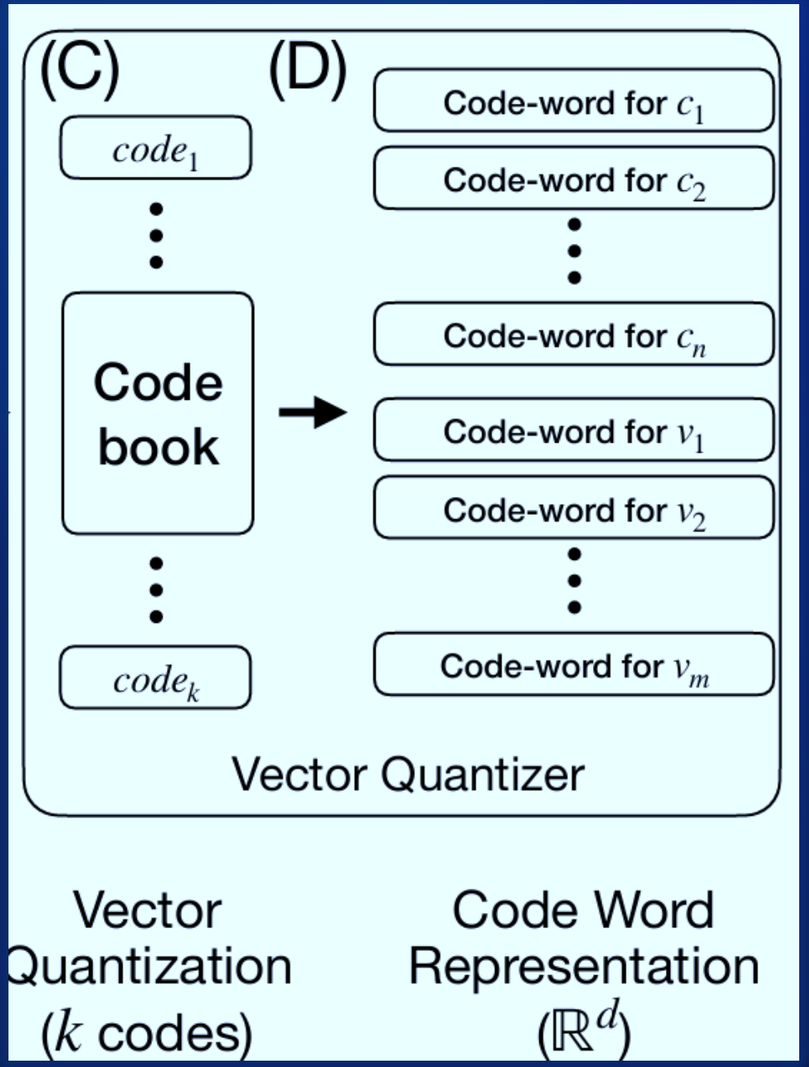
Each node in the bipartite graph is **assigned a discrete code** from the codebook.

Alignment

The codewords are **aligned with the dimensionality** of the hidden GNN layers.

Codeword Mapping

Each code in the **codebook** is then mapped into a d-dimensional **codeword**, producing codeword representations for constraints and variables.



E) Codeword-to-Bipartite Reconstruction

Codewords corresponding to each constraint and variable node are used to **reconstruct the original bipartite representation** of the MIP instance.

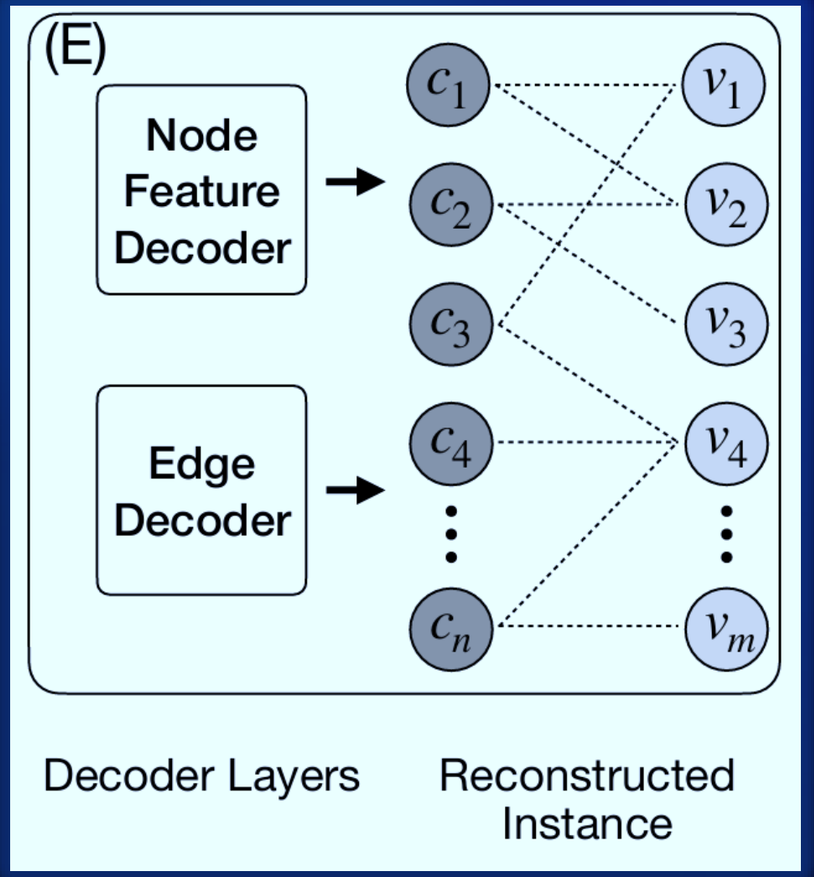
Reconstruction Process

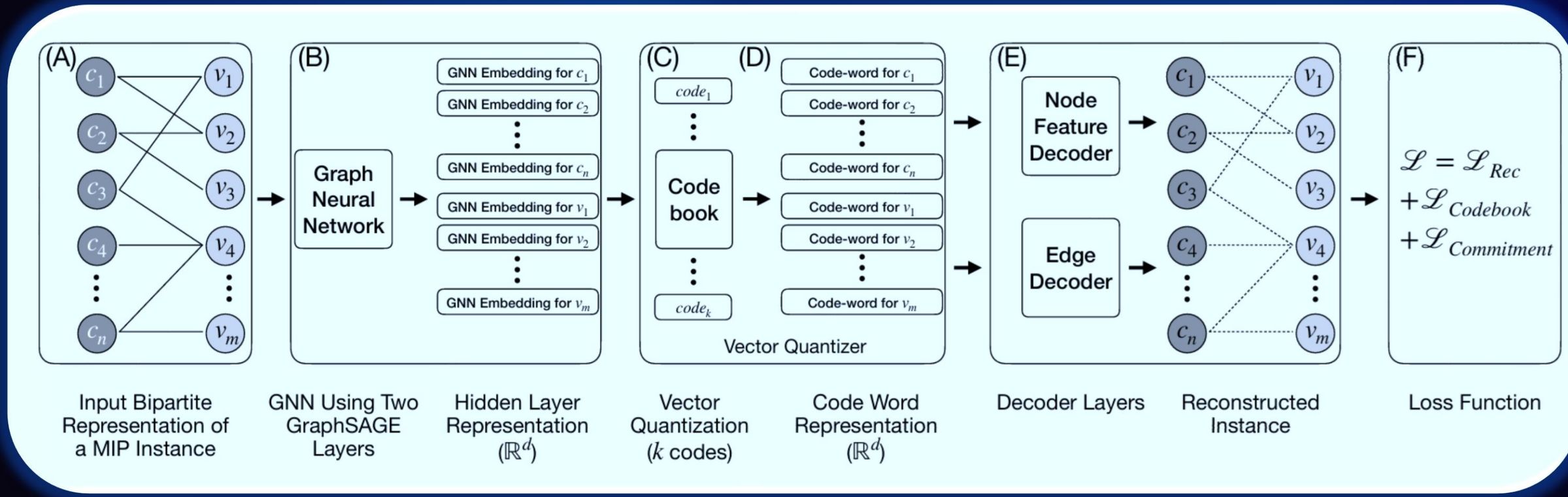
Codewords are passed into **linear decoders** to reconstruct:

- **Node feature decoder**
- **Edge structure decoder**

Unsupervised Learning

By reconstructing the input, Forge **learns from the structure** of MIP instances **without requiring labels or solutions**.




