CONREP4CO: CONTRASTIVE REPRESENTATION LEARNING OF COMBINATORIAL OPTIMIZATION INSTANCES ACROSS TYPES

Anonymous authors
Paper under double-blind review

ABSTRACT

Considerable efforts have been devoted to machine learning (ML) for combinatorial optimization (CO) problems, especially on graphs. Compared to the active and well-established research for representation learning of text and vision, etc., it remains under-studied for the representation learning of CO problems, especially across different types. In this paper, we try to fill this gap (especially for NP-complete (NPC) problems, as they, in fact, can be reduced to one another). Our so-called ConRep4CO framework, performs contrastive learning by first transforming CO instances in various original forms into the form of Boolean satisfiability (SAT). This scheme is readily doable, especially for NPC problems, including those practical graph decision problems (GDPs) which are inherently related to their NP-hard optimization versions. Specifically, each positive pair of instances for contrasting consists of an instance in its original form and its corresponding transformed SAT form, while the negative samples are other instances not in correspondence. Extensive experiments on seven GDPs (most of which are NPC) show that ConRep4CO significantly improves the representation quality and generalizability to problem scale. Furthermore, we conduct extensive experiments on NP-hard optimization versions of the GDPs, including MVC, MIS, MC and MDS. The results show that introducing ConRep4CO can yield performance improvements of **61.27%**, **32.20%**, **36.46%**, and **45.29%** in objective value gaps compared to problem-specific baselines, highlighting the potential of ConRep4CO as a unified pre-training paradigm for CO problems.

1 INTRODUCTION

Combinatorial optimization (CO) has been attracting wide interest for its practical importance from logistics [41] to finance [36]. Compared with the vector or matrix-like data, e.g., image, text, and the associated short-range tasks, e.g., classification and regression, the CO problems are inherently more challenging due to their discrete and non-convex nature with complex constraints, which often leads to NP-complete (NPC) or even NP-hard complexity [26].

Despite the recent extensive research on machine learning (ML) for CO [3; 15], there still exist many limitations: compared to the well-developed learning approaches for representation of text [35] and vision [50], the tailored representation learning framework for CO problems remain under-explored. In fact, existing ML4CO literature in technique is mainly tailored to a single problem type, e.g., TSP [43], which may become a bottleneck for their ability in the sense of not leveraging the potential cross-domain learning. This gap is especially pronounced with the fast development of multi-modality joint representation learning out of the CO area, e.g. CLIP [38] for both text and image.

In this paper, we try to fill the above gap by advancing the pre-training representation learning paradigm for CO problems, particularly by selecting various NPC problems, mainly including the so-called graph decision problems (GDPs), as pre-training tasks [1]. The hope is that the model trained jointly on various problem types will exhibit better expressiveness and generalization ability. To

[1]GDPs are the decision versions of general NP-hard CO problems, such as the k-independent set problem corresponding to the maximum independent set (MIS) problem, and encapsulate the core challenges of CO. From the 21 NP-complete problems identified by [26], 10 are GDPs, highlighting their fundamental importance.

achieve our objective, we particularly leverage an important fact: NPC problems can be reduced to one another, and can all be transformed into Boolean satisfiability (SAT), making SAT a common form to bridge different original forms. This also suggests that they inherently share a latent structure worth being exploited for effective representations and ultimately for the goal of high-quality problem solving. Furthermore, there is a strong connection between GDPs and their optimization versions, which are more practical in real-world applications. Most CO problems can be converted into their decision versions by adding a target value. Prior results [25] have also established polynomial-time equivalences between decision and optimization for key CO problems. We select GDPs as pretraining tasks to learn representations, maintaining the potential to leverage the learned representations to enhance general CO problem-solving beyond GDPs.

Specifically, we develop a contrastive pre-training learning paradigm, ConRep4CO, tailored to CO problems beyond the vanilla version for images [9]. The pre-training is performed by contrasting problem instances, where the positive sample is defined as a pair of a vanilla GDP instance with its corresponding SAT form, while the negative sample is a pair that is not in correspondence. Also, a decision loss is applied to guide each model to effectively learn the feature representations of the respective instances and capture the unique characteristics of its assigned problem domain. Extensive experiments are conducted to evaluate the effectiveness of the ConRep4CO paradigm, comprising two parts: **1) representation evaluation**, and **2) enhancement of CO problem-solving**. For 1), since there is no universal metric for CO representation evaluation, in the context of our discussion, we use the accuracy of solving GDPs as a measure. We first assess the representation quality and generalizability to problem scale by solving GDPs on both pre-training identical distribution and more difficult instances. We also evaluate the cross-domain generalizability by the solving performance on unseen GDP domains. For 2), we incorporate ConRep4CO into the training of problem-specific neural solvers for minimum vertex cover (MVC), maximum independent set (MIS), maximum clique (MC), and minimum dominating set (MDS). The neural solvers enhanced by ConRep4CO consistently show significant performance improvements, demonstrating the practical applicability of ConRep4CO beyond GDPs. **The highlights of the paper are as follows.**

1) We try to advance the frontier of representation learning, beyond the classic instance forms, e.g., text/vision, by proposing ConRep4CO, a novel contrastive pre-training paradigm to learn general representations across different CO problems with complex discrete constraints and variables.

2) We leverage the SAT form to build the positive/negative pairs for our carefully designed contrastive learning scheme, based on the fact that GDPs (i.e., NPC problems) can be reduced into the SAT form. A merit is that our contrastive approach is augmentation-free, as CO instance augmentation itself is a notoriously challenging task due to unique problem structures. This is in contrast to the trivial image augmentation as done in contrast to vision problems, which, in fact, is also a bottleneck for directly reusing the contrastive learning approaches in vision to combinatorial tasks.

3) Our method learns the representation across different types beyond a single type. Such a unified paradigm facilitates representation learning through knowledge transfer among problem domains and mutual enhancement. Extensive experiments show that ConRep4CO not only improves representation quality but also significantly enhances problem-solving for various CO problems.

2 Related Work

Machine Learning for CO. The application of machine learning to graph-based CO problems has a rich history, with recent research demonstrating substantial advancements in this domain [27; 3; 32]. Most ML-based approaches for CO follow a two-stage framework: *(1) Graph representation learning*, where graph instances are embedded into low-dimensional vector spaces through graph neural networks (GNNs) [20; 6; 8]; and *(2) The utilization of these learned representations to solve CO problems* [24; 37; 40]. Our ConRep4CO paradigm focuses on enhancing the first stage by proposing a more general training approach. While previous work has largely focused on designing network architectures [28; 19; 46], our approach emphasizes the development of a training paradigm that leverages information from multiple problem types. There are also recent works on training with different types of CO, e.g., GOAL [10], UniCO [34] and MAB-MTL [49]. However, these works are orthogonal to ours as they directly follow the multi-task paradigm without contrastive pre-training.

Graph Contrastive Learning. Many graph contrastive frameworks rely on graph augmentations, which can be broadly categorized into two types: *(1) structural perturbations*, such as node dropping, edge sampling, and graph diffusion [11; 23]; and *(2) feature perturbations*, such as adding noise to node features [21]. These augmentation strategies have demonstrated effectiveness across a range of tasks, from graph-level representations [21; 55] to node-level representations [48; 45]. Our ConRep4CO paradigm moves beyond traditional graph augmentations by contrasting graph instances across multiple problem types. Instead of solely relying on structural and feature perturbations, ConRep4CO leverages the inherent characteristics of different CO problems, enabling the model to capture higher-level characteristics. Note that our approach is augmentation-free. We believe this is a nice property as graph augmentation[2] itself is a notoriously complex problem and much more complex than that on image data, as done in the vanilla contrastive paradigm SimCLR [9], where the augmented positive samples are generated by adding perturbations on the image e.g,. cropping, translating, warping etc.

3 Methodology

We present details of our **Con**trastive **Rep**resentation alignment and learning **for** **C**ombinatorial **O**ptimization (**ConRep4CO**) paradigm. We start by introducing the preliminary background on representations of graph decision problems and SAT in Sec. 3.1. Then, we elaborate on our approach to aligning multiple problem types in Sec. 3.2. Finally, we introduce the overall pipeline and implementation of our ConRep4CO, as well as some important training details in Sec. 3.3.

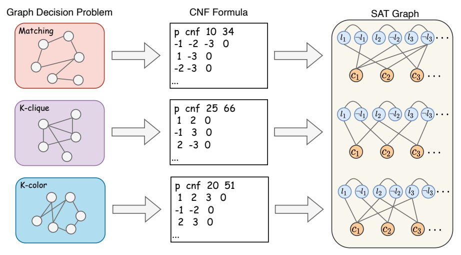
3.1 Preliminaries

Graph decision problem (GDP). As a fundamental computational challenge, its goal is to determine the existence of specific properties within a given graph. These properties can vary, from identifying whether a graph contains a particular substructure, e.g. a clique or cycle, to assessing whether it meets conditions like connectivity or planarity. Graph decision problems are typically the decision versions of general NP-hard CO problems, e.g., the k-independent set problem corresponding to the maximum independent set problem, and the k-vertex cover problem corresponding to the minimum vertex cover problem, making them essential in the context of NPC problems. In particular, ML-based models can be effectively utilized to address GDPs. The objective is to learn a representation of a specific GDP type and use it to predict decisions based on the input graph. These representations can be understood as mappings that translate the structural properties of the input graphs into corresponding decisions, thereby capturing the underlying patterns required for decision-making in GDPs.

SAT problem. A Boolean formula in propositional logic consists of Boolean variables connected by logical operators "and" ($\wedge$), "or" ($\vee$), and "not" ($\neg$). A literal, denoted as l_i, is defined as either a variable or its negation, and a clause c_j is represented as a disjunction of n literals, $\bigvee_{i=1}^{n} l_i$. A Boolean formula is in Conjunctive Normal Form (CNF) if it is expressed as a conjunction of clauses $\bigwedge_{j=1}^{m} c_j$. Given a CNF formula, the Boolean Satisfiability Problem (SAT) aims to determine whether there exists an assignment π of Boolean values to its variables under which the formula evaluates to true. If such an assignment π exists, the formula is called satisfiable, where π is called a satisfying assignment; otherwise, it is unsatisfiable. Graph representations play an important role in analyzing SAT formulas, with four common primary forms [4]: the literal-clause graph (LCG), literal-incidence graph (LIG), variable-clause graph (VCG), and variable-incidence graph (VIG). The LCG is a bipartite graph consisting of two types of nodes—literals and clauses—where an edge between a literal and a clause signifies the occurrence of that literal in the clause. LIG, in contrast, consists solely of literal nodes, with edges representing the co-occurrence of two literals within the same clause. VCG and VIG are derived from LCG and LIG by merging each literal with its negation.

3.2 Modal Alignment

We aim to enhance the learned representations of graph instances across a diverse range of GDPs by incorporating and synthesizing information from multiple GDP types. Specifically, we conceptualize each GDP type as a distinct problem modality. By adopting this multi-modal perspective, we explore the potential for cross-modal information-passing schemes. Note that



```
p cnf 10 34
-1 -2 -3 0
 1 -3 0
-2 -3 0
...
```

```
p cnf 25 66
 1 2 0
-1 3 0
 2 -3 0
...
```

```
p cnf 20 51
 1 2 3 0
-1 -2 0
 2 3 0
...
```

Figure 1: Transformation process from various GDP instances to the unified LCG representation of SAT.

[2]The CO instance can often be represented as a certain graph, e.g., a bipartite graph in [14].

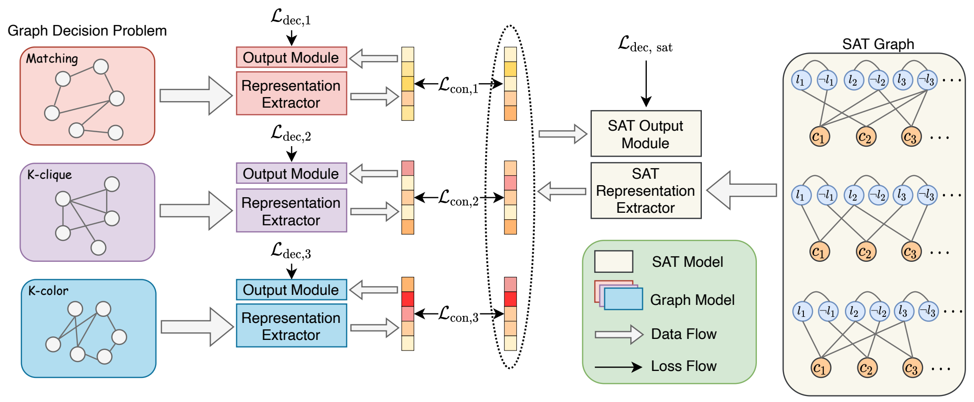

Graph Decision Problem
Matching
K-clique
K-color
Output Module
Representation Extractor
$\mathcal{L}_{\text{dec},1}$
$\mathcal{L}_{\text{dec},2}$
$\mathcal{L}_{\text{dec},3}$
$\mathcal{L}_{\text{con},1}$
$\mathcal{L}_{\text{con},2}$
$\mathcal{L}_{\text{con},3}$
$\mathcal{L}_{\text{dec, sat}}$
SAT Output Module
SAT Representation Extractor
SAT Graph
SAT Model
Graph Model
Data Flow
Loss Flow

Figure 2: Overview of ConRep4CO with the proposed contrastive learning scheme. Given instances from multiple GDP types and their corresponding SAT graphs, a graph model is trained for each GDP type alongside a SAT model. Each model is composed of a Representation Extractor and an Output Module. The input graphs are processed by the Representation Extractor to generate instance-level representations, which are subsequently fed into the Output Module to produce the final decisions for each instance. The decision loss is applied individually to each model, while the contrastive loss is applied to each graph model. All contrastive losses are applied to the SAT model.

the term ‘modality’ is not strictly defined. We hope to express that the problems represent different forms of a higher-level underlying difficulty and share a common underlying structure.

Significant challenges arise due to the inherent disparities and structural gaps between different GDP types, often exhibiting varying graph topologies and problem characteristics. To fill this gap, we propose introducing SAT as a unified intermediary modality. The core concept involves transforming each GDP instance into its corresponding CNF formula, effectively converting it into a SAT instance. Once transformed, we construct a SAT-based graph representation for each instance, ensuring that all GDP instances, regardless of their original modalities, are standardized into an equivalent SAT graph representation. This transformation allows for uniform modeling across disparate problem types. Fig. 1 clarifies our approach to modal transformation.

After this transformation, we leverage contrastive learning to align the different modalities. Specifically, each GDP instance and its corresponding SAT instance form a positive pair to train both the SAT and graph models, while SAT instances derived from other GDP instances within the same GDP type serve as negative samples for the graph model. Similarly, other GDP instances within the same type serve as negative samples for the SAT model. Fig. 3 illustrates the contrastive learning process. The SAT modality, in turn, aligns with all other modalities. The goal is to facilitate information transfer across GDP modalities while preserving the distinct characteristics of each problem type.

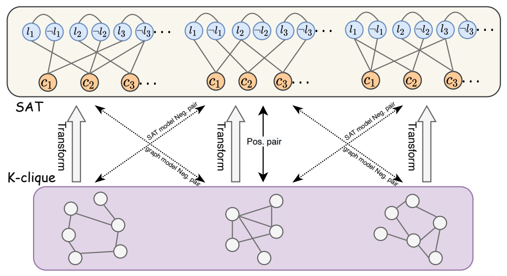


Figure 3: An example of the contrastive learning process, where ’Pos. pair’ and ’Neg. pair’ refer to positive and negative pairs, respectively. A similar process applies to other modalities with SAT.