(A)
(B)
Graph Neural Network
GNN Embedding for c_1
GNN Embedding for c_2
GNN Embedding for c_n
GNN Embedding for v_1
GNN Embedding for v_2
GNN Embedding for v_m
(C)
(D)
$code_1$
Code book
$code_k$
Code-word for c_1
Code-word for c_2
Code-word for c_n
Code-word for v_1
Code-word for v_2
Code-word for v_m
Vector Quantizer
(E)
Node Feature Decoder
Edge Decoder
(F)
$\mathcal{L} = \mathcal{L}_{Rec}$
$+\mathcal{L}_{Codebook}$
$+\mathcal{L}_{Commitment}$
Input Bipartite Representation of a MIP Instance
GNN Using Two GraphSAGE Layers
Hidden Layer Representation ($\mathbb{R}^d$)
Vector Quantization (k codes)
Code Word Representation ($\mathbb{R}^d$)
Decoder Layers
Reconstructed Instance
Loss Function

F) Loss Function

The loss function minimizes **edge reconstruction** loss, **node feature reconstruction** loss, and losses related to **vector quantization**.

$$L = L_{Rec} + L_{Codebook} + L_{Commitment}$$

Reconstruction Loss

Measures how well the model reconstructs the **original bipartite graph** structure and node features.

Codebook Loss

Moves codewords closer to node embeddings, similar to k-means clustering.

Commitment Loss

Encourages **node embeddings to commit** to their assigned codewords.

Components of the Loss Function

The loss function components work together to learn meaningful representations through **reconstruction** and **vector quantization.**

1

Codebook Loss Intuition

Can be interpreted as k-means clustering, where **codewords (cluster centroids) move closer to node embeddings** while embeddings are fixed via stop-gradient.

2

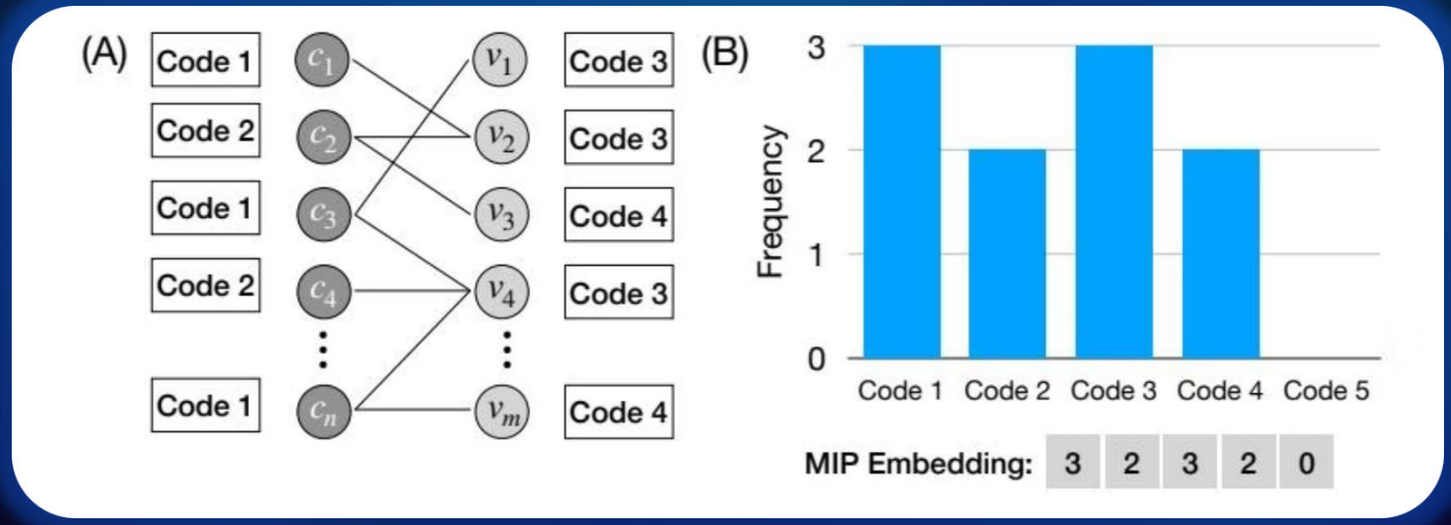
Commitment Loss Intuition

Fixes codewords using stop-gradient and **moves embeddings towards codewords** instead.

3

Balancing Act

The hyperparameter α weighs the importance of the commitment loss, balancing the two objectives.



What does Forge Produce?

Local Representations

Each constraint and variable node is assigned a discrete code mapped to a codeword, providing fine-grained embeddings.

Global Representations

Instance-level embeddings are created from the distribution of codes across all nodes in the MIP instance.

Embedding Structure

Each instance is represented by a vector of size |codebook|, where each value indicates **the frequency of the corresponding code.**

Our Contributions

Foundational Model

Forge captures **both local and global structures**. A single pre-trained model provides embeddings at multiple levels: instance, variable, constraint.

Unsupervised Generalization

Forge embeddings cluster **previously unseen instances** across diverse problem types with high accuracy without any supervision.

Supervised Adaptability

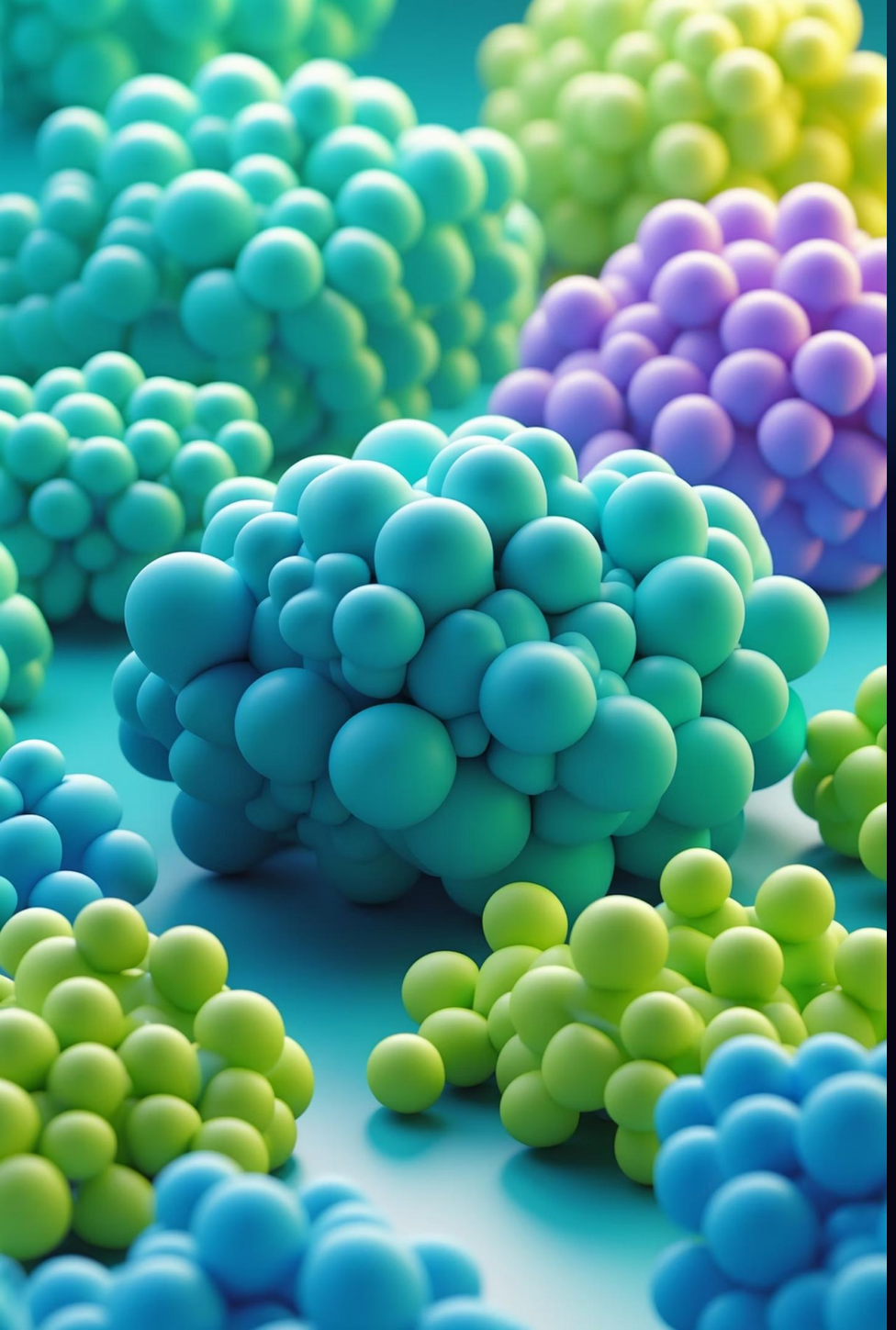
Pre-trained embeddings can be **fine-tuned on diverse downstream tasks** using minimal additional data and low-cost labeling strategies.