3.3 ConRep4CO Paradigm

3.3.1 Overview

In this section, we provide a detailed introduction to ConRep4CO. Fig. 2 exhibits an overview.

Consider n types of GDPs, denoted as $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_n$, along with n corresponding graph sets $\mathbf{G}_1, \mathbf{G}_2, \ldots, \mathbf{G}_n$. For simplicity, assume that each graph set $\mathbf{G}_i$ contains m graphs, i.e., $\mathbf{G}_i = \{\mathcal{G}_i^1, \mathcal{G}_i^2, \ldots, \mathcal{G}_i^m\}$, for $i = 1, 2, \ldots, n$. The objective is to solve problem $\mathcal{P}_i$ on graphs in $\mathbf{G}_i$. In total, there are $m \times n$ instances, denoted by $I_i^j = (\mathcal{P}_i, \mathcal{G}_i^j)$, where $i = 1, 2, \ldots, n$ and $j = 1, 2, \ldots, m$.

We first transform each of the $m \times n$ GDP instances into CNF, thereby generating their corresponding SAT graphs, i.e., $(\mathcal{P}_i, \mathcal{G}_i^j) \rightarrow \mathcal{B}_i^j$, where $\mathcal{B}_i^j$ is the constructed SAT graph.

Then, we develop n distinct graph models, $\mathbb{M}_1, \ldots, \mathbb{M}_n$, each for one GDP type, and one unified SAT model $\mathbb{M}_{sat}$ to address the problem space. Both the graph models and the SAT model are structured around two key components: the **Representation Extractor** and the **Output Module**. The Representation Extractor, implemented as a GNN-based network, is responsible for learning and extracting representations from the input graph instances, whether derived from GDP or SAT transformations. The Output Module, implemented as an MLP, then utilizes these learned representations to produce task-specific outputs, thereby enabling the resolution of the given problem.

In training, we jointly train the $(n+1)$ models corresponding to the n GDP modalities along with the SAT modality. The supervision is derived from two parts: the decision loss and the contrastive loss. The decision loss is applied independently to each model, guiding the encoder to learn domain-specific feature representations. Meanwhile, contrastive loss is employed to facilitate feature fusion and message passing across different modalities, enabling the models to leverage complementary information from other problem domains. For training sample and label preparation, one cost comes from generating the corresponding SAT form for each instance, which requires a polynomial complexity. While this cost is negligible for computing labels for the training instances.

3.3.2 Loss Function

We introduce the definition and computation of the two key loss functions used in ConRep4CO.

Decision Loss $\mathcal{L}_{\text{dec}}$. It is defined as a binary cross-entropy loss, which can be computed by:

$$\mathcal{L}_{\text{dec}} = \sum_{i \in \text{Batch}} \left\{ -d_i^{\text{gt}} \log(d_i^{\text{out}}) - (1 - d_i^{\text{gt}}) \log(1 - d_i^{\text{out}}) \right\}, \tag{1}$$

where d^{out} denotes the output decision of the models, and d^{gt} refers to the ground truth label. For each model, the decision loss is independently computed and applied.

Contrastive Loss $\mathcal{L}_{\text{con}}$. Here we adopt the classic contrastive objective as widely used in literature [9; 22; 44], to facilitate the alignment between the GDP and SAT modalities. Taking $\mathcal{P}_n$ and the SAT modality as an example, $\mathcal{L}_{\text{con}}$ is formulated as:

$$\mathcal{L}_{\text{con},n} = \sum_{i=1}^{N} \left\{ -\log \frac{\exp(sim(\hat{\mathbf{r}}_n^i, \hat{\mathbf{r}}_{sat}^i)/\tau)}{\sum_{j=1}^{N} \mathbb{I}_{j \neq i} \exp(sim(\hat{\mathbf{r}}_n^i, \hat{\mathbf{r}}_{sat}^j)/\tau)} - \log \frac{\exp(sim(\hat{\mathbf{r}}_n^i, \hat{\mathbf{r}}_{sat}^i)/\tau)}{\sum_{j=1}^{N} \mathbb{I}_{j \neq i} \exp(sim(\hat{\mathbf{r}}_n^j, \hat{\mathbf{r}}_{sat}^i)/\tau)} \right\} \tag{2}$$

where N represents the number of instance pairs in a batch, $\hat{\mathbf{r}}_n^i$ denotes the normalized representation of the i-th instance in the $\mathcal{P}_n$ modality, and $\hat{\mathbf{r}}_{sat}^i$ denotes the normalized representation of the corresponding instance in the SAT modality, derived from the i-th instance of the $\mathcal{P}_n$ modality. The parameter τ is the temperature scalar, and $\mathbb{I}$ is an indicator function. The function $sim(\cdot, \cdot)$ measures the cosine similarity between two representations, defined as $sim(\mathbf{r}_i, \mathbf{r}_j) = \frac{\mathbf{r}_i^\top \mathbf{r}_j}{\|\mathbf{r}_i\| \|\mathbf{r}_j\|}$.

Each GDP modality is trained using the contrastive loss with the SAT modality, allowing independent optimization for each GDP model. In parallel, the SAT model is optimized using the average contrastive losses computed across all GDP modalities, ensuring effective alignment.

3.3.3 Training Details

We adopt a warm start strategy to ensure the models learn robust representations. During the initial training phase, only the decision loss is utilized, while the contrastive loss is temporarily disabled. This phase allows the models to focus on learning meaningful task-specific representations based solely on the decision outcomes. Our insight is to provide a stable foundation for representation learning before introducing the more complex cross-modal alignment enforced by contrastive loss.

After the warm start phase, we introduce the contrastive loss alongside the decision loss. To balance the influence of these two losses, we introduce a parameter β, which controls the relative weight of the decision loss during the joint training phase.

Table 1: GDP solving accuracy (%) with confidence intervals ($\alpha = 0.05$) of the graph models trained on an identical distribution, measuring the quality of learned representations. 'SAT Back.' refers to SAT model backbone, and 'Graph Back.' denotes graph model backbone. The 'Overall' column represents the average accuracy across all datasets.

SAT Back.	Graph Back.	Difficulty	Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
LCG+NeuroSAT	GCN	Easy	Graph Model	77.0±0.2	58.5±0.3	60.3±0.6	86.1±0.2	62.7±0.4	71.2±0.2	63.6±0.4	68.5
			Graph Model+ConRep4CO	**79.3±0.3**	**62.0±0.1**	**67.3±0.2**	**90.2±0.1**	**67.5±0.5**	**71.7±0.3**	**65.4±0.3**	**71.9**
		Medium	Graph Model	63.2±0.5	62.2±0.2	59.9±0.4	79.6±0.2	61.1±0.2	70.6±0.5	63.3±0.4	65.7
			Graph Model+ConRep4CO	**71.3±0.5**	**64.6±0.2**	**63.3±0.3**	**82.2±0.2**	**64.0±0.1**	**72.8±0.4**	**65.7±0.4**	**69.1**
LCG+GCN	GCN	Easy	Graph Model	77.0±0.3	58.5±0.2	60.3±0.4	86.1±0.2	62.7±0.3	**71.2±0.2**	63.6±0.3	68.5
			Graph Model+ConRep4CO	**79.3±0.2**	**61.1±0.3**	**65.0±0.3**	**89.6±0.1**	**67.7±0.4**	71.1±0.3	**64.6±0.2**	**71.2**
		Medium	Graph Model	63.2±0.4	62.2±0.2	59.9±0.3	79.6±0.3	61.1±0.2	70.6±0.4	63.3±0.3	65.7
			Graph Model+ConRep4CO	**71.5±0.3**	**65.4±0.4**	**63.4±0.2**	**81.7±0.2**	**64.0±0.3**	**72.3±0.3**	**64.4±0.4**	**69.0**
VCG+GCN	GCN	Easy	Graph Model	77.0±0.3	58.5±0.2	60.3±0.4	86.1±0.2	62.7±0.3	**71.2±0.2**	63.6±0.3	68.5
			Graph Model+ConRep4CO	**78.0±0.4**	**60.6±0.3**	**62.9±0.3**	**88.8±0.1**	**66.3±0.2**	71.1±0.3	**64.2±0.2**	**70.3**
		Medium	Graph Model	63.2±0.4	62.2±0.2	59.9±0.3	79.6±0.3	61.1±0.2	70.6±0.4	63.3±0.3	65.7
			Graph Model+ConRep4CO	**70.8±0.3**	**64.2±0.4**	**63.0±0.2**	**80.4±0.2**	**62.1±0.3**	**71.8±0.3**	**64.0±0.4**	**68.0**
LCG+NeuroSAT	GraphSAGE	Easy	Graph Model	57.9±0.5	50.0±0.3	50.7±0.4	61.8±0.2	52.2±0.3	58.2±0.3	53.8±0.4	54.9
			Graph Model+ConRep4CO	**79.7±0.2**	**63.2±0.3**	**70.8±0.3**	**93.3±0.1**	**75.3±0.4**	**71.0±0.2**	**63.9±0.3**	**73.9**
		Medium	Graph Model	52.8±0.4	56.5±0.2	56.0±0.3	55.2±0.3	50.0±0.4	58.2±0.3	54.8±0.2	54.8
			Graph Model+ConRep4CO	**72.8±0.3**	**64.1±0.4**	**66.7±0.2**	**85.9±0.2**	**70.1±0.3**	**71.7±0.4**	**64.8±0.3**	**70.9**
LCG+NeuroSAT	PGN	Easy	Graph Model	76.2±0.3	58.4±0.2	66.4±0.4	91.6±0.1	67.9±0.3	68.7±0.2	61.7±0.3	70.1
			Graph Model+ConRep4CO	**77.3±0.2**	**61.9±0.3**	**69.7±0.3**	**93.7±0.2**	**71.6±0.4**	**70.3±0.3**	61.7±0.2	**72.3**
		Medium	Graph Model	**72.4±0.4**	62.8±0.3	64.7±0.2	83.0±0.3	**68.1±0.2**	58.8±0.4	50.4±0.3	65.7
			Graph Model+ConRep4CO	72.0±0.3	**63.3±0.4**	**66.0±0.3**	**86.4±0.2**	67.2±0.3	**70.8±0.3**	**63.3±0.4**	**69.9**
LCG+NeuroSAT	GraphGPS	Easy	Graph Model	82.4±0.2	77.2±0.3	85.5±0.1	89.9±0.2	76.4±0.3	69.4±0.4	**67.4±0.2**	78.3
			Graph Model+ConRep4CO	**83.9±0.3**	**77.4±0.2**	**88.5±0.2**	**90.6±0.1**	**78.4±0.4**	**76.3±0.3**	66.4±0.3	**80.2**
		Medium	Graph Model	70.7±0.4	62.5±0.3	66.8±0.2	84.9±0.3	61.8±0.2	**69.4±0.3**	62.6±0.4	68.2
			Graph Model+ConRep4CO	**71.7±0.3**	**72.9±0.4**	**81.8±0.3**	**85.6±0.2**	**73.0±0.3**	57.2±0.4	**63.2±0.3**	**72.2**

3.3.4 Incorporating ConRep4CO into Problem-Specific Neural Solvers

Despite selecting GDPs as the pre-training tasks, ConRep4CO is not restricted to only GDP solving in practical applications. ConRep4CO can be incorporated into problem-specific neural solvers to enhance the learned representation and ultimately improve problem-solving. Suppose one has pre-trained a SAT model (and graph models) using ConRep4CO. The following steps can be taken:

1) Convert the decision version instances of the neural solver's corresponding problem domain into CNFs. This can typically be implemented through off-the-shelf tools, such as CNFGen [30], and requires polynomial complexity, which is negligible for computing labels for the training instances.

2) Use the loss function defined in Sec. 3.3.2 to perform contrastive learning between the neural solver model and the pre-trained SAT model. The neural solver's architecture may need minor adjustments, such as adding an output module, while the parameters of the SAT model are fixed, as the pre-trained SAT model already contains unified representations for various problems.

3) Fine-tune the neural solver in the original problem domain to adapt the learned representations.

4 Experiments

4.1 Experimental Setup

Datasets. To evaluate the broad applicability of our approach, we select seven GDPs: k-Clique, k-Dominating Set (k-Domset), k-Vertex Cover (k-Vercov), k-Coloring (k-Color), k-Independent Set (k-Indset), Perfect Matching (Matching), and Graph Automorphism (Automorph). For each problem, we randomly generate graph instances that adhere to a distribution specific to the problem. To ensure a comprehensive and rigorous evaluation, we create datasets with varying levels of difficulty, categorized as easy, medium, and hard, based on the size and distribution of the generated graphs. For each easy and medium dataset, we generate 160,000 instances for training, 20,000 instances for validation, and 20,000 instances for testing. For each hard dataset, we only produce 20,000 instances for testing to evaluate the generalizability of the learned representations. Additionally, we ensure an equal distribution of labels, with 50% of instances labeled as satisfiable (1) and 50% as unsatisfiable (0) across the training, validation, and test sets. The graph instances were transformed into CNF using generators from CNFGen [30]. Furthermore, we evaluate the effectiveness of ConRep4CO in enhancing CO problem-solving on four practical CO problems: minimum vertex cover (MVC), maximum independent set (MIS), maximum clique (MC), and minimum dominating set (MDS). For MVC, we follow the setting in [54], using Erdős–Rényi (ER) graphs with three scales, containing approximately 50 to 100, 100 to 200, and 400 to 500 vertices, respectively. For MIS, MC, and MDS, we follow the setting in [56], using RB graphs [53] for MIS and MC, and BA graphs [2] for

Table 2: GDP solving accuracy (%) with confidence intervals ($\alpha = 0.05$) of the graph models on the hard datasets, measuring the generalizability of learned representations. 'SAT Back.' refers to SAT model backbone, and 'Graph Back.' denotes graph model backbone. The terms 'Easy' and 'Medium' in parentheses indicate the difficulty level of the datasets used for training. The 'Overall' column represents the average accuracy across all datasets.

SAT Back.	Graph Back.	Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
LCG+NeuroSAT	GCN	Graph Model (Easy)	54.5±0.2	50.0±0.1	50.0±0.1	54.6±0.4	**50.5±0.2**	66.4±0.3	63.1±0.1	55.6
		Graph Model+ConRep4CO (Easy)	**57.1±0.1**	**50.1±0.1**	50.0±0.1	**60.5±0.3**	50.3±0.2	**67.9±0.4**	**63.6±0.2**	**57.1**
		Graph Model (Medium)	57.1±0.1	56.2±0.1	50.0±0.1	63.7±0.5	53.1±0.5	68.3±0.3	63.2±0.2	58.8
		Graph Model+ConRep4CO (Medium)	**57.8±0.2**	**56.5±0.1**	**57.7±0.3**	**67.6±0.5**	**56.5±0.4**	**70.0±0.2**	**65.3±0.2**	**61.6**
LCG+GCN	GCN	Graph Model (Easy)	**54.5±0.3**	50.0±0.0	50.0±0.0	54.6±0.4	**50.5±0.2**	66.4±0.3	63.1±0.4	55.6
		Graph Model+ConRep4CO (Easy)	52.5±0.4	50.0±0.0	**53.9±0.3**	**55.7±0.2**	49.9±0.3	**68.6±0.2**	63.1±0.3	**56.2**
		Graph Model (Medium)	57.1±0.2	56.2±0.3	50.0±0.0	63.7±0.3	53.1±0.4	68.3±0.4	63.2±0.2	58.8
		Graph Model+ConRep4CO (Medium)	**57.9±0.3**	**58.9±0.2**	**57.4±0.4**	**65.6±0.3**	**55.2±0.3**	**71.2±0.3**	**64.5±0.4**	**61.5**
VCG+GCN	GCN	Graph Model (Easy)	**54.5±0.3**	50.0±0.0	50.0±0.0	54.6±0.4	**50.5±0.2**	66.4±0.3	63.1±0.4	55.6
		Graph Model+ConRep4CO (Easy)	53.1±0.2	50.0±0.0	50.0±0.0	**55.4±0.3**	49.6±0.4	**68.4±0.4**	**63.4±0.3**	**55.7**
		Graph Model (Medium)	57.1±0.2	56.2±0.3	50.0±0.0	63.7±0.3	53.1±0.4	68.3±0.4	63.2±0.2	58.8
		Graph Model+ConRep4CO (Medium)	**57.7±0.4**	**60.5±0.3**	**57.7±0.3**	**64.8±0.2**	**53.6±0.3**	**69.0±0.3**	**64.3±0.3**	**61.1**
LCG+NeuroSAT	GraphSAGE	Graph Model (Easy)	50.9±0.4	50.3±0.3	48.1±0.2	50.8±0.3	50.5±0.2	57.8±0.4	55.7±0.3	52.0
		Graph Model+ConRep4CO (Easy)	**52.9±0.3**	**59.9±0.4**	**55.9±0.4**	**60.2±0.2**	**58.5±0.3**	**67.9±0.3**	**62.1±0.4**	**59.6**
		Graph Model (Medium)	50.9±0.4	57.3±0.2	54.7±0.3	50.2±0.4	48.9±0.3	58.4±0.3	55.8±0.2	53.7
		Graph Model+ConRep4CO (Medium)	**59.7±0.3**	**59.5±0.3**	**60.3±0.2**	**70.2±0.4**	**56.4±0.4**	**68.4±0.2**	**64.2±0.3**	**62.7**
LCG+NeuroSAT	PGN	Graph Model (Easy)	54.2±0.3	59.3±0.2	59.5±0.4	63.1±0.3	54.9±0.2	66.3±0.4	60.3±0.3	59.7
		Graph Model+ConRep4CO (Easy)	**54.6±0.2**	**59.8±0.3**	**59.9±0.3**	**63.3±0.4**	**55.1±0.3**	**66.7±0.3**	**61.0±0.4**	**60.1**
		Graph Model (Medium)	60.4±0.4	58.6±0.3	59.7±0.2	69.1±0.3	55.9±0.4	67.1±0.2	**63.5±0.3**	62.0
		Graph Model+ConRep4CO (Medium)	**61.2±0.3**	**58.9±0.4**	**60.7±0.3**	**69.7±0.2**	**58.1±0.3**	**67.5±0.3**	63.3±0.2	**62.8**
LCG+NeuroSAT	GraphGPS	Graph Model (Easy)	**59.6±0.2**	50.0±0.0	49.9±0.1	50.0±0.0	53.5±0.3	**68.0±0.4**	57.6±0.3	55.5
		Graph Model+ConRep4CO (Easy)	59.3±0.3	**50.7±0.1**	**60.9±0.4**	**59.6±0.3**	53.5±0.4	58.9±0.3	**59.5±0.4**	**57.5**
		Graph Model (Medium)	63.2±0.3	55.2±0.4	56.8±0.2	68.3±0.4	63.0±0.3	**63.9±0.3**	58.3±0.4	61.2
		Graph Model+ConRep4CO (Medium)	**63.8±0.4**	**60.8±0.3**	**77.9±0.3**	**68.9±0.3**	**65.7±0.4**	60.1±0.2	**61.4±0.3**	**65.5**

Table 3: GDP solving accuracy (%) with confidence intervals ($\alpha = 0.05$) of the graph models on Easy datasets. The 'Overall' column represents the average accuracy across all datasets.

Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
Graph Model	76.2±0.2	58.4±0.4	66.4±0.2	91.6±0.5	67.9±0.3	68.7±0.3	61.7±0.2	70.1
Graph Model-Unseen	**76.9±0.3**	**61.0±0.1**	**70.1±0.2**	**93.4±0.4**	**68.6±0.3**	**69.7±0.1**	61.7±0.2	**71.6**

MDS, generating two scales of datasets with approximately 200 to 300 and 800 to 1200 vertices, respectively. Please refer to Appendix C for more details about the dataset description and statistics.

Graph/SAT Model Backbones. We implement multiple GNN backbones for the Representation Extractor in both graph and SAT models. For the graph models, we adopt two widely used backbones, GCN [28] and GraphSAGE [19], and two advanced backbones with stronger representational capacity, PGN [47] and GraphGPS [39]. For the SAT model, we implement NeuroSAT and a GCN architecture specifically tailored for SAT graphs. Moreover, we employ both LCG and VCG as SAT graph representations. Please refer to Appendix D for more details.

Tasks. The evaluation tasks can be divided into two parts: **1) Representation evaluation**, measured by the performance of graph models on the GDP-solving task, focusing on how the learned representations can accurately determine the solution for each specific problem type. The GDP-solving performance on more difficult instances and unseen GDP domains is used to assess both the in-domain (problem scale) and cross-domain generalizability of the learned representations. **2) Enhancement of CO problem-solving**, measured by the performance on four CO problems—MVC, MIS, MC, and MDS—when incorporating ConRep4CO into problem-specific neural solvers, as described in Sec. 3.3.4. In our experiments, we also observe that the representations learned by the SAT model have also been enhanced and show potential for use in downstream SAT tasks. Detailed experiments related to the SAT model can be found in Appendix F.3 and F.4.

Baselines. For **1) representation evaluation**, to ensure a fair comparison, we establish baselines for the graph models trained by ConRep4CO by keeping the architectures identical while modifying only the training procedures. Each baseline graph model is trained independently on its corresponding dataset. For **2) enhancement of CO problem-solving**, OptGNN and GCNN from [54] are used as baseline neural solvers for MVC, while GFlowNet from [56] serves as the baseline neural solver for MIS, MC, and MDS. These models are trained using the methods described in their respective papers and compared to those incorporating ConRep4CO as outlined in Sec. 3.3.4.

Table 4: Performance on MVC. ‘OBJ’ refers to the average objective value, where lower is better for MVC. ‘Optimal’ represents the best-known solution obtained using Gurobi [18].

Graph	Optimal	OptGNN		OptGNN+ConRep4CO		gain	GCNN		GCNN+ConRep4CO		gain
		OBJ	gap_{abs}	OBJ	gap_{abs}		OBJ	gap_{abs}	OBJ	gap_{abs}	
ER(50,100)	54.62	55.87	1.25	**54.70**	**0.08**	**93.60%**	55.34	0.72	**55.17**	**0.55**	**23.61%**
ER(100,200)	122.79	126.04	3.25	**124.37**	**1.58**	**51.40%**	128.29	5.50	**126.75**	**3.96**	**28.00%**
ER(400,500)	417.42	420.51	3.09	**419.31**	**1.89**	**38.83%**	443.43	26.01	**436.77**	**19.35**	**25.61%**
avg. gain	-	-	-	-	-	**61.27%**	-	-	-	-	**25.74%**

4.2 Representation Evaluation

4.2.1 Representation Quality Evaluation

We evaluate the quality of the learned representations by comparing the accuracy of graph models in solving seven GDPs. The baseline model is referred to as **Graph Model**, which is trained independently on its corresponding dataset. Our proposed approach, denoted **Graph Model+ConRep4CO**, initializes the model parameters with a pre-trained checkpoint from ConRep4CO trained on the seven GDP datasets and is then fine-tuned individually. Table 1 presents the results for six combinations of SAT and graph backbones, evaluating the performance of both models trained and tested on datasets with identical distributions, including the easy and medium difficulty datasets. The proposed approach consistently outperforms the baseline model across most GDP tasks, at both difficulty levels, and for all six backbone combinations. These findings indicate that integrating ConRep4CO substantially enhances the quality of the learned representations, enabling more effective capture of the underlying features and characteristics of GDPs. Consequently, the enhanced representations lead to improved accuracy in solving GDPs. Notably, when employing the GraphSAGE backbone, our approach demonstrates a particularly significant performance improvement over the baseline.

4.2.2 In-Domain Generalizability Evaluation

To assess the in-domain generalization capabilities of the learned representations, particularly in relation to problem scale, we evaluate their performance on previously unseen hard GDP datasets, which consist of problem instances with increased scale and complexity. Table 2 presents the results for six combinations of SAT and graph backbones, with graph models trained on the easy and medium datasets and tested on the hard datasets. The results clearly demonstrate that the representations learned by ConRep4CO show improved performance across most GDP tasks, both difficulty levels, and all six backbone combinations. This indicates that ConRep4CO also improves generalizability to more challenging and complex problem instances that were previously unseen. The improvement likely stems from the information transfer between various GDPs during pre-training, allowing the model to learn shared, more general, and high-level representations. This generality is applicable to problem scale, further reinforcing the robustness of the learned representations.

4.2.3 Cross-Domain Generalizability Evaluation

To further evaluate cross-domain generalizability, particularly the ability to generalize to unseen problem domains during pre-training, we select 6 out of 7 GDPs as pre-training domains, with the remaining GDP serving as the generalizing domain. After pre-training, we define a new graph model for the generalizing domain, align it with the pre-trained SAT model from the pre-training domains, and fine-tune it on 20,000 instances from the generalizing domain. The graph model for the generalizing domain is referred to as **Graph Model-Unseen**. We conduct experiments on all combinations of pre-training domains and report the GDP-solving accuracy for the generalizing domain in Table 3. Note that each number in the second row corresponds to a complete independent experiment, where the dataset indicated in the header is the generalizing domain, and the remaining six GDPs serve as pre-training domains. The results demonstrate that ConRep4CO enables the graph model to learn better representations for an unseen GDP domain with minimal data, outperforming the baseline. This indicates that ConRep4CO not only enhances representation quality for specific tasks but also improves representation learning for problem domains that are completely unseen. The improvement is likely due to the unified SAT model capturing shared features among pre-training domains, allowing it to learn a general GDP representation applicable to unseen GDPs. Aligning with the SAT model helps the graph model learn new GDP representations more effectively and efficiently.

The results also reveal that it is not necessary to include all GDP domains during pre-training when scaling to a large number of GDPs. Aligning with the pre-trained SAT model allows for effective transfer to new GDPs with minimal data, showing the practical applicability of ConRep4CO.

Table 5: Performance on MIS, MC, and MDS. 'OBJ' refers to the average objective value, where higher is better for MIS and MC, and lower is better for MDS. 'Optimal' denotes the best-known solution, obtained using KAMIS [29] for MIS and Gurobi [18] for MC and MDS.

Problem Type	**Graph**	**Optimal**	**GFlowNet**		**GFlowNet+ConRep4CO**		**gain**	**avg. gain**
			OBJ	**gap$_{\text{abs}}$**	**OBJ**	**gap$_{\text{abs}}$**		
MIS$\uparrow$	RB(200,300)	20.10	19.18	0.92	19.56	**0.54**	**41.30%**	**32.20%**
	RB(800,1200)	43.15	37.48	5.67	38.79	**4.36**	**23.10%**	
MC$\uparrow$	RB(200,300)	19.05	16.24	2.81	17.47	**1.58**	**43.77%**	**36.46%**
	RB(800,1200)	33.89	31.42	2.47	32.14	**1.75**	**29.15%**	
MDS$\downarrow$	BA(200,300)	27.89	28.61	0.72	28.19	**0.30**	**58.33%**	**45.29%**
	BA(800,1200)	103.80	110.28	6.48	108.19	**4.39**	**32.25%**	

Table 6: Ablation study on cross-domain information transfer. The table presents GDP-solving accuracy (%) with confidence intervals ($\alpha = 0.05$). 'Graph Model+ConRep4CO+Single Domain' refers to the ablated method by disabling cross-domain information transfer.

Difficulty	Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
Easy	Graph Model+ConRep4CO+Single Domain	78.4±0.4	61.7±0.1	67.1±0.1	89.9±0.2	65.6±0.4	71.5±0.2	65.3±0.2	71.4
	Graph Model+ConRep4CO	**79.3±0.3**	**62.0±0.1**	**67.3±0.2**	**90.2±0.1**	**67.5±0.5**	**71.7±0.3**	**65.4±0.3**	**71.9**
Medium	Graph Model+ConRep4CO+Single Domain	70.9±0.3	64.0±0.1	62.9±0.1	81.0±0.3	59.9±0.4	72.5±0.2	64.3±0.4	67.9
	Graph Model+ConRep4CO	**71.3±0.5**	**64.6±0.2**	**63.3±0.3**	**82.2±0.2**	**64.0±0.1**	**72.8±0.4**	**65.7±0.4**	**69.1**

4.3 Enhancement of CO Problem-Solving

We incorporate ConRep4CO into the training of neural solvers for four CO problems to evaluate how it enhances general CO problem-solving. The incorporation process is described in Sec. 3.3.4, denoted as **+ConRep4CO**. The pre-training datasets are the k-Vercov dataset for MVC, the k-Indset for MIS, the k-Clique for MC, and the k-Domset for MDS, all with the easy difficulty level. Table 4 presents the results for MVC, while Table 5 shows the results for MIS, MC, and MDS. 'gap$_{\text{abs}}$' refers to the absolute gap between the output objective and the optimal value, and 'gain' represents the improvement from incorporating ConRep4CO over the baseline, which is calculated as $(\text{gap}_{\text{abs, baseline}} - \text{gap}_{\text{abs, ours}})/\text{gap}_{\text{abs, baseline}} \times 100\%$. 'avg. gain' is the average gain for each problem across all problem scales. Incorporating ConRep4CO yields average gains of 61.27% (OptGNN), 32.20%, 36.46%, and 45.29% for MVC, MIS, MC, and MDS, respectively. We observe that the gain tends to be smaller for larger-scale problems, likely due to the increased difficulty and the growing discrepancy between the scales of the pre-training datasets and the evaluated problem. However, across all problem scales, we consistently see substantial performance enhancements. These results demonstrate that ConRep4CO enhances general CO problem-solving beyond GDPs by effectively integrating with problem-specific neural solvers, highlighting its strong application potential.

4.4 Analysis on Cross-Domain Information Transfer

A central component of our framework is the facilitation of information transfer across different problem domains. To evaluate the effectiveness of this mechanism, we conduct an ablation study by disabling the cross-domain information transfer. Specifically, we train each graph model independently with its own SAT model, without leveraging cross-domain information. We then compare this ablated approach with our original method, as shown in Table 6. The results indicate that the ablated approach yields inferior performance, thereby highlighting the importance and effectiveness of the cross-domain information transfer in enhancing representation learning.

We also conduct more supplementary experiments to provide further and comprehensive analysis on ConRep4CO. Please refer to Appendix G for more results.