Solver Integration

Forge predictions **integrate into Gurobi** demonstrating consistently lower primal gap across tasks, domains, and sizes.

ML Augmentation

Evaluate against state-of-the-art methods, **improving their performance** on large sets of instances they were trained on, **yet unseen by Forge.**



Initial Analysis: Clustering Unseen Instances

We evaluate Forge embeddings on clustering unseen instances across various problem domains, comparing against two baseline approaches.

1. **Training**

   MIPLIB instances and its relaxation

2. **Testing**

   D-MIPLIB instances across 21 domain-difficulty pairs

3. **Evaluation**

   Quantitative (NMI score) and **qualitative** (visualization) analysis

Training Configuration

Training Data

600 instances from MIPLIB, sorted by size to ensure bipartite graphs fit on GPU memory. Generated **two additional instances** per MIPLIB instance by randomly dropping 5% and 10% of constraints.

Total: 1,800 MIP instances

Data Augmentation

Dropping constraints only **relaxes the problem,** providing valid augmented training instances without changing the fundamental structure.

Model Architecture

Two GraphSage layers with **d=1024 dimensions** and a **codebook with k=5000** codes (vocabulary size).

Test Dataset: D-MIPLIB

We evaluate on **1,050 instances from D-MIPLIB** (Weimin et. al. 2024) categorized into **21 domain-difficulty pairs**, covering a broad spectrum of problem types and complexity levels.

Independent Set

MIS easy, medium;
GIS easy to ext-hard

Set Cover

Easy, medium, hard

Vertex Cover

MVC easy, medium, hard

Other Problems

Item Placement, Maritime
Inventory Routing

Combinatorial Auction

Very-easy to very-hard2

Baseline Comparisons

1

Mean Readout

Averages all GNN node embeddings within the trained Forge model. Ablation without vector quantization.

2

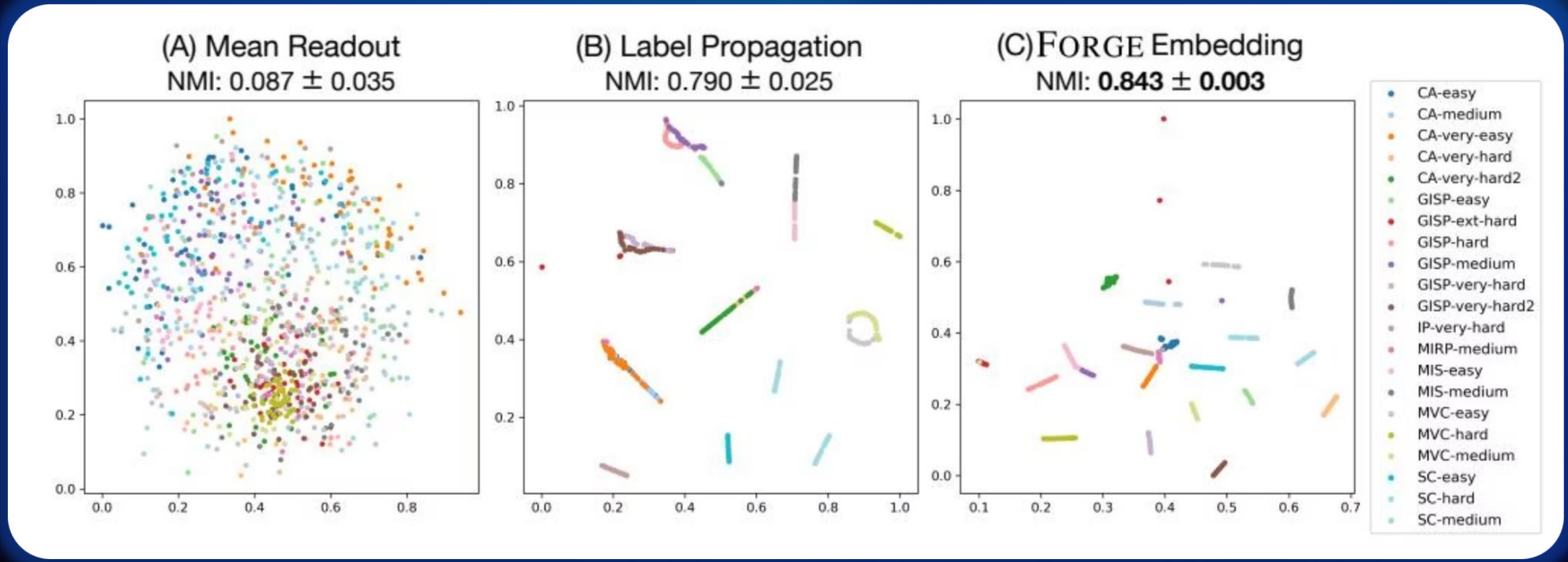
Label Propagation

Two-hop label propagation on the 10-dimensional static node features and averages the resulting node vectors.

3

Forge Embeddings

Uses the **distribution of discrete codes** assigned to constraints and variables as instance-level embeddings.



0.087

Mean Readout

Only slightly better than random (0.047). Suffers from over-smoothing when averaging dense GNN embeddings.

0.790

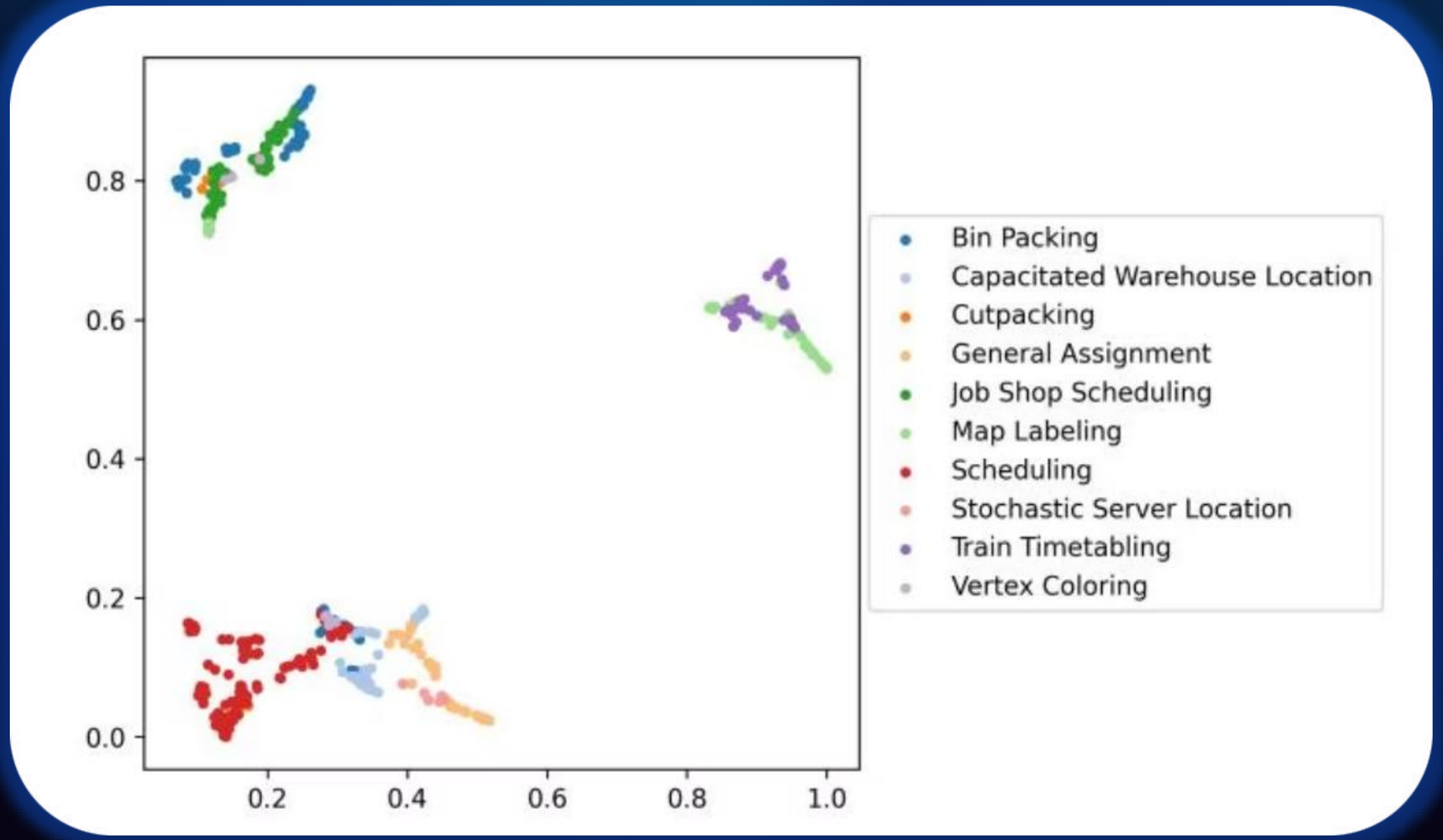
Label Propagation

Better performance operating directly on sparse input features, avoiding over-smoothing.

0.843

Forge Embeddings

Best performance by utilizing distribution of discrete codes, circumventing over-smoothing and capturing global structure.



- We repeat the experiment using our MIPLIB-pretrained Forge to **cluster strIPlib instances**.
- We select 50 instances from each of 10 previously unseen problem types.
- Forge cleanly clusters different problems despite never seeing these instances.
- **Interesting patterns:** Train Timetabling and Map Labeling appearing close to each other, potential TL opportunities.

Set Cover Problem

Find smallest number of subsets that cover all elements. A covering problem with ≥ constraints.

Vertex Cover Problem

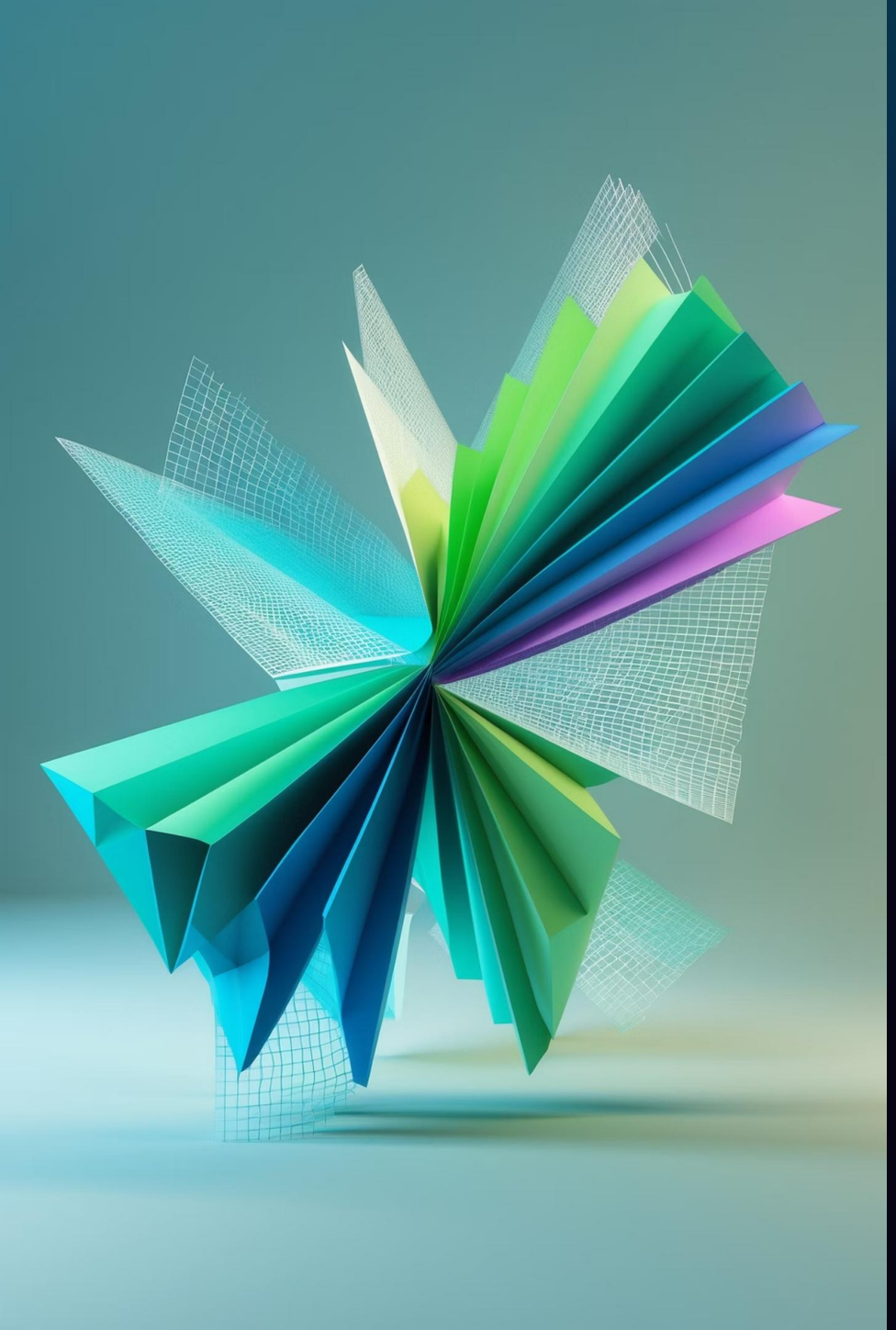
Find smallest set of vertices such that every edge has at least one endpoint. Also, a covering problem.

Bin Packing Problem

Find smallest number of bins that pack all items within capacity. A packing problem with ≤ constraints.

Independent Set Problem

Find largest set of vertices with no adjacent nodes. Also, a packing problem, complementary to Vertex Cover.



Vector Arithmetic in Latent Optimization Space

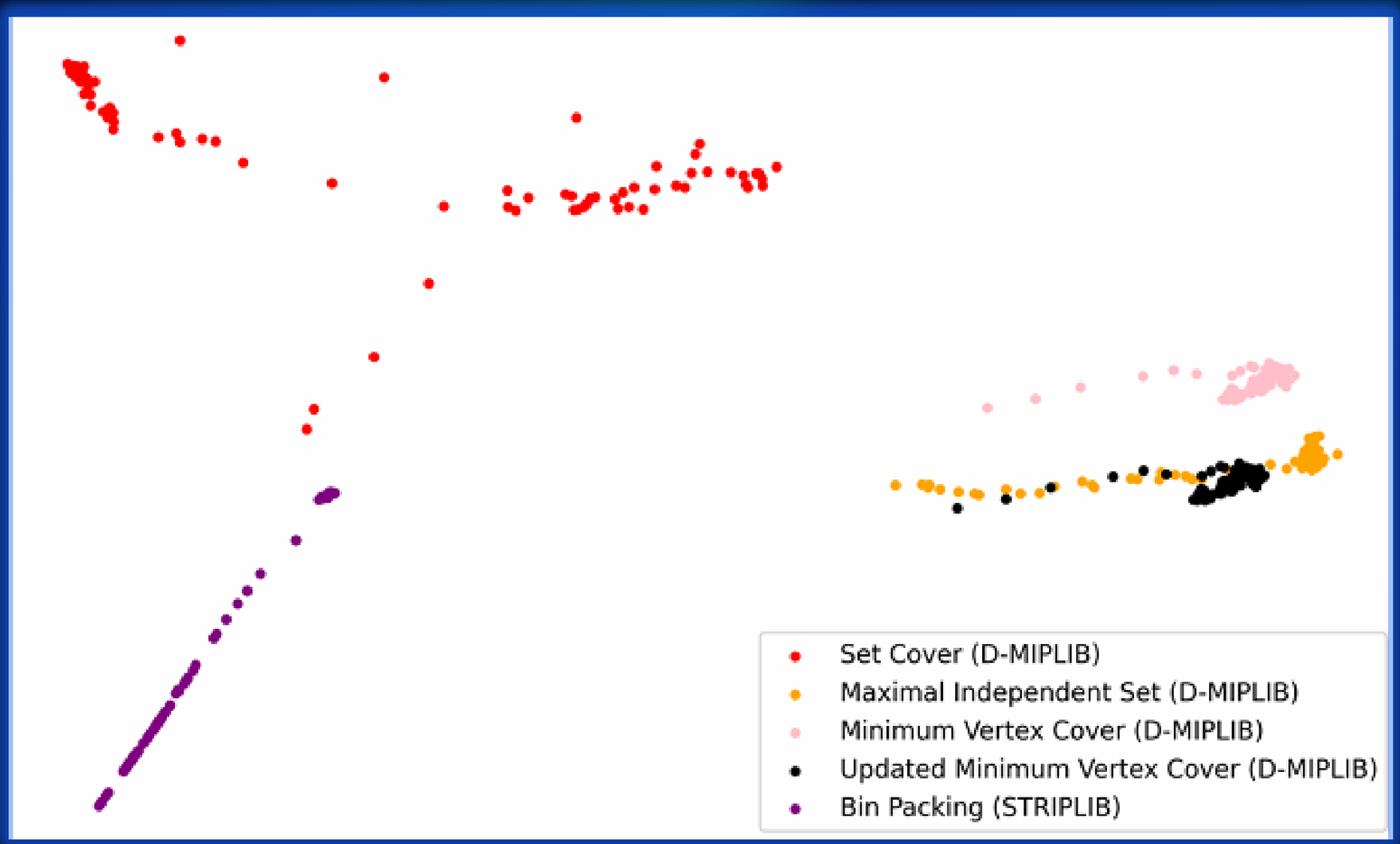
$$VertexCover - SetCover + BinPacking \approx IndependentSet$$