5 Conclusion and Outlook

We have introduced ConRep4CO, a novel contrastive paradigm designed to promote learning representation for CO problems across problem types. The results indicate that it not only improves the quality and generalizability of the learned representations but also significantly enhances general CO problem-solving, highlighting the potential of ConRep4CO as a unified pre-training paradigm for CO research. Future work will focus on exploring unsupervised approaches to reduce reliance on labeled data, thereby increasing applicability in data-sparse scenarios. Additionally, we will investigate other potential forms to replace the currently used SAT form to perform contrastive learning, aiming for broader application to more general CO problems, such as mixed-integer linear programs (MILP).

ETHICS STATEMENT

This research follows the ICLR Code of Ethics and ensures that no ethical concerns are overlooked. The study does not involve human subjects or sensitive personal data. All datasets used are either synthetic or publicly available, and the methods for data generation and transformation are transparent and reproducible. We have tried to minimize bias in our experiments and ensure that the results do not perpetuate unfair outcomes. Additionally, the authors declare no conflicts of interest or financial sponsorships that could have influenced the research. Our work is committed to contributing positively to the field of combinatorial optimization.

REPRODUCIBILITY STATEMENT

We prioritize reproducibility and have taken steps to ensure that our experiments can be independently replicated. Detailed descriptions of the datasets are provided in Appendix C. Additionally, the detailed network architectures are outlined in Appendix D. Once accepted, we will release the source code for our experiments to facilitate further validation and experimentation by the community.

REFERENCES

[1] JL Ba. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *science*, 286(5439):509–512, 1999.

[3] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[4] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009.

[5] Béla Bollobás and Paul Erdös. Cliques in random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 80, pp. 419–427. Cambridge University Press, 1976.

[6] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE transactions on knowledge and data engineering*, 30(9):1616–1637, 2018.

[7] Junyang Cai, Taoan Huang, and Bistra Dilkina. Multi-task representation learning for mixed integer linear programming. *arXiv preprint arXiv:2412.14409*, 2024.

[8] Fenxiao Chen, Yun-Cheng Wang, Bin Wang, and C-C Jay Kuo. Graph representation learning: a survey. *APSIPA Transactions on Signal and Information Processing*, 9:e15, 2020.

[9] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

[10] Darko Drakulic, Sofia Michel, and Jean-Marc Andreoli. Goal: A generalist combinatorial optimization agent learner. In *The Thirteenth International Conference on Learning Representations*, 2024.

[11] Haonan Duan, Pashootan Vaezipoor, Max B Paulus, Yangjun Ruan, and Chris Maddison. Augment with care: Contrastive learning for combinatorial problems. In *International Conference on Machine Learning*, pp. 5627–5642. PMLR, 2022.

[12] Paul Erdos, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. math. inst. hung. acad. sci*, 5(1):17–60, 1960.

[13] ABKFM Fleury and Maximilian Heisinger. Cadical, kissat, paracooba, plingeling and treengeling entering the sat competition 2020. *SAT COMPETITION*, 2020:50, 2020.

[14] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. *Advances in neural information processing systems*, 32, 2019.

[15] Maxime Gasse, Simon Bowly, Quentin Cappart, Jonas Charfreitag, Laurent Charlin, Didier Chételat, Antonia Chmiela, Justin Dumouchelle, Ambros Gleixner, Aleksandr M Kazachkov, et al. The machine learning for combinatorial optimization competition (ml4co): Results and insights. In *NeurIPS 2021 competitions and demonstrations track*, pp. 220–231. PMLR, 2022.

[16] Jesús Giráldez-Cru and Jordi Levy. A modularity-based random sat instances generator. 2015.

[17] Jesús Giráldez-Cru and Jordi Levy. Locality in random sat instances. International Joint Conferences on Artificial Intelligence, 2017.

[18] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024. URL `https://www.gurobi.com`.

[19] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017.

[20] William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[21] Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International conference on machine learning*, pp. 4116–4126. PMLR, 2020.

[22] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

[23] Taoan Huang, Aaron M Ferber, Yuandong Tian, Bistra Dilkina, and Benoit Steiner. Searching large neighborhoods for integer linear programs with contrastive learning. In *International Conference on Machine Learning*, pp. 13869–13890. PMLR, 2023.

[24] Chaitanya K Joshi, Thomas Laurent, and Xavier Bresson. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

[25] Richard M Karp. Reducibility among combinatorial problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, pp. 219–241. Springer, 2009.

[26] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 2010.

[27] Elias Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. Learning combinatorial optimization algorithms over graphs. *Advances in neural information processing systems*, 30, 2017.

[28] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[29] Sebastian Lamm, Peter Sanders, Christian Schulz, Darren Strash, and Renato F Werneck. Finding near-optimal independent sets at scale. In *2016 Proceedings of the eighteenth workshop on algorithm engineering and experiments (ALENEX)*, pp. 138–150. SIAM, 2016.

[30] Massimo Lauria, Jan Elffers, Jakob Nordström, and Marc Vinyals. Cnfgen: A generator of crafted benchmarks. In *Theory and Applications of Satisfiability Testing–SAT 2017: 20th International Conference, Melbourne, VIC, Australia, August 28–September 1, 2017, Proceedings 20*, pp. 464–473. Springer, 2017.

[31] Zhaoyu Li, Jinpei Guo, and Xujie Si. G4satbench: Benchmarking and advancing sat solving with graph neural networks. *arXiv preprint arXiv:2309.16941*, 2023.

[32] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134:105400, 2021.

[33] Emils Ozolins, Karlis Freivalds, Andis Draguns, Eliza Gaile, Ronalds Zakovskis, and Sergejs Kozlovics. Goal-aware neural sat solver. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

[34] Wenzhen Pan, Hao Xiong, Jiale Ma, Wentao Zhao, Yang Li, and Junchi Yan. Unico: On unified combinatorial optimization via problem reduction to matrix-encoded general tsp. In *International Conference on Learning Representations*, 2025.

[35] Rajvardhan Patil, Sorio Boit, Venkat Gudivada, and Jagadeesh Nandigam. A survey of text representation and embedding techniques in nlp. *IEEE Access*, 11:36120–36146, 2023.

[36] Aleksandar Pekeč and Michael H Rothkopf. Combinatorial auction design. *Management science*, 49(11):1485–1503, 2003.

[37] Marcelo Prates, Pedro HC Avelar, Henrique Lemos, Luis C Lamb, and Moshe Y Vardi. Learning to solve np-complete problems: A graph neural network for decision tsp. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4731–4738, 2019.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

[39] Ladislav Rampášek, Michael Galkin, Vijay Prakash Dwivedi, Anh Tuan Luu, Guy Wolf, and Dominique Beaini. Recipe for a general, powerful, scalable graph transformer. *Advances in Neural Information Processing Systems*, 35:14501–14515, 2022.

[40] Ryoma Sato, Makoto Yamada, and Hisashi Kashima. Approximation ratios of graph neural networks for combinatorial problems. *Advances in Neural Information Processing Systems*, 32, 2019.

[41] Abdelkader Sbihi and Richard W Eglese. Combinatorial optimization and green logistics. *Annals of Operations Research*, 175:159–175, 2010.

[42] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations*, 2019.

[43] Yong Shi and Yuanying Zhang. The neural network methods for solving traveling salesman problem. *Procedia Computer Science*, 199:681–686, 2022.

[44] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pp. 776–794. Springer, 2020.

[45] Zekun Tong, Yuxuan Liang, Henghui Ding, Yongxing Dai, Xinke Li, and Changhu Wang. Directed graph contrastive learning. *Advances in neural information processing systems*, 34: 19580–19593, 2021.

[46] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[47] Petar Veličković, Lars Buesing, Matthew Overlan, Razvan Pascanu, Oriol Vinyals, and Charles Blundell. Pointer graph networks. *Advances in Neural Information Processing Systems*, 33: 2232–2244, 2020.

[48] Sheng Wan, Yibing Zhan, Liu Liu, Baosheng Yu, Shirui Pan, and Chen Gong. Contrastive graph poisson networks: Semi-supervised learning with extremely limited labels. *Advances in Neural Information Processing Systems*, 34:6316–6327, 2021.

[49] Chenguang Wang and Tianshu Yu. Efficient training of multi-task combinarotial neural solver with multi-armed bandits. *arXiv preprint arXiv:2305.06361*, 2023.

[50] Jiaqi Wang, Zhengliang Liu, Lin Zhao, Zihao Wu, Chong Ma, Sigang Yu, Haixing Dai, Qiushi Yang, Yiheng Liu, Songyao Zhang, et al. Review of large vision models and visual prompt engineering. *Meta-Radiology*, 1(3):100047, 2023.

[51] Nathan Wetzler, Marijn JH Heule, and Warren A Hunt Jr. Drat-trim: Efficient checking and trimming using expressive clausal proofs. In *International Conference on Theory and Applications of Satisfiability Testing*, pp. 422–429. Springer, 2014.

[52] Ben Wieland and Anant P Godbole. On the domination number of a random graph. *the electronic journal of combinatorics*, pp. R37–R37, 2001.

[53] Ke Xu and Wei Li. Exact phase transitions in random constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 12:93–103, 2000.

[54] Morris Yau, Nikolaos Karalias, Eric Lu, Jessica Xu, and Stefanie Jegelka. Are graph neural networks optimal approximation algorithms? *Advances in Neural Information Processing Systems*, 37:73124–73181, 2024.

[55] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. Graph contrastive learning with augmentations. *Advances in neural information processing systems*, 33:5812–5823, 2020.

[56] Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron C Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with gflownets. *Advances in neural information processing systems*, 36:11952–11969, 2023.

A Discussion with Related Work

In this section, we discuss the key differences of our ConRep4CO with some prior works [10; 7] proposing general representation learning methods for CO.

A.1 Comparison with [10]

GOAL [10] aims to develop a generalist model that uses a single backbone to represent multiple CO problems. The model employs a multi-type Transformer architecture and attention blocks, which suggest that different CO problems activate distinct portions of the backbone parameters. Consequently, GOAL accommodates different problems by utilizing different parts of the shared parameters, rather than exploiting the commonalities between different problems. Furthermore, GOAL's performance is slightly inferior to problem-specific baselines, indicating that the learned representations are not enhanced for each problem.

In contrast, ConRep4CO focuses on capturing the shared underlying structure across multiple CO problems, seeking a higher-level, abstract representation that encapsulates the essence common to different problem domains. By using SAT as a unified intermediary, ConRep4CO facilitates knowledge transfer and mutual enhancement among problems, enabling insights gained from one problem to improve the performance on others. Therefore, ConRep4CO outperforms baselines trained on individual problems.

Overall, ConRep4CO differs from GOAL in its emphasis on leveraging the shared structure of multiple CO problems to improve the learned representation for individual problems, rather than simply accommodating problem-specific variations within a single model.

A.2 Comparison with [7]

The MILP multi-task framework [7] focuses on learning shared representations across tasks within a single problem domain. It seeks to exploit commonalities among different tasks within the MILP domain. One could draw an analogy between the front-end network architecture in the framework and the representation extractor in our graph model, with the task-specific layers serving as different output modules. The MILP multi-task framework emphasizes improving these output modules with a single representation extractor, while ConRep4CO prioritizes enhancing the representation extractor by knowledge transfer and mutual enhancement across diverse problem domains. By doing so, ConRep4CO learns a higher-level, abstract representation that spans various CO problems, whereas the MILP framework primarily concentrates on task-specific output layers for a single problem domain.

To the best of our knowledge, ConRep4CO is the first framework to leverage representations across different problem domains to improve representations for individual problem domains.

B Discussion on Applicability of ConRep4CO

During the pre-training phase, our framework is designed to handle any NP problem. As long as a problem can be reduced to SAT, it can be trained within our framework. We would like to clarify that all NP problems can be reduced to NPC problems, which in turn can be transformed into SAT problems. This theoretical foundation ensures that our framework is applicable to all NP decision problems, regardless of their specific structure or variable types. In this sense, our approach is problem-agnostic, enabling effective training on a wide range of NP decision problems.

Furthermore, our approach can facilitate vast NP-hard CO problems by pre-training on their decision versions, which are typically NPC. Most CO problems can be converted into their decision versions by adding a target value. In our approach, we can pre-train the models on the decision versions of these CO problems to learn effective representations. These representations can be applied not only to the decision versions but also to the optimization versions of these problems (original problems).

Overall, ConRep4CO can serve as a unified pre-training paradigm for a broad range of CO problems.

Table 7: Details of generated GDP datasets.