Experimental Setup

- **Fixed graph size**: 1,000 vertices
- 50 random instances per problem
- **Controlled difficulty:** solvable within 60s

Methodology

Compute **mean embeddings** for each problem type, calculate difference vector between covering and packing, apply transformation to Vertex Cover instances.



Updated **Vertex Cover instances** (shown in black) move closer to **Independent Set instances** after applying the transformation. This validates that **meaningful semantic directions** exist in the optimization embedding space.

Our Contributions

Foundational Model

Forge captures **both local and global structures**. A single pre-trained model provides embeddings at multiple levels: instance, variable, constraint.

Unsupervised Generalization

Forge embeddings cluster **previously unseen instances** across diverse problem types with high accuracy without any supervision.

Supervised Adaptability

Pre-trained embeddings can be **fine-tuned on diverse downstream tasks** using minimal additional data and low-cost labeling strategies.

Solver Integration

Forge predictions **integrate into Gurobi** demonstrating consistently lower primal gap across tasks, domains, and sizes.

ML Augmentation

Evaluate against state-of-the-art methods, **improving their performance** on large sets of instances they were trained on, **yet unseen by Forge.**

Supervised Experiments

We now shift to **supervised evaluations** to demonstrate Forge's utility in **improving MIP solving** across fundamentally different downstream tasks.

Task Selection Criteria

Tasks must **provide utility** for MIP solving, enable **fair comparison**, be **radically different** from each other, and be **solver-agnostic**

Same Pre-trained Model

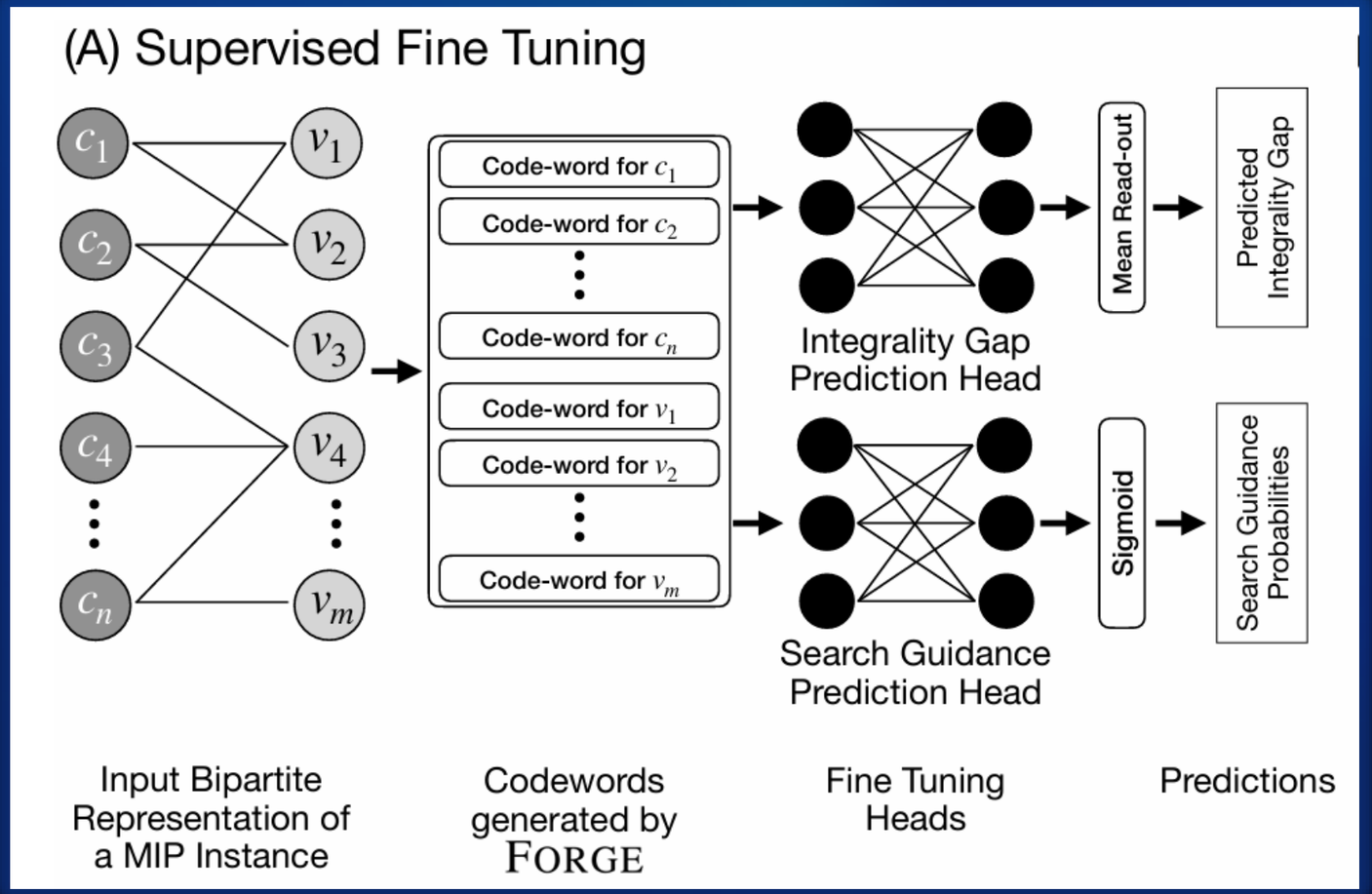
We use the **same pre-trained Forge model** for all tasks, problems, sizes, validating general applicability

Task I: Integrality Gap Prediction

Used to **generate a pseudo-cut** added to the original problem formulation to tighten its bound at the beginning of the search.

Task II: Variable Guidance

Used to **provide hints to the solver** during search about which variables are likely to be in the solution.



Fine-Tuning Philosophy

The same pre-trained model then fine-tuned on a small and **cheaply labeled data** to learn prediction heads for completely different tasks.

NLP Analogy

Fine-tune prediction heads for **entity extraction** in a specific domain (e.g., finance) using a small set of labels. Revive the success of foundational models

Training the Foundational Forge Model

Expanded Dataset

- 1,800 **MIPLIB** instances
- 1,050 **D-MIPLIB** instances
- Total: **2,850 MIP** instances

Model Specifications

- **3.25 million parameters**
- Two GraphSage layers
- d = 1,024 dimensions
- k = 5,000 codebook size

Task I: Predicting the Integrality Gap

There is **no magic constant** that one could always use heuristically, wide distribution %5-95%
Makes integrality gap prediction a deliberate learning task.

1

Challenge

Predict gap without solving to optimality

2

Application

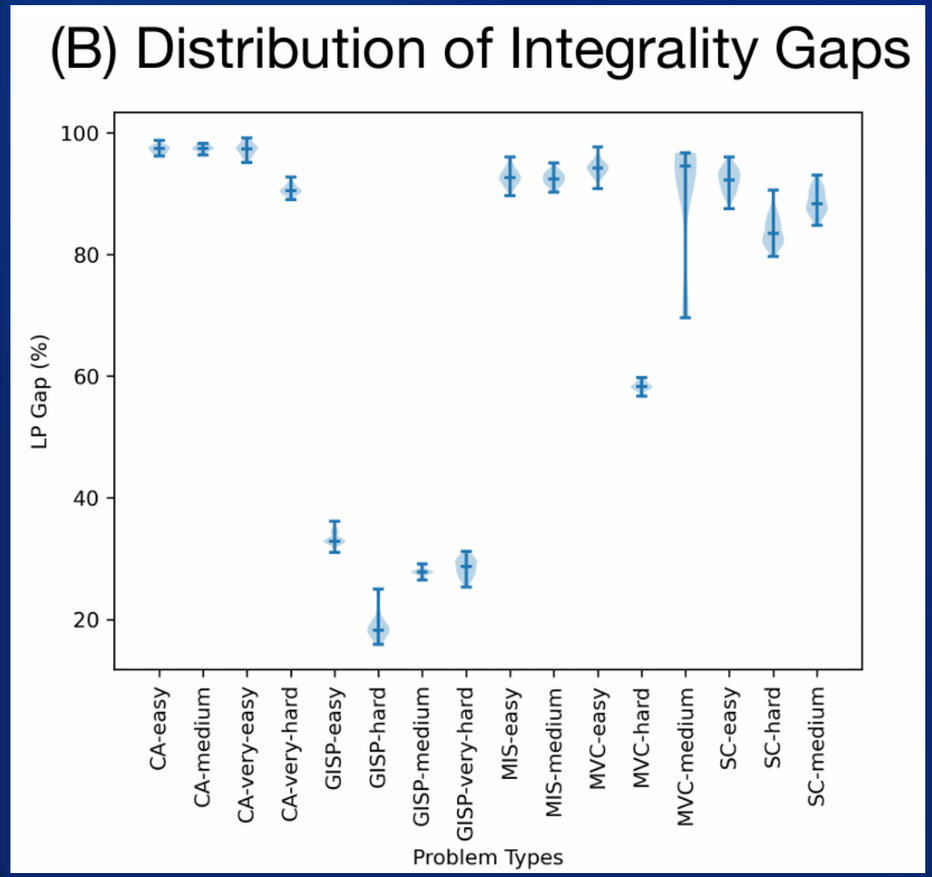
Generate pseudo-cut as additional constrain

3

Risk & Benefit

Incorrect prediction may over/underestimate objective

Pseudo-cuts can speed up solving by tightening bounds



Integrality Gap: Training Setup

We train on **450 instances** from CA, SC, and GIS problems with varying difficulty levels.
This is **considerably smaller** than the pre-training dataset (2850 instances)

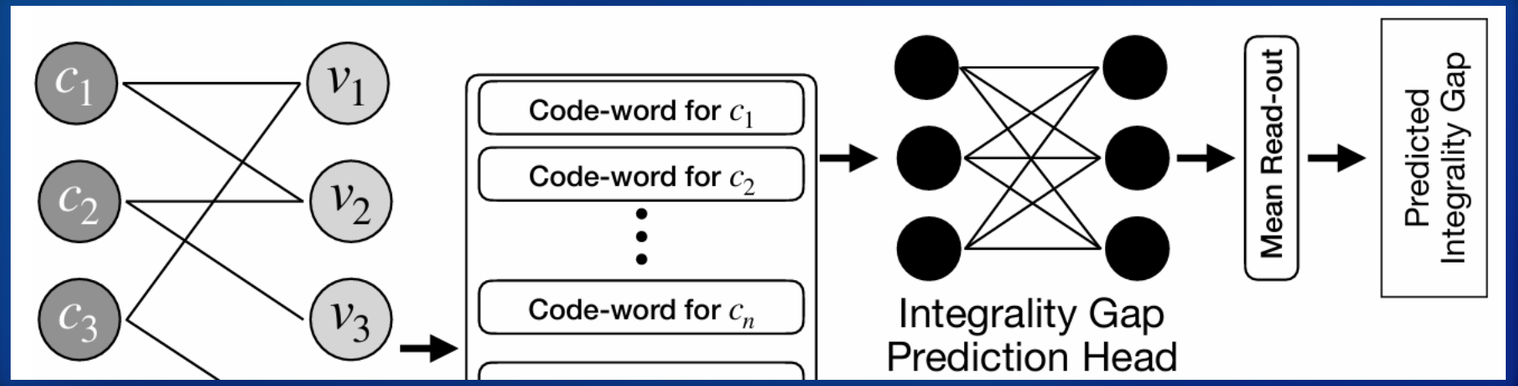
Training Data

- CA: very-easy, easy, medium
- SCP: easy, medium, hard
- GISP: easy, medium, hard
- 50 instances per category

Label Collection

Each instance solved with Gurobi using 120s. Label is the ratio between integer solution at timeout and LP relaxation. Conservative label strategy **does not require solving to optimality.**

Fine-Tuning



1

Prediction Head

Dense layer added to pre-trained Forge model,
takes codewords as input and outputs a real number using mean readout

2

Training Objective

Regression task trained with mean error loss in an end-to-end manner

3

Minimal Data

Only 450 labeled instances needed, with **cheap labeling strategy** that doesn't require optimality

Integrality Gap: Test Setup

Evaluate on **50 very-hard instances** each of CA, SC, GIS, and MVC.
Fine-tuning did not include 'very hard' category, and **MVC is entirely unseen.**

1

Prediction

Forge predicts integrality gap for each test instance

2

Pseudo-Cut Generation

Adjust initial LP relaxation objective based on prediction

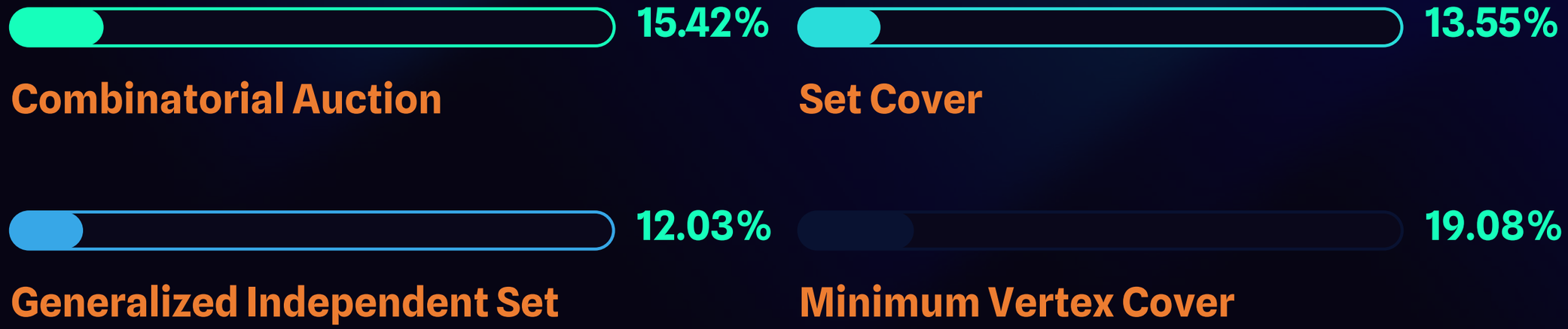
3

Integration

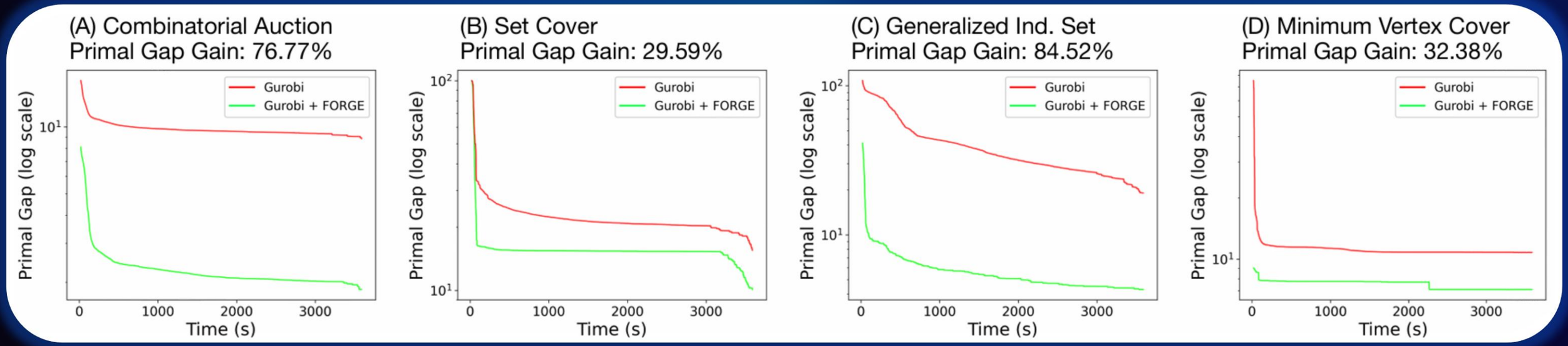
Add pseudo-cut as additional constraint to original formulation

Integrality Gap: Prediction Accuracy

We measure the deviation in the **mean absolute error** between the known integrality gap and the gap predicted by Forge on very-hard test instances.