Dataset	Description	Parameters	Notes
k-Clique	The k-Clique dataset consists of graph instances of the k-Clique problem, which involves determining whether a given graph contains a clique of size k. A clique is a subset of vertices in which every pair of vertices is connected by an edge. The goal is to identify whether such a fully connected subset of k vertices exists within the graph. Instances are built on randomly generated Erdős-Rényi graphs. Parameters include number of vertices v, edge probabilities p, and clique size k.	General: $p = \binom{v}{k}^{-1/\binom{v}{2}}$, Easy dataset: $v \sim$ Uniform(5, 15), $k \sim$ Uniform(3, 4), Medium dataset: $v \sim$ Uniform(15, 20), $k \sim$ Uniform(3, 5), Hard dataset: $v \sim$ Uniform(20, 25), $k \sim$ Uniform(4, 6).	The parameter p is selected based on [5], ensuring that the expected number of k-cliques in the generated graph is equal to 1.
k-Domset	The k-Domset dataset consists of graph instances of the k-Dominating Set problem, which involves determining whether a given graph contains a dominating set of size k. A dominating set is a subset of vertices such that every vertex in the graph is either in the subset or adjacent to at least one vertex in the subset. The goal is to identify whether such a subset of k vertices exists that can 'dominate' the entire graph, ensuring that all other vertices are either in the subset or connected to it. Instances are built on randomly generated Erdős-Rényi graphs. Parameters include number of vertices v, edge probabilities p, and dominating set size k.	General: $p = 1 - \left(1 - \binom{v}{k}^{-1/(v-k)}\right)^{1/k}$, Easy dataset: $v \sim$ Uniform(5, 15), $k \sim$ Uniform(2, 3), Medium dataset: $v \sim$ Uniform(15, 20), $k \sim$ Uniform(3, 5), Hard dataset: $v \sim$ Uniform(20, 25), $k \sim$ Uniform(4, 6).	The parameter p is selected based on [52], ensuring that the expected number of k-dominating sets in the generated graph is equal to 1.
k-Vercov	The k-Vercov dataset consists of graph instances of the k-Vertex Cover problem, which involves determining whether a given graph contains a vertex cover of size k. A vertex cover is a subset of vertices such that every edge in the graph is incident to at least one vertex in the subset. The goal is to identify whether a subset of k vertices exists that can 'cover' all the edges in the graph, ensuring that each edge is connected to at least one vertex in the subset. Instances are built on randomly generated Erdős-Rényi graphs. Parameters include number of vertices v, edge probabilities p, and vertex set size k.	General: $p = \binom{v}{k}^{-1/\binom{v}{2}}$, Easy dataset: $v \sim$ Uniform(5, 15), $k \sim$ Uniform(3, 5), Medium dataset: $v \sim$ Uniform(10, 20), $k \sim$ Uniform(6, 8), Hard dataset: $v \sim$ Uniform(15, 25), $k \sim$ Uniform(9, 10).	The parameter p is selected based on the relationship between k-Clique and k-Vercov, ensuring that the expected size of the minimum vertex cover in the generated graph is k.
k-Color	The k-Color dataset consists of graph instances of the k-Coloring problem, which involves determining whether a given graph can be colored with k colors such that no two adjacent vertices share the same color. A valid coloring assigns one of k different colors to each vertex, ensuring that vertices connected by an edge have different colors. The goal is to identify whether such a coloring scheme exists for the graph using at most k colors. Instances are built on randomly generated Erdős-Rényi graphs. Parameters include number of vertices v, edge probabilities p, and number of colors k.	General: $p = \binom{v}{k}^{-1/\binom{v}{2}}$, Easy dataset: $v \sim$ Uniform(5, 15), $k \sim$ Uniform(3, 4), Medium dataset: $v \sim$ Uniform(15, 20), $k \sim$ Uniform(3, 5), Hard dataset: $v \sim$ Uniform(20, 25), $k \sim$ Uniform(4, 6).	The parameter p is selected based on the relationship between k-Clique and k-Color, ensuring that the expected minimum number of colors for the generated graph is k.
k-Indeset	The k-Indset dataset consists of graph instances of the k-Independent Set problem, which involves determining whether a given graph contains an independent set of size k. An independent set is a subset of vertices in which no two vertices are adjacent, meaning there are no edges connecting any pair of vertices in the subset. The goal is to identify whether such a subset of k vertices exists within the graph, ensuring that the selected vertices are mutually non-adjacent. Instances are built on randomly generated Erdős-Rényi graphs. Parameters include number of vertices v, edge probabilities p, and independent set size k.	General: $p = 1 - \binom{v}{k}^{-1/\binom{v}{2}}$, Easy dataset: $v \sim$ Uniform(5, 15), $k \sim$ Uniform(3, 4), Medium dataset: $v \sim$ Uniform(15, 20), $k \sim$ Uniform(3, 5), Hard dataset: $v \sim$ Uniform(20, 25), $k \sim$ Uniform(4, 6).	The parameter p is selected based on the relationship between k-Clique and k-Indset, ensuring that the expected number of k-independent sets in the generated graph is equal to 1.
Matching	The Matching dataset consists of graph instances of the Perfect Matching problem, which involves determining whether a given graph contains a perfect matching. A perfect matching is a subset of edges in which every vertex in the graph is incident to exactly one edge in the subset. In other words, the graph's vertices can be paired off so that no vertex is left unpaired and no two edges share a vertex. The goal is to identify whether such a perfect matching exists within the graph, ensuring that all vertices are perfectly matched. Instances are built on randomly generated Erdős-Rényi graphs. Parameters include number of vertices v and edge probabilities p.	General: $p = \ln(v)/v$, Easy dataset: $v \sim$ Uniform(6, 16), should be an even number, Medium dataset: $v \sim$ Uniform(16, 24), should be an even number, Hard dataset: $v \sim$ Uniform(24, 30), should be an even number.	The selected parameter p is a sharp threshold for graph connectivity based on [12], ensuring that the generated graph is neither too dense nor too sparse.
Automorph	The Automorph dataset consists of graph instances of the Graph Automorphism problem, which involves determining whether a given graph has a non-trivial automorphism. An automorphism is a mapping of the graph's vertices to itself such that the structure of the graph is preserved, meaning that the adjacency relationships between vertices remain unchanged. The goal is to identify whether there exists a way to rearrange the vertices of the graph such that it appears identical to its original form. Instances are built on randomly generated Erdős-Rényi graphs. Parameters include number of vertices v and edge probabilities p.	General: $p = \ln(v)/v$, Easy dataset: $v \sim$ Uniform(4, 8), Medium dataset: $v \sim$ Uniform(8, 10), Hard dataset: $v \sim$ Uniform(10, 12).	The selected parameter p is a sharp threshold for graph connectivity based on [12], ensuring that the generated graph is neither too dense nor too sparse.

C More Details on Datasets

In this section, we provide more details on the utilized datasets in our main paper, including the parameters of GDP instances and the statistics of SAT instances.

C.1 GDP Instances

To ensure the generation of high-quality GDP instances that accurately capture the inherent characteristics of each problem, we carefully select the graph distributions and parameters used for instance generation. Some parameters refer to [31]. Table 7 provides a detailed overview of the specific GDP datasets employed in the main paper.

Note that six of the seven GDPs are NP-complete, while the Perfect Matching problem is a P problem.

Table 8: SAT dataset statistics. # Variables refers to average number of variables, # Clauses denoted average number of clauses, Mod. (LCG) represents average modularity of LCG graphs, and Mod. (VCG) represents average modularity of VCG graphs.

Dataset	Easy				Medium				Hard			
	# Variables	# Clauses	Mod. (LCG)	Mod. (VCG)	# Variables	# Clauses	Mod. (LCG)	Mod. (VCG)	# Variables	# Clauses	Mod. (LCG)	Mod. (VCG)
k-Clique	35.69	613.25	0.49	0.46	70.86	2298.03	0.49	0.48	114.49	5670.10	0.50	0.49
k-Domset	40.73	345.75	0.53	0.47	89.70	1708.06	0.51	0.49	137.32	4025.85	0.51	0.49
k-Vercov	46.33	498.06	0.52	0.48	108.19	2681.55	0.51	0.49	192.57	8409.32	0.51	0.50
k-Color	33.91	112.64	0.69	0.65	69.92	321.25	0.71	0.68	112.16	719.32	0.69	0.66
k-Indset	38.38	702.92	0.49	0.46	72.55	2388.22	0.49	0.48	113.12	5549.79	0.50	0.49
Matching	27.48	95.03	0.69	0.59	30.92	107.67	0.70	0.61	45.48	169.49	0.72	0.64
Automorph	56.76	943.54	0.51	0.47	82.74	1856.26	0.51	0.48	121.56	3612.56	0.51	0.49

C.2 SAT Instances

After generating the seven GDP datasets, the corresponding seven SAT datasets are generated by transforming the GDP datasets, utilizing the python toolkit CNFGen [30]. We also compute the statistics of those SAT datasets to provide comprehensive information on datasets. The dataset statistics are shown in Table 8.

Moreover, to evaluate the effectiveness of the learned representations on unseen SAT instances, we synthetically generate four more SAT datasets, including two random problems and two pseudo-industrial problems. Specifically, for random problems, we generate the SR dataset with the SR generator in NeuroSAT [42], and the 3-SAT dataset with the 3-SAT generator in CNFGen [30]. For pseudo-industrial problems, we generate the CA dataset via the Community Attachment model [16], and the PS dataset by the Popularity-Similarity model [17]. The generation process of the four datasets follows [31], where the dataset descriptions and statistics can also be found.

The ground truth of satisfiability and satisfying assignments are calculated by calling the state-of-the-art modern SAT solver CaDiCaL [13], and the truth labels for unsat core variables are generated by invoking the proof checker DRAT-trim [51].

D Details on Model Architecture

D.1 Basic Architecture Implementation

Graph Model. Each graph model is designed to address a specific type of GDP, and all models maintain a consistent architecture. To illustrate this, we focus on problem $\mathcal{P}_n$ and its corresponding graph model $\mathbb{M}_n$. The graph model $\mathbb{M}_n$ takes graphs in the set $\mathbf{G}_n$ as input and processes them through the Representation Extractor. The input graph primarily consists of edge information, which is often a critical aspect of GDPs. For the initial vertex features, we introduce a d-dimensional embedding for all vertices, represented as $\mathbf{h}_n^{(0)}$.

For the Representation Extractor, we adopt the vanilla Graph Convolutional Network (GCN) [28], which is widely used as a backbone for node embeddings in graph-based tasks. Assume there are k layers, the embedding extraction at the i-th layer of the network is expressed as:

$$\mathbf{H}_n^{(i)} = \text{ReLU}(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{H}_n^{(i-1)}\mathbf{W}_n^{(i-1)}),\ i = 1, 2, \ldots, k, \tag{3}$$

where $\mathbf{H}$ denotes the node embedding matrix, with each row corresponding to a node embedding. The matrix $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix augmented with self-loops through the identity matrix $\mathbf{I}$. $\tilde{\mathbf{D}}_{ii} = \sum_j \tilde{\mathbf{A}}_{ij}$ is the degree matrix, and $\mathbf{W}$ is the learnable weight matrix. Following the extraction of node features, we apply average pooling to the node embedding matrix $\mathbf{H}_n^{(k)}$ to aggregate the node-level information into a single representation for the entire graph instance, denoted as $\mathbf{r}_n$. This aggregation is computed as follows:

$$\mathbf{r}_n = \frac{\sum_{v \in \mathcal{V}} \mathbf{h}_{n,v}^{(k)}}{|\mathcal{V}|}, \tag{4}$$

where $\mathcal{V}$ represents the set of vertices in the input graph, $|\mathcal{V}|$ denotes the total number of vertices, and $\mathbf{h}_{n,v}^{(k)}$ is the extracted embedding for node v. $\mathbf{r}_n$ serves as the instance-level feature representation, and is subsequently fed into the Output Module, which is implemented as an MLP to produce the final decision for the instance.

SAT Model. Apart from the graph models, the SAT model $\mathbb{M}_{sat}$ processes the constructed SAT graphs via its own Representation Extractor. For illustration, we consider the LCG representation. For the initial node features, we define two distinct d-dimensional embeddings: $\mathbf{h}_l^{(0)}$ for all literal nodes and $\mathbf{h}_c^{(0)}$ for all clause nodes.

The architecture of the Representation Extractor is inspired by NeuroSAT [42]. For notational clarity, we assume that the extractor consists of k layers, with both literal and clause node embeddings being iteratively aggregated and updated at each layer. At the i-th layer, the updates for the literal and clause node embeddings are formulated as follows:

$$\mathbf{h}_l^{(i)} = \text{LayerNormLSTM}\left(\underset{c\in\mathcal{N}(l)}{\text{SUM}}\left(\text{MLP}\left(\mathbf{h}_c^{i-1}\right)\right), \mathbf{h}_l^{(i-1)}, \mathbf{h}_{\neg l}^{(i-1)}\right), \tag{5}$$

$$\mathbf{h}_c^{(i)} = \text{LayerNormLSTM}\left(\underset{l\in\mathcal{N}(c)}{\text{SUM}}\left(\text{MLP}\left(\mathbf{h}_l^{i-1}\right)\right), \mathbf{h}_c^{(i-1)}\right), \tag{6}$$

where l and c represent an arbitrary literal node and clause node, respectively, $\mathcal{N}(\cdot)$ refers to the set of neighboring nodes. The summation operator (SUM) serves as the aggregation function, while LayerNormLSTM [1] is employed as the update function.

Similar to the graph models, the instance-level representation $\mathbf{r}_{sat}$ derives by averaging the literal node embeddings after the k-th layer. The instance-level representation, along with the literal-level embeddings, is passed to the Output Module, which is also implemented as an MLP, to generate the final task-specific decisions or predictions.

D.2 Initial Vertex Features

As illustrated in the main paper, the input graphs primarily provide edge information instead of vertex features. Therefore, we should devise initial vertex features for the models. In this section, we introduce the definition of initial vertex features for the graph and SAT models.

Graph Model Vertex Feature. We begin by generating a normalized, learnable d-dimensional vector, which serves as the initial embedding shared across all vertices. For GDP datasets that do not require additional problem-specific information, such as Matching and Automorph, this initial embedding is directly used as the vertex feature for all vertices. In contrast, for GDP datasets where the parameter k plays a critical role in defining the instance characteristics, such as k-Clique and k-Vercov, we first embed k into a d-dimensional vector. The initial vertex embedding is then fused with the k embedding through an MLP to generate the final initial vertex features.

SAT Model Vertex Feature. For the SAT model, we generate initial vertex features based on the type of SAT graph representation, whether it is a Literal-Clause Graph (LCG) or a Variable-Clause Graph (VCG). In the case of the LCG graph, we initialize a normalized, learnable d-dimensional vector for all literal nodes and a separate normalized, learnable d-dimensional vector for all clause nodes. Similarly, for the VCG graph, we generate a normalized, learnable d-dimensional vector for all variable nodes and another for all clause nodes.

D.3 More Backbones

To demonstrate that the performance improvement brought about by our ConRep4CO is consistent, and independent with specialized model architectures, we conduct experiments on more backbones.

Graph Model Backbone. For the graph model, we employ an additional mainstream network architecture for node embedding, GraphSAGE [19], which is widely recognized for its ability to generate inductive representations of graph nodes by aggregating information from a node's local neighborhood. The update rule for the i-th layer of GraphSAGE is defined as follows:

$$\mathbf{n}_u^{(i)} = \text{AGG}\left(\text{ReLU}\left(\mathbf{Q}^{(i)}\mathbf{h}_v^{(i)} + \mathbf{q}^{(i)} \mid v \in N(u)\right)\right), \tag{7}$$

$$\mathbf{h}_u^{(i+1)} = \text{ReLU}\left(\mathbf{W}^{(i)}\,\text{CONCAT}\left(\mathbf{h}_u^{(i)}, \mathbf{n}_u^{(i)}\right)\right), \tag{8}$$

where $\mathbf{h}_u$ denotes the embedding for vertex u, $N(u)$ refers to the neighbors of vertex u, $\mathbf{Q}, \mathbf{q}, \mathbf{W}$ are trainable parameters, and AGG is the aggregation function. In our implementation, AGG is defined as the mean function, which computes the element-wise average of the neighbor embeddings.

In addition, we implement two more advanced GNN backbones for our graph models, PGN [47] and GraphGPS [39]. Please refer to the corresponding papers for details on the architectures of these two backbones.

SAT Model Backbone. For the SAT model, we incorporate a GCN architecture specifically tailored for SAT graphs as an additional backbone. The node updates at the i-th layer are defined as follows:

$$\mathbf{h}_l^{(i)} = \text{MLP}\left(\underset{c\in\mathcal{N}(l)}{\text{SUM}}\left(\text{MLP}\left(\mathbf{h}_c^{i-1}\right)\right), \mathbf{h}_l^{(i-1)}, \mathbf{h}_{\neg l}^{(i-1)}\right), \tag{9}$$

$$\mathbf{h}_c^{(i)} = \text{MLP}\left(\underset{l\in\mathcal{N}(c)}{\text{SUM}}\left(\text{MLP}\left(\mathbf{h}_l^{i-1}\right)\right), \mathbf{h}_c^{(i-1)}\right), \tag{10}$$

where l and c represent an arbitrary literal node and clause node, respectively. The aggregation of neighboring node information is performed using the summation operator (SUM), which serves as the aggregation function. The updates for both literal and clause nodes are computed using an MLP.

Furthermore, we extend the backbone to VCG graph modeling, where all literal nodes are replaced by variable nodes, and each literal and its negation are merged into a single variable node. The node updates at the i-th layer of the VGC-based GCN are formulated as:

$$\mathbf{h}_v^{(i)} = \text{MLP}\left(\underset{c\in\mathcal{N}(v)}{\text{SUM}}\left(\text{MLP}\left(\mathbf{h}_c^{i-1}\right)\right), \mathbf{h}_v^{(i-1)}\right), \tag{11}$$

$$\mathbf{h}_c^{(i)} = \text{MLP}\left(\underset{v\in\mathcal{N}(c)}{\text{SUM}}\left(\text{MLP}\left(\mathbf{h}_v^{i-1}\right)\right), \mathbf{h}_c^{(i-1)}\right), \tag{12}$$

where v and c represent an arbitrary variable node and clause node, respectively.