Ablation: Training from scratch without pre-trained Forge **worsens error by ~33%** on average, highlighting the importance of unsupervised pre-training.



Comparison with Gurobi Solver

We compare the commercial Gurobi solver on **very-hard instances** with and without our predicted pseudo-cut. Each subplot shows the primal gap (lower is better) averaged over 50 instances with 3600s time limit.

Without exception, across all problem types, the use of pseudo-cuts generated by Forge consistently results in better primal gaps. **The performance gains reach up to 85%.**

Take Aways from Solver Comparison

Early Improvements

The solver improves the gap early in the search, and our pseudo-cuts make these **gains immediately more pronounced**.

Minimal Training Required

Fine-tuning **only needed 50 instances** per problem type, **not even including MVC** in fine-tuning.

No Optimality Dependency

Label collection had **no dependency on optimal solutions**, making the approach practical and scalable.

Consistent Performance

Forge consistently improves solver performance across **all tested problem types and difficulty levels**.

Comparison with State-of-the-Art ML

Forge

- **Used as-is without additional training**
- Tested on 17,500 previously **unseen** instances
- From 400 generated problem types

18.63%

Li et al. (2025)

- **Trained on massive dataset**
- Problem-specific training

20.14%

Forge achieves **lower error rates** despite being tested on entirely unseen problem types without additional training.

Task II: Guiding the Search

The previous task evaluates the global representations at instance level. **Next task evaluates the local representations**, specifically the variable embeddings for search guidance.

The Approach

Fine-tune on a smaller dataset with a labeling strategy that does not depend on solving to optimality.

The Goal

Provide variable hints to the Gurobi solver to guide the search toward promising solutions.

Training & Labeling for Search Guidance

We collect 100 instances from CA (easy, medium), SC (easy, medium, hard) and GIS (easy, medium) for a total of 700 training instances.

1

Solution Pool Generation

Each instance is solved using Gurobi to find a pool of **five feasible solutions** within five minutes. Optimality is not required for labeling.

2

Variable Labeling

Variables that never appear in any solution are marked as **'negative**'. Variables that appear in a solution at least once are marked as **'positive'**.

3

Triplet Construction

Variables that appear **in the same number of solutions** are treated as 'positive' and **'anchor**' pairs for triplet loss.

Supervised Fine-Tuning with Dual Loss

Fine-tune using a combination of binary cross-entropy and triplet loss to learn which variable assignments

Cross-Entropy Loss

We add a **dense prediction head** to pre-trained Forge to predict whether each variable is positive or negative.

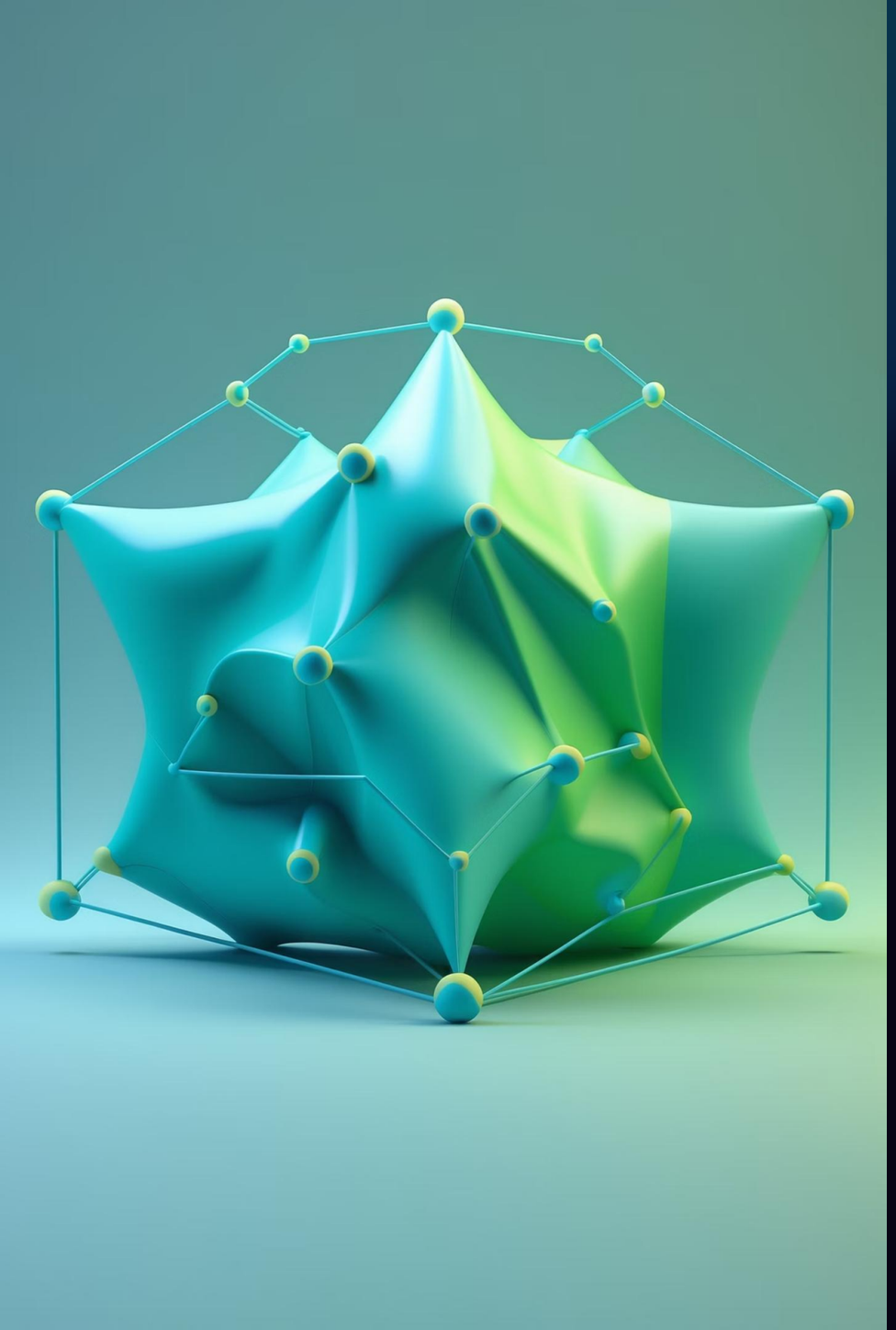
Triplet Loss

Standard triplet loss where variables appearing in the **same number of solutions** are treated as 'positive' and 'anchor' pairs.

Negative Selection

For every positive/anchor pair, we select the negative **variable that is closest to the anchor in the** unsupervised Forge embedding space.

Pre-trained Forge circumvent the challenge of identifying good negatives, as trivial negatives do not help learning.



Search Guidance: Test Setup

We test on 50 medium instances from each of CA, SC, GIS, and MVC. Again, **MVC is unseen** in fine-tuning.