D.4 Case Study on Model Outut

In this section, we illustrate the model outputs for specific GDP and corresponding SAT problems for better understanding.

In the context of GDP, the model's output is typically binary, represented as 0 or 1, at the instance level. For instance, in the case of the k-Clique problem, the input consists of a graph, and the output indicates whether the graph contains a clique of size k. Specifically, if a k-Clique is present, the output is 1; otherwise, it is 0.

Similarly, for the corresponding SAT problem, the output denotes the satisfiability of the formula. If the formula is satisfiable, the output is 1; if not, it is 0. The satisfiability result is directly linked to the solution of the original GDP problem. For example, a satisfiable formula indicates the existence of a k-Clique in the original graph.

However, the framework is not restricted to this specific task alone. By making appropriate modifications to the architecture of the output module, the models can be adapted to solve other related tasks, including both SAT-based and GDP-based tasks.

E Loss Function for SAT-based Tasks

For the unsat core variable prediction task, we manually generate labels for the datasets, and adopt a binary cross-entropy loss on the label and the prediction.

For the satisfying assignment prediction task, we employ an unsupervised loss function as defined in [33]:

$$V_c(x) = 1 - \prod_{i\in c^+}(1-x_i)\prod_{i\in c^-}x_i, \quad \mathcal{L}_\phi(x) = -\log\left(\prod_{c\in\phi}V_c(x)\right) = -\sum_{c\in\phi}\log\left(V_c(x)\right) \tag{13}$$

Table 9: Parameters used for training.

Parameter	Value	Description
lr	1e-04	Learning rate.
lr_step_size	50	Learning rate step size.
lr_factor	0.5	Learning rate factor.
lr_patience	10	Learning rate patience.
clip_norm	1.0	Clipping norm.
weight_decay	1e-08	L2 regularzation weight.
sat_model_gnn_layer	32	Iteration number of GNN layers in SAT model.
graph_model_gnn_layer	12	Iteration number of GNN layers in graph model.
mlp_layer	2	Number of Linear layers in an MLP.
τ	0.1 (easy) / 0.5 (medium)	Temperature scalar in the contrastive loss.
β	0.5$\sim$1.0	Weight of the decision loss during training.

where ϕ refers to the CNF formula, x is the predicted assignment consisting of binary values (0 or 1) for variables, c denotes an arbitrary clause. The sets c^+ and c^- comprise the variables present in clause c in positive and negative forms, respectively. It is important to note that the loss function achieves its minimum value only when the predicted assignment x corresponds to a satisfying assignment. Minimizing this loss can effectively aid in constructing a possible satisfying assignment.

F More Experimental Results

F.1 Training Parameters

For reproducibility, we present some important parameters used for training in Table 9. More details can be found in our source code, which will be released once the paper is accepted.

F.2 Computational Cost

All training and inference tasks were conducted on a single NVIDIA H100 GPU with 80GB of memory.

The pre-training process for the SAT model and the graph models with ConRep4CO totally takes approximately 40 hours, with convergence typically occurring around the 20th epoch. Each epoch requires roughly 2 hours. Following the pre-training phase, fine-tuning takes an additional 5 to 6 hours for each model to achieve optimal performance. In comparison, training the baseline SAT model takes about 45 hours, with convergence reached by the 30th epoch, and each epoch requiring approximately 1.5 hours. Notably, pre-training with ConRep4CO demonstrates a faster convergence rate, leading to a shorter training time. Moreover, training the baseline graph model independently each requires around 15 hours, with convergence occurring around the 60th epoch, and each epoch taking between 12 to 18 minutes.

Overall, the computational cost of training with ConRep4CO is comparable to that of the conventional training approach, with no significant increase in computational burden.

F.3 Further Evaluation on SAT Model

For the SAT model, we also assess its effectiveness on the **satisfiability prediction** task. The baseline SAT model is trained concurrently on seven GDP datasets, utilizing standard supervised learning.

We assess the satisfiability prediction accuracy of the SAT model using instances transformed from seven distinct GDPs. The baseline model is denoted as **SAT Model**. The training of the baseline model capitalizes on the relatively coherent graph representations of the SAT instances. Our proposed approach, denoted as **SAT Model+ConRep4CO**, initializes model parameters with a pre-trained checkpoint from ConRep4CO, trained on the seven GDP datasets. The model is then fine-tuned on the instances transformed from all seven GDPs simultaneously. Table 10 shows the results, where our approach consistently outperforms the baseline model on most datasets, with particularly notable improvements on more challenging datasets. The results demonstrate the effectiveness of leveraging the inherent connections between different CO problems. By drawing on the common underlying

Table 10: Experimental results across various model backbones with confidence intervals ($\alpha = 0.05$). The table presents the satisfiability prediction accuracy of the SAT models. ‘SAT Back.’ refers to SAT model backbone, and ‘Graph Back.’ denotes graph model backbone.

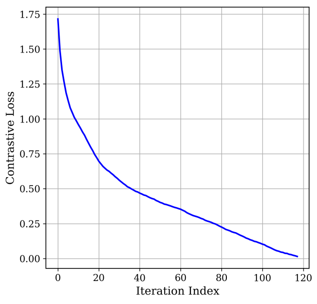
SAT Backbone	Graph Backbone	Difficulty	Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
LCG+NeuroSAT	GCN	Easy	SAT Model	95.9±0.4	99.1±0.2	99.8±0.1	97.4±0.4	95.4±0.2	99.5±0.1	99.9±0.1	98.1
			SAT Model+ConRep4CO	**98.9±0.2**	**99.6±0.1**	**99.9±0.1**	**98.8±0.3**	**98.9±0.4**	**99.9±0.1**	99.9±0.1	**99.4**
		Medium	SAT Model	87.6±0.5	98.7±0.3	99.1±0.2	81.7±0.4	88.7±0.6	99.7±0.1	98.8±0.2	93.5
			SAT Model+ConRep4CO	**92.3±0.4**	**99.1±0.1**	**99.6±0.2**	**94.6±0.4**	**93.0±0.5**	**99.9±0.1**	**99.9±0.1**	**96.9**
LCG+GCN	GCN	Easy	SAT Model	76.3±0.4	79.0±0.3	89.0±0.2	86.8±0.3	78.0±0.5	80.1±0.2	61.6±0.4	78.7
			SAT Model+ConRep4CO	**82.7±0.3**	**93.2±0.2**	**95.3±0.1**	**93.7±0.2**	**82.0±0.4**	**96.7±0.1**	**68.9±0.3**	**87.5**
		Medium	SAT Model	72.4±0.5	65.2±0.4	83.6±0.3	85.8±0.4	72.1±0.3	83.5±0.2	66.8±0.5	75.6
			SAT Model+ConRep4CO	**75.2±0.4**	**95.3±0.1**	**97.9±0.1**	**88.7±0.3**	**74.8±0.5**	**99.4±0.1**	**78.4±0.2**	**87.1**
VCG+GCN	GCN	Easy	SAT Model	51.1±0.5	84.0±0.3	91.9±0.2	82.8±0.4	49.1±0.4	81.3±0.3	56.8±0.5	71.0
			SAT Model+ConRep4CO	**80.9±0.4**	**95.9±0.1**	**99.3±0.1**	**94.7±0.2**	**79.5±0.3**	**99.3±0.1**	**74.4±0.4**	**89.1**
		Medium	SAT Model	66.9±0.4	94.6±0.2	95.0±0.2	86.0±0.3	67.7±0.5	98.8±0.1	64.2±0.4	81.9
			SAT Model+ConRep4CO	**74.8±0.5**	**98.8±0.1**	**99.5±0.1**	**89.8±0.4**	**74.5±0.4**	**99.4±0.1**	**73.4±0.3**	**87.2**
LCG+NeuroSAT	GraphSAGE	Easy	SAT Model	95.9±0.2	99.1±0.1	99.8±0.1	97.4±0.3	95.4±0.4	99.5±0.1	99.9±0.1	98.1
			SAT Model+ConRep4CO	**99.0±0.1**	**99.6±0.1**	**99.9±0.1**	**98.8±0.2**	**99.1±0.2**	**99.9±0.1**	99.9±0.1	**99.5**
		Medium	SAT Model	87.6±0.4	98.7±0.1	99.1±0.2	81.7±0.5	88.7±0.3	99.7±0.1	98.8±0.2	93.5
			SAT Model+ConRep4CO	**92.5±0.3**	**99.1±0.1**	**99.6±0.1**	**95.3±0.4**	**93.5±0.5**	**99.9±0.1**	**99.7±0.1**	**97.1**
LCG+NeuroSAT	PGN	Easy	SAT Model	95.9±0.2	99.1±0.1	99.8±0.1	97.4±0.3	95.4±0.4	99.5±0.1	99.9±0.1	98.1
			SAT Model+ConRep4CO	**98.9±0.1**	**99.6±0.1**	99.8±0.1	**98.8±0.2**	**99.1±0.2**	**99.9±0.1**	99.9±0.1	**99.4**
		Medium	SAT Model	87.6±0.4	98.7±0.1	99.1±0.2	81.7±0.5	88.7±0.3	99.7±0.1	98.8±0.2	93.5
			SAT Model+ConRep4CO	**90.5±0.5**	**99.0±0.1**	**99.5±0.1**	**94.1±0.3**	**91.4±0.4**	**99.9±0.1**	**99.7±0.1**	**96.3**
LCG+NeuroSAT	GraphGPS	Easy	SAT Model	95.9±0.2	99.1±0.1	99.8±0.1	97.4±0.3	95.4±0.4	99.5±0.1	99.9±0.1	98.1
			SAT Model+ConRep4CO	**98.6±0.2**	**99.6±0.1**	**99.9±0.1**	**98.5±0.2**	**98.7±0.3**	**99.8±0.1**	99.9±0.1	**99.3**
		Medium	SAT Model	87.6±0.4	98.7±0.1	99.1±0.2	81.7±0.5	88.7±0.3	99.7±0.1	98.8±0.2	93.5
			SAT Model+ConRep4CO	**91.4±0.4**	**99.0±0.1**	**99.6±0.1**	**93.9±0.4**	**92.2±0.5**	99.7±0.1	**99.6±0.2**	**96.5**

Table 11: Generalization performance across various model backbones on the hard datasets with confidence intervals ($\alpha = 0.05$). The table presents the satisfiability prediction accuracy of the SAT models. ‘SAT Back.’ refers to SAT model backbone, and ‘Graph Back.’ denotes graph model backbone. The terms ‘Easy’ and ‘Medium’ in parentheses indicate the difficulty level of the datasets used for training. The ‘Overall’ column represents the average accuracy across all datasets.

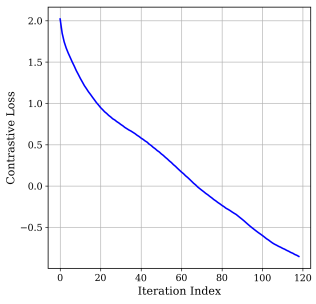
SAT Backbone	Graph Backbone	Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
LCG+NeuroSAT	GCN	SAT Model (Easy)	47.5±0.4	50.5±0.1	50.0±0.1	58.8±0.2	47.3±0.6	99.5±0.1	72.9±0.6	60.9
		SAT Model+ConRep4CO (Easy)	**66.2±0.5**	**50.6±0.1**	50.0±0.1	**60.0±0.4**	**66.5±0.6**	**99.8±0.1**	**79.0±0.3**	**67.4**
		SAT Model (Medium)	69.2±0.3	96.4±0.2	85.2±0.7	67.9±0.3	69.4±0.6	99.6±0.1	99.0 ±0.1	83.8
		SAT Model+ConRep4CO (Medium)	**82.7±0.4**	**97.2±0.2**	**93.6±0.5**	**74.5±0.2**	**83.6±0.4**	**99.7±0.1**	**99.1±0.1**	**90.1**
LCG+GCN	GCN	SAT Model (Easy)	50.0±0.0	50.0±0.0	50.0±0.0	45.9±0.3	50.0±0.0	53.9±0.4	50.0±0.0	50.0
		SAT Model+ConRep4CO (Easy)	50.0±0.0	**59.2±0.2**	50.0±0.0	**50.0±0.3**	50.0±0.0	**59.1±0.3**	**51.3±0.2**	**52.8**
		SAT Model (Medium)	50.0±0.0	50.0±0.0	50.0±0.0	49.4±0.4	50.0±0.0	47.0±0.5	50.0±0.0	49.5
		SAT Model+ConRep4CO (Medium)	50.0±0.0	50.0±0.0	50.0±0.0	**52.6±0.3**	50.0±0.0	**49.9±0.4**	50.0±0.0	**50.4**
VCG+GCN	GCN	SAT Model (Easy)	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0
		SAT Model+ConRep4CO (Easy)	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0
		SAT Model (Medium)	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0±0.0	50.0
		SAT Model+ConRep4CO (Medium)	50.0±0.0	50.0±0.0	50.0±0.0	**50.3±0.2**	50.0±0.0	50.0±0.0	50.0±0.0	50.0
LCG+NeuroSAT	GraphSAGE	SAT Model (Easy)	47.5±0.4	50.5±0.3	50.0±0.0	58.8±0.5	47.3±0.4	99.5±0.1	72.9±0.6	60.9
		SAT Model+ConRep4CO (Easy)	**59.6±0.3**	50.5±0.2	50.0±0.0	**61.5±0.4**	**58.7±0.5**	**99.6±0.1**	**82.1±0.4**	**66.0**
		SAT Model (Medium)	69.2±0.5	96.4±0.2	85.2±0.4	67.9±0.6	69.4±0.5	99.6±0.1	99.0±0.2	83.8
		SAT Model+ConRep4CO (Medium)	**79.3±0.4**	**97.3±0.1**	**89.1±0.3**	**73.1±0.5**	**79.3±0.4**	99.6±0.1	**99.6±0.1**	**88.2**
LCG+NeuroSAT	PGN	SAT Model (Easy)	47.5±0.4	50.5±0.3	50.0±0.0	58.8±0.5	47.3±0.4	**99.5±0.1**	72.9±0.6	60.9
		SAT Model+ConRep4CO (Easy)	**59.7±0.3**	**50.7±0.2**	50.0±0.0	**61.4±0.4**	**59.6±0.5**	97.9±0.2	**77.2±0.5**	**65.2**
		SAT Model (Medium)	69.2±0.5	96.4±0.2	85.2±0.4	67.9±0.6	69.4±0.5	99.6±0.1	99.0±0.2	83.8
		SAT Model+ConRep4CO (Medium)	**78.7±0.4**	**97.4±0.1**	**90.0±0.3**	**73.6±0.5**	**79.6±0.4**	**99.8±0.1**	**99.3±0.1**	**88.3**
LCG+NeuroSAT	GraphGPS	SAT Model (Easy)	47.5±0.4	50.5±0.3	50.0±0.0	58.8±0.5	47.3±0.4	**99.5±0.1**	72.9±0.6	60.9
		SAT Model+ConRep4CO (Easy)	**50.5±0.3**	**50.6±0.2**	**50.4±0.2**	**59.6±0.4**	**50.3±0.3**	99.3±0.1	**76.2±0.5**	**62.4**
		SAT Model (Medium)	69.2±0.5	96.4±0.2	85.2±0.4	67.9±0.6	69.4±0.5	**99.6±0.1**	99.0±0.2	83.8
		SAT Model+ConRep4CO (Medium)	**76.0±0.4**	**96.9±0.2**	**96.1±0.3**	**73.8±0.5**	**76.0±0.4**	99.4±0.1	**99.3±0.1**	**88.2**

characteristics among different problem types, our approach enhances the performance of the SAT model, showcasing the advantages of cross-domain learning.