1

Initial Solution

Find feasible solution with Gurobi within 1s
variables serve **as anchors**

2

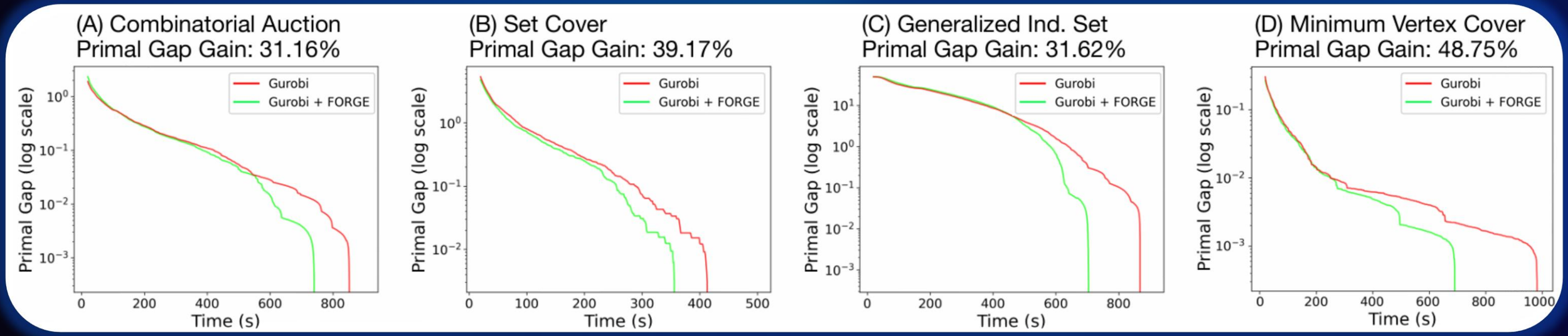
Neighbor Selection

Identify neighbors of positive/negative anchors within fixed radius in embedding space

3

Hint Generation

Top-decile neighbors of positive anchors hinted for inclusion, **bottom-decile neighbors** of negative anchors for exclusion



Comparison with Gurobi Solver

48%

Primal Gap Improvement

Up to 48% improvement in primal gap across tested problems

35%

Speed-Up

Up to 35% faster convergence to optimal solutions

Search Guidance: Augmenting SOTA ML

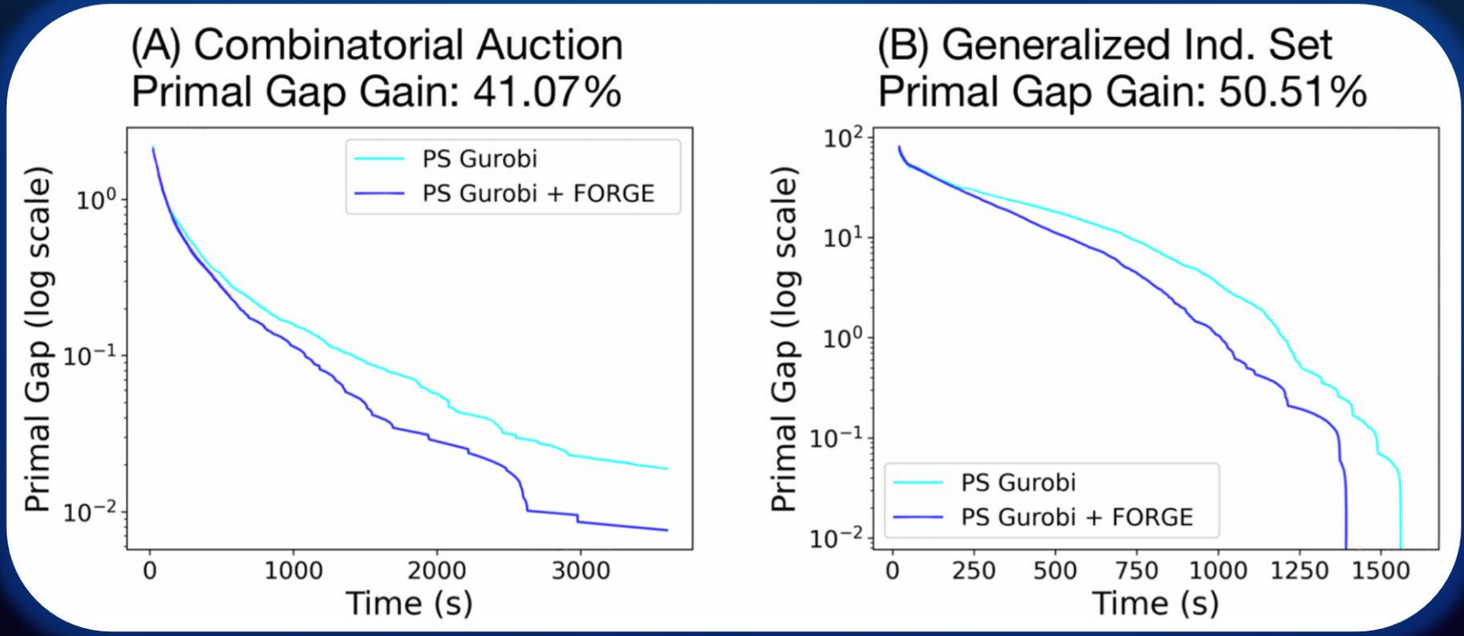
We test Forge's ability to **augment not only MIP solvers but also other ML methods**.
We **concatenate PS-Gurobi** (Han et al., 2023) by Forge embeddings with their variable and node embeddings.

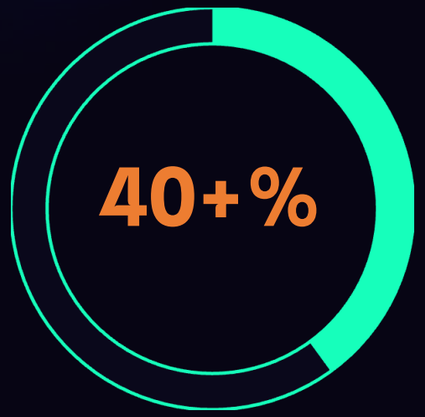
Integration Method

- Use pre-trained Forge embeddings **as-is**
- Apply PCA to reduce to 64 dimensions
- **Concatenate with PS-Gurobi embeddings**

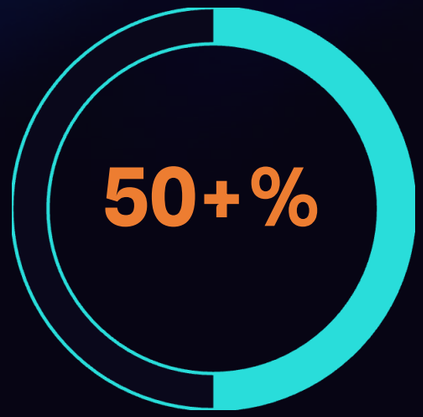
Evaluation

Test on **common subset of problems** used in PS-Gurobi experiments: Combinatorial Auction and Generalized Independent Set





Combinatorial Auction



Generalized Independent Set

Limitations

Scale

Forge is **compact (3.25M parameters trained on ~2.8K instances)**. In principle, it is feasible to train on all publicly available and synthetically generated MIP instances.

Interpretability

The semantics of the learned optimization vocabulary remain unexplored. Preliminary evidence suggests certain codes capture local structure like cliques of variables.

Solver Integration

Current experiments are one-shot. Extending to operate throughout the branch-and-bound tree could **enable tighter integration** for further improvements.

Future Directions

Other Downstream Tasks

Leverage Forge for warm-starts, variable selection, node selection, cut selection, solver configuration, and portfolio construction

Broader Generalization

Extend **beyond optimization** to constraint satisfaction problems and from complete to incomplete search methods

A New Paradigm for Optimization

Novel unsupervised framework for learning structural representations of optimization at multiple levels, without requiring access solvers/labels

1

Unsupervised Learning

Discrete vector quantization captures global structure

2

Foundational Model

Generalizes across diverse tasks, domains, and difficulty levels

3

Measurable Impact

Consistently improves both solvers and ML pipelines

REPRESENTATION LEARNING FOR COMBINATORIAL OPTIMIZATION

Forge: Foundational Optimization Representations from Graph Embeddings

Representational learning for combinatorial optimization. Generate embeddings from MIP instances, pre-train models, and fine-tune for downstream tasks uch as predicting integral gap, search guidance, backdoor prediction, and solver configuration.

Serdar Kadıoğlu[1,2] Zohair Shafi[2,3]

[1] Department of Computer Science, Brown University [2] AI Center of Excellence, Fidelity Investments [3] Northeastern University

Paper | Code | Slides | Models | Dataset

https://skadio.github.io/forge

Strategic Pillars of Enterprise AI @ Fidelity AI Center

AI Learning from Offline Data

Robust, scalable, reproducible features from structured, unstructured, and semi-structured datasets.

Selective, TextWiser, Seq2Pat

AI for Learning from Online Feedback

Adaptive, real-time, A/B testing systems that continuously learn from user interaction.

Mab2Rec, MABWiser

AI for Decision Making

Large-scale, integrated, (meta) solvers for resource management and optimization.

Forge, Balans, PathFinder

AI for Automated Assistants

Extraction and translation of natural language into downstream tasks and intents for human-computer interaction

Text2Model, Learn2Zinc, Gala, Ner4Opt, Text2Zinc, iCBS

Responsible AI

Horizontal capabilities for explainability, evaluation, fairness, and bias mitigation across all systems

Jurity, BoolXAI, BoolXLLM

Open-Source AI at Scale: Establishing an Enterprise AI Strategy [AI Magazine'25]

skadio.github.io