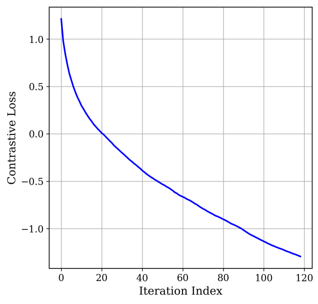
We also evaluate the generalization capabilities of the SAT models on instances transformed from hard GDP datasets, with the results presented in Table 11. Our proposed approach consistently outperforms the baseline model across most datasets, underscoring the robustness and transferability of the representations learned through ConRep4CO, and its ability to generalize across complex, unseen problem instances.



(a) CL of k-Clique



(b) CL of k-Domset



(c) CL of k-Vercov

Figure 4: Contrastive loss w.r.t. training iterations across various datasets. CL denotes the contrastive loss of the training process.

F.4 More SAT-based Task Results.

Furthermore, the evaluation is extended to two essential downstream tasks critical to SAT solving: **satisfying assignment prediction** and **unsat core variable prediction**. Satisfying assignment prediction requires the model to determine a specific variable assignment that satisfies the given SAT instance, while unsat core variable prediction involves identifying the minimal subset of variables that contribute to the unsatisfiability of the instance.

We evaluate the generalizability of the SAT model on the satisfying assignment prediction task and the unsat core variable prediction task. To assess performance, we compare three different approaches by tracking the accuracy over training iterations. For our proposed approach, referred to as **SAT Model+Contrast**, we initialize the model using a pre-trained checkpoint obtained from ConRep4CO, trained on the seven GDP datasets, and subsequently fine-tune it on individual datasets. For comparison, we include two baseline models: **SAT Model**, which is initialized with a pre-trained checkpoint trained in a conventional manner on the seven GDP datasets, and **Un-Pretrained SAT Model**, which is trained from scratch. The results are shown in Fig. 5.

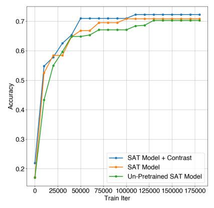
On the datasets encountered during pre-training, both our approach and the pre-trained baseline significantly outperform the un-pretrained baseline. However, our approach demonstrates superior performance by achieving faster convergence and higher final accuracy. On the unseen datasets, our approach still outperforms the baseline models, whereas the pre-trained and un-pretrained baselines exhibit comparable performance. These results highlight the effectiveness of ConRep4CO, which not only improves convergence rates but also enhances the model's ability to generalize to previously unseen datasets, thereby demonstrating the strength of leveraging contrastive learning across multiple problem types.

We show more results on the satisfying assignment prediction task and the unsat core variable prediction task in Fig. 6. Our approach outperforms the baseline models with faster convergence and higher final accuracy.

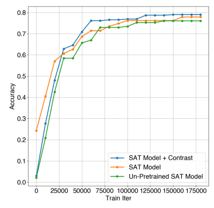
G Further Studies

G.1 Further Study on Contrastive Loss

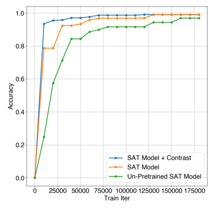
We revise the negative sampling strategy within our contrastive learning framework to mitigate the issue of false negative samples. Specifically, within each training batch, unsatisfiable instances are selected as negative samples for satisfiable instances, and conversely, satisfiable instances are chosen as negative samples for unsatisfiable instances. This adjustment ensures that false negative samples are avoided. Consequently, we modify the contrastive loss function to reflect this change and proceed with the training of the models. The results, as shown in Table 12, demonstrate that the models trained with the revised contrastive loss exhibit performance comparable to that of those trained with the original loss. We also plot the contrastive loss curves for several GDPs during the original training



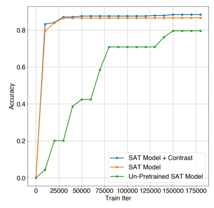
(a) Assign on 3-SAT



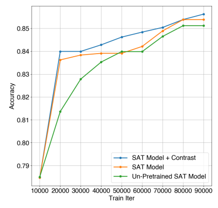
(b) Assign on CA



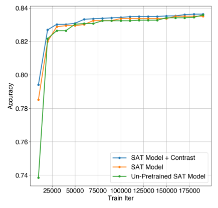
(c) Assign on k-Vercov



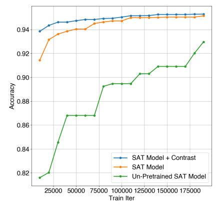
(d) Assign on k-Indset



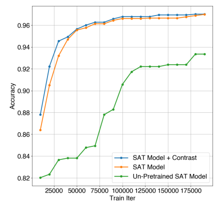
(e) Core Var on SR



(f) Core Var on PS



(g) Core Var on k-Color



(h) Core Var on Matching

Figure 5: Model performance w.r.t. training iterations on SAT-based tasks across various datasets. The top four graphs display the results for the satisfying assignment prediction task (Assign), while the bottom four graphs present the results for the unsat core variable prediction task (Core Var). The left four graphs depict the model's performance on unseen datasets, whereas the right four graphs illustrate the performance on datasets encountered during the pre-training phase.

Table 12: Experimental results on the modified and original contrastive loss function. The table presents the GDP-solving accuracy (%) with confidence intervals ($\alpha = 0.05$) for the graph models and the satisfiability prediction accuracy (%) with confidence intervals ($\alpha = 0.05$) for the SAT models. 'Graph/SAT Model+ConRep4CO+Modified Loss' denotes training with the modified contrastive loss. 'SAT Back.' refers to SAT model backbone, and 'Graph Back.' denotes graph model backbone.

SAT Back.	Graph Back.	Difficulty	Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
LCG+NeuroSAT	GCN	Easy	Graph Model+ConRep4CO+Modified Loss	77.1±0.3	57.9±0.2	61.5±0.3	88.7±0.1	64.2±0.4	71.5±0.2	64.4±0.4	69.3
			Graph Model+ConRep4CO	**79.3±0.3**	**62.0±0.1**	**67.3±0.2**	**90.2±0.1**	**67.5±0.5**	**71.7±0.3**	**65.4±0.3**	**71.9**
		Medium	Graph Model+ConRep4CO+Modified Loss	70.7±0.3	63.0±0.4	61.2±0.3	79.8±0.4	58.9±0.2	72.4±0.2	63.7±0.4	67.1
			Graph Model+ConRep4CO	**71.3±0.5**	**64.6±0.2**	**63.3±0.3**	**82.2±0.2**	**64.0±0.1**	**72.8±0.4**	**65.7±0.4**	**69.1**
		Easy	SAT Model+ConRep4CO+Modified Loss	98.3±0.2	99.6±0.1	99.9±0.1	98.5±0.2	98.1±0.3	99.9±0.1	99.9±0.1	99.2
			SAT Model+ConRep4CO	**98.9±0.2**	99.6±0.1	99.9±0.1	**98.8±0.3**	**98.9±0.4**	99.9±0.1	99.9±0.1	**99.4**
		Medium	SAT Model+ConRep4CO+Modified Loss	90.7±0.5	99.1±0.1	99.5±0.1	92.3±0.4	91.7±0.4	99.9±0.1	99.9±0.1	96.0
			SAT Model+ConRep4CO	**92.3±0.4**	99.1±0.1	**99.6±0.2**	**94.6±0.4**	**93.0±0.5**	99.9±0.1	99.9±0.1	**96.9**

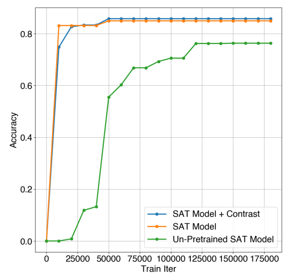
Table 13: Experimental results across two graph models under different training methods. 'Graph Model (fully-trained)' refers to the graph model that was trained from scratch with full training data. 'Graph Model+ConRep4CO (fine-tuned)' refers to the fine-tuned graph model after pre-training by ConRep4CO on small datasets.

Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
Graph Model (fully-trained)	67.3±0.4	66.7±0.2	65.4±0.4	79.1±0.2	59.1±0.3	72.4±0.1	65.4±0.2	67.9
Graph Model+ConRep4CO (fine-tuned)	**67.9±0.2**	**67.0±0.1**	**66.6±0.4**	**79.4±0.2**	**61.5±0.4**	**72.6±0.1**	**65.7±0.2**	**68.7**

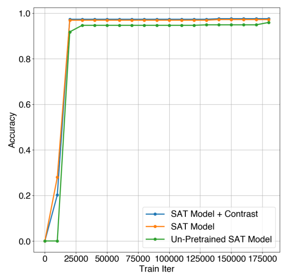
process in Fig. 4, all of which exhibit smooth trajectories. These results suggest that the influence of false negative samples on model performance is minimal.

G.2 Further Study on Graph Model Generalization to Large-Scale Data

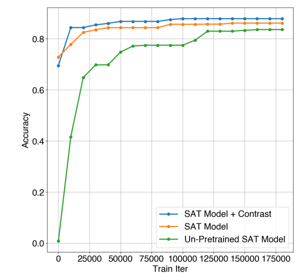
To further assess the generalization ability of our graph models, we generate large-scale instances for each GDP, with instance sizes ranging from 7 to 20 times larger than those used during pre-training. We then fine-tune the pre-trained models on this large-scale data, using a subset comprising $\frac{1}{8}$ of the training data. We compare the performance of the fine-tuned models with those trained from scratch with full training data, and the results are presented in Table 13, indicating that models pre-trained on smaller instances using ConRep4CO can generalize effectively to larger instances through fine-tuning.



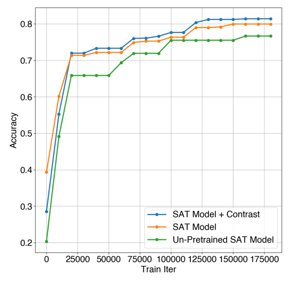
(a) Assign on k-Clique



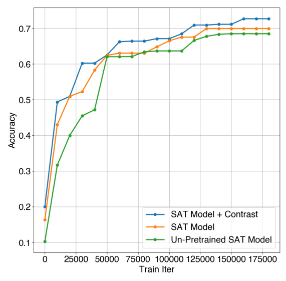
(b) Assign on k-Domset



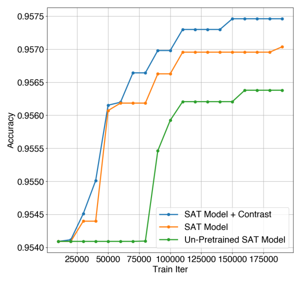
(c) Assign on Matching



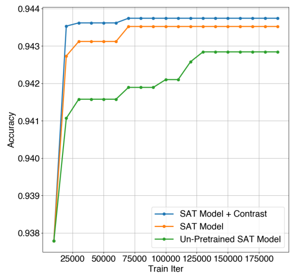
(d) Assign on PS



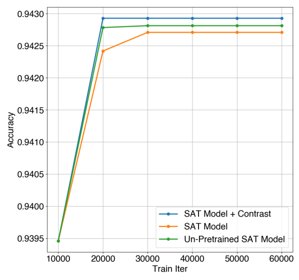
(e) Assign on SR



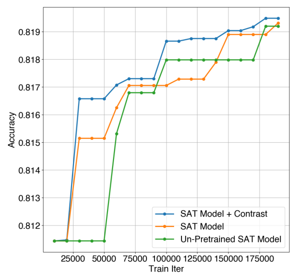
(f) Core Var on k-Domset



(g) Core Var on k-Vercov



(h) Core Var on 3-SAT

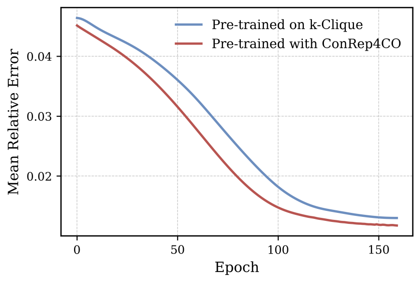


(i) Core Var on CA

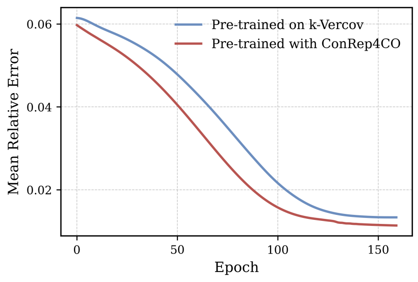
Figure 6: Model performance w.r.t. training iterations on SAT-based tasks across various datasets. Assign denotes the satisfying assignment prediction task, and Core Var denotes the unsat core variable prediction task.

G.3 Further Study on Graph Model Generalization to Other Graph Tasks

To further assess the generalization ability of the graph models, we conduct experiments on two GDP-related tasks: **maximum clique size prediction** adapted from k-Clique problem domain and **minimum vertex number prediction for edge cover** adapted from k-Vercov problem domain. We first pre-train the models on the original problem domains and with ConRep4CO, respectively. The pre-trained models are then fine-tuned with $\frac{1}{8}$ of the training data. To compare the performance, we employ the mean relative error (MRE) as the metric: MRE $= \frac{1}{N}\sum_{i=1}^{N}|\frac{y_i-\hat{y}_i}{y_i}|$, where y_i refers to the ground truth, $\hat{y}_i$ refers to the predicted value, and N refers to the sample size. Figure 7 illustrates that the pre-trained models with ConRep4CO achieve faster convergence and superior final performance, underscoring their enhanced generalization ability.



(a) Max Clique Size Prediction



(b) Min Vertex Num. Prediction

Figure 7: Mean relative error w.r.t. epoch on related graph tasks, including maximum clique size prediction and minimum vertex number prediction for edge cover.

Table 14: Experimental results on perturbed instances. The table presents the GDP-solving accuracy (%) with confidence intervals ($\alpha = 0.05$) for the graph models and the satisfiability prediction accuracy (%) with confidence intervals ($\alpha = 0.05$) for the SAT models on perturbed instances. 'SAT Back.' refers to SAT model backbone, and 'Graph Back.' denotes the graph model backbone.

SAT Back.	Graph Back.	Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
LCG+NeuroSAT	GCN	Graph Model	65.2±0.4	51.1±0.3	53.4±0.5	61.4±0.3	53.6±0.4	58.8±0.6	37.7±0.5	54.5
		Graph Model+ConRep4CO	**67.8±0.3**	**54.7±0.2**	**61.8±0.4**	**71.9±0.3**	**66.4±0.2**	**65.6±0.1**	**42.1±0.4**	**61.5**
		SAT Model	97.6±0.3	92.3±0.2	98.2±0.4	93.3±0.1	97.1±0.2	85.4±0.2	92.3±0.3	93.7
		SAT Model+ConRep4CO	**98.3±0.2**	**94.0±0.3**	**99.7±0.2**	**93.9±0.3**	**98.4±0.4**	**86.1±0.1**	**93.9±0.4**	**94.9**

Table 15: GDP solving accuracy (%) with confidence intervals ($\alpha = 0.05$) of the graph models on Easy datasets. The 'Overall' column represents the average accuracy across all datasets.

Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
Graph Model-FullData	77.8±0.1	59.0±0.2	61.4±0.3	87.8±0.4	63.2±0.1	71.3±0.1	64.3±0.3	69.3
Graph Model+ConRep4CO	**79.3±0.3**	**62.0±0.1**	**67.3±0.0**	**90.2±0.1**	**67.5±0.1**	**71.7±0.0**	**65.4±0.4**	**71.9**

G.4 Further Study on Model Sensitivity

The solution to GDP is known to be sensitive to graph structures. Therefore, we aim to evaluate the sensitivity of our model to perturbations in graph structure. To do so, we generate modified instances by adding or removing edges from the original graphs until either the satisfiability status reverses or the number of modified edges reaches $\frac{1}{10}$ of the original edge count. These generated instances are structurally similar to the original graphs but exhibit a reversed satisfiability status. We then assess the performance of both the graph models and the SAT model on these perturbed instances. The results, presented in Table 14, reveal that the SAT model is sensitive to changes in graph structure, and it continues to perform well. Additionally, the graph models significantly outperform the baseline models, as they are more closely aligned with the SAT model and demonstrate enhanced sensitivity to structural changes.

G.5 Further Study on Data Volume

During the pre-training phase, ConRep4CO utilizes all instances from different domains, while the baselines only have access to the instances from their single domain. To further assess the impact of data volume, we increase the number of training instances for baselines, and train 7 baseline graph models with the GCN backbone separately, each on $7 \times 160{,}000$ graph instances generated from a single problem type, denoted as **Graph Model-FullData**, and compare their performance with our approach in Table 15. It proves that the improved performance is not from increased training data.

G.6 Further Study on Multiple Domain Information Transfer

This section investigates the impact of multi-domain information transfer on model performance. We address two key questions: 1) **What is the effect of transferring information from multiple**

Table 16: Problem domains utilized for pre-training.

Problem	Ours-1	Ours-2	Ours-3	Ours-4	Ours-5	Ours-6	Ours-7
MVC	k-Vercov	k-Vercov, k-Indset	k-Vercov, k-Indset, k-Clique	k-Vercov, k-Indset, k-Clique, k-Domset	Ours-4+Automorph	Ours-5+Matching	Ours-6+k-color
MIS	k-Indset						
MC	k-Clique	k-Clique, k-Domset	k-Clique, k-Domset, k-Vercov				
MDS	k-Domset						

Table 17: The effect of the number of pre-training domains on downstream performance. Values in parentheses represent the gain over baseline, calculated as in the main text.

Problem	Graph	Optimal	Baseline	Ours-1	Ours-2	Ours-3	Ours-4	Ours-5	Ours-6	Ours-7
MVC	ER(50,100)	54.62	55.87	55.75 (9.60%)	55.39 (38.40%)	55.07 (64.00%)	54.76 (88.80%)	54.74 (90.40%)	54.68 (95.20%)	54.70 (93.60%)
MVC	ER(100,200)	122.79	126.04	125.86 (5.54%)	125.10 (28.92%)	124.89 (35.38%)	124.51 (47.08%)	124.44 (49.23%)	124.39 (50.77%)	124.37 (51.40%)
MVC	ER(400,500)	417.42	420.51	420.40 (3.56%)	419.86 (21.04%)	419.63 (28.48%)	419.49 (33.01%)	419.39 (36.25%)	419.33 (38.19%)	419.31 (38.83%)
MIS	RB(200,300)	20.10	19.18	19.22 (4.35%)	19.47 (31.52%)	19.52 (36.96%)	19.55 (40.22%)	19.53 (38.04%)	19.57 (42.39%)	19.56 (41.30%)
MIS	RB(800,1200)	43.15	37.48	37.62 (2.47%)	38.47 (17.46%)	38.66 (20.81%)	38.74 (22.22%)	38.79 (23.10%)	38.77 (22.75%)	38.79 (23.10%)
MC	RB(200,300)	19.05	16.24	16.41 (6.05%)	16.90 (23.49%)	17.23 (35.23%)	17.36 (39.86%)	17.44 (42.70%)	17.47 (43.77%)	17.47 (43.77%)
MC	RB(800,1200)	33.89	31.42	31.49 (2.83%)	31.75 (13.36%)	31.90 (19.43%)	32.07 (26.31%)	32.11 (27.94%)	32.12 (28.34%)	32.14 (29.15%)
MDS	RB(200,300)	27.89	28.61	28.56 (6.94%)	28.42 (26.39%)	28.28 (45.83%)	28.24 (51.39%)	28.18 (59.72%)	28.18 (59.72%)	28.19 (58.33%)
MDS	RB(800,1200)	103.80	110.28	110.04 (3.70%)	109.12 (17.90%)	108.68 (24.69%)	108.34 (29.94%)	108.22 (31.79%)	108.17 (32.56%)	108.19 (32.25%)

domains to downstream CO tasks? 2) **How does this effect scale with the number of source domains?** This analysis aims to provide a comprehensive evaluation of our proposed cross-domain transfer mechanism.

We design seven pre-training configurations, utilizing data from 1 to 7 distinct problem domains, denoted as **Ours-1** to **Ours-7**. The specific domains used in each setting are detailed in Table 16. Each pre-trained SAT model is subsequently used to enhance problem-specific neural solvers via contrastive learning. We employ OptGNN [54] as the baseline solver for MVC and GFlowNet [56] for MIS, MC, and MDS. The alignment between the pre-trained SAT models and the solvers is performed on 5,000 easy-level instances, with an additional 1,000 instances for validation. The subsequent training procedures for the solvers remain consistent with their original implementations [54; 56].

The results are presented in Table 17. The **Ours-1** configuration, which does not leverage cross-domain information transfer, shows a modest performance gain of approximately 5%. This result primarily reflects the benefit of the SAT transformation itself and suggests that pre-training on a single domain provides limited representational enhancement. As the number of pre-training domains increases, the performance gain grows in a sublinear fashion. The improvement is sharpest when expanding from one to two domains, after which the rate of gain decelerates, converging at around five or six domains. This saturation effect is likely attributable to diminishing returns in novel information.

G.7 Empirical Comparison with Multi-task Baselines

G.7.1 Comparison with UniCO [34]

[34] proposes a method that converts four CO problems into a general TSP formulation, then transforms the TSP solution back to solve the original problem. The authors introduce two network architectures, MatPOENet and MatDIFFNet, and explore training these models on a mixture of four different problems. In contrast to our approach, [34] does not introduce innovations in the training approach; their work is thus largely orthogonal to ours, making a perfectly matched comparison challenging to design.

To facilitate as fair a comparison as possible with [34], we select two general TSP tasks from their work—non-metric Asymmetric TSP (ATSP) and 2D Euclidean TSP (2DTSP)—and use MatPOENet as the backbone model. We compare three distinct training strategies:

1) **MatPOENet-Single**: The model is trained on a single task.

2) **MatPOENet-Mixed-Tuned**: The model is first trained on a mixture of four tasks from [34] and then fine-tuned on the single target task. To isolate the effect of the training strategy from the influence of data volume, the amount of data used for fine-tuning is kept identical to that used for single-task training. Thus, any performance difference can be attributed to the multi-task training paradigm.

Table 18: Average tour length with different training approaches. The notation 'ATSP20' denotes instances of ATSP with 20 nodes; other abbreviations follow the same convention.

Model	**ATSP20**	**ATSP50**	**ATSP100**	**2DTSP20**	**2DTSP50**	**2DTSP100**
MatPOENet-Single	1.5784	1.5864	1.6139	3.8427	5.7345	8.0972
MatPOENet-Mixed-Tuned	1.5778	1.5870	1.6143	3.8419	5.7342	8.1007
Ours	**1.5692**	**1.5809**	**1.6098**	**3.8368**	**5.7296**	**8.0931**

3) **Ours**: We utilize the method from [34] to accomplish the transformation between 3SAT and the general TSP format (Hamiltonian Cycle Problem, HCP). We then generate a dataset to align our SAT model (pre-trained on easy-level instances from the 7 domains in the main text) with the MatPOENet backbone through contrastive learning. This alignment dataset consists of 10,000 training pairs (HCP instances with 20 nodes) and 1,000 validation pairs. After alignment, the model is trained using the same procedure and data amount as the single-task setting.

The training of all MatPOENet models follows the original paper, using 10,000 randomly generated instances per epoch for over 1,000 epochs. It is important to note that the data volume required for alignment is quite small compared to the total training data.

In line with [34], we report the average tour length (lower is better) in Table 18. The results indicate that the multi-task pre-training and fine-tuning strategy yields no significant performance difference compared to single-task training, suggesting that simply mixing data from different problems does not effectively enhance performance on individual tasks. In contrast, our method outperforms both baseline strategies, demonstrating its ability to leverage knowledge from multiple problem domains to learn improved representations that benefit performance on single tasks.

G.7.2 Comparison with GOAL [10]

GOAL [10] introduces a framework for multi-task learning that employs distinct input and output adapters for different CO problems. The authors train a single model on a mixture of eight CO problems, enabling it to solve multiple tasks without fine-tuning, albeit with performance inferior to that of models trained on individual tasks. Similar to UniCO, the GOAL framework does not innovate on the training approach, making direct comparison with our approach infeasible using their original setup.

To enable a fair comparison, we select three tasks from [10]—ATSP, MVC, and MIS—and use the GOAL framework as the backbone solver. We compare the following three training strategies:

1) **GOAL-Single**: The model is trained exclusively on a single task.

2) **GOAL-Multi-Tuned**: The model is first trained on a mixture of eight tasks from [10] (with MIS added as a ninth task for the MIS experiments) and is subsequently fine-tuned on the single target task. To mitigate the influence of data volume, the fine-tuning dataset is kept identical in size to the single-task training set, ensuring that performance differences are attributable to the multi-task training paradigm.

3) **Ours**: We first align our SAT model (pre-trained on easy-level instances from the 7 domains listed in the main text) with the GOAL backbone via contrastive learning. For ATSP, the alignment data is the dataset generated for the UniCO comparison; for MVC and MIS, we use 10,000 easy-level k-VerCov and k-IndSet instances for training, respectively, with an additional 1,000 for validation. After alignment, the model is trained using the same procedure and data volume as the single-task setting.

The data volume for training GOAL is consistent with the original paper, i.e., 1 million random instances per problem.

Following [10], we report the average gap (lower is better) in Table 19. The results indicate that, similar to the UniCO findings, simply mixing training data from multiple tasks does not effectively enhance performance on individual target tasks. In contrast, our contrastive learning-based training strategy outperforms both baselines, demonstrating the utility of incorporating contrastive learning into a multi-task framework.

Table 19: Average gap with different training approaches. The notation 'ATSP100' denotes instances of ATSP with 100 nodes; other abbreviations follow the same convention.

Model	ATSP100	MVC100	MIS100
GOAL-Single	0.32%	0.21%	0.16%
GOAL-Multi-Tuned	0.30%	0.22%	0.15%
Ours	**0.25%**	**0.17%**	**0.13%**

Table 20: Performance on large-scale CO problems. Values in parentheses represent the gain over baseline, calculated as in the main text.

Problem	Graph	Optimal	Baseline	Ours-Easy	Ours-Medium	Ours-Hard
MVC	ER(600,1000)	798.54	806.81	805.36 (17.53%)	804.83 (23.94%)	**803.99 (34.10%)**
MVC	ER(1000,2000)	1320.78	1331.19	1329.75 (13.83%)	1328.48 (26.03%)	**1327.90 (31.60%)**
MVC	ER(2000,3000)	2476.30	2498.79	2496.46 (10.36%)	2493.87 (21.88%)	**2492.07 (29.88%)**
MIS	ER(9000,11000)	381.31	356.47	358.73 (9.10%)	360.36 (15.66%)	**362.11 (22.71%)**

G.8 Further Study on Large-Scale CO Problems

To comprehensively evaluate the efficacy of our method on real-world CO problems, we conduct a series of experiments on large-scale instances. Based on our initial observation that pre-training solely on easy-level instances leads to performance gains that diminish with increasing problem scale, we further analyze this trend of gain degradation and investigate the scale of pre-training data required to effectively mitigate it.

Specifically, we select larger-scale instances of MVC, namely Erdős–Rényi (ER) graphs with 600–1000, 1000–2000, and 2000–3000 vertices, denoted as ER(600, 1000), ER(1000, 2000), and ER(2000, 3000), respectively. Additionally, we test on large-scale MIS instances represented by ER graphs with 9000–11000 vertices, i.e., ER(9000, 11000). We employ OptGNN [54] as the baseline solver for MVC and GFlowNet [56] for MIS. The training procedures for these baseline solvers are kept consistent with their respective original papers [54; 56]. For MVC, 32 test instances are used for each graph size, while 16 test instances are used for the MIS experiments. The optimal for MVC is acquired through Gurobi with a 24-hour time limit, and for MIS, it is obtained by KAMIS.

Beyond the baseline, we evaluate three variants of our method pre-trained on data of different scales: **Ours-Easy** (the reported results in the main text), pre-trained on easy-level instances (graphs with 5-15 vertices); **Ours-Medium**, pre-trained on medium-level instances (graphs with 10-20 vertices for MVC, 15-20 vertices for MIS); and **Ours-Hard**, pre-trained on hard-level instances (graphs with 15-25 vertices for MVC, 20-25 vertices for MIS). For each variant, 5,000 instances are utilized for training, with an additional 1,000 instances for validation. The subsequent fine-tuning procedures for these variants remain identical to the baseline training.

The results are presented in Table 20. The rate at which performance gains diminish with problem scale is significantly reduced when the pre-training scale is increased. Notably, effective mitigation does not require pre-training on graphs of massive scale (e.g., with 10,000 vertices). Pre-training on hard-level instances (comprising graphs with up to 25 vertices) alone is sufficient to stabilize the performance gains; our method still achieves over 20% gain even on the very large ER(9000, 11000) graphs. This finding robustly demonstrates the potential of our approach for enhancing real-world, large-scale CO problem-solving.

G.9 Ablation Study on Warm Start

During training, we incorporate a warm start phase. To better understand its contribution, we conduct an ablation study. We selected the LCG+NeuroSAT+GraphSAGE backbone and, during training on easy-level instances, omitted the warm start phase while keeping all other procedures unchanged. This variant is denoted as **w/o warm start** and is compared against the standard method (**with warm start**).

The results, presented in Table 21, show a significant performance degradation when the warm start is skipped. This decline is because applying contrastive learning from the very beginning can disrupt the

Table 21: Graph model performance comparison with and without warm start.

Model	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall
w/o warm start	71.4	61.9	65.3	90.4	71.1	65.9	62.6	69.8
with warm start	**79.7**	**63.2**	**70.8**	**93.3**	**75.3**	**71.0**	**63.9**	**73.9**

model's acquisition of task-specific representations. Consequently, the information transferred during the subsequent phase may carry greater bias, which is detrimental to representation enhancement. While contrastive learning is designed to facilitate information transfer, the warm start phase ensures that the information being shared across different problems is meaningful. Therefore, the warm start plays a critical role in the effectiveness of our method.

G.10 Ablation Study on Temperature Selection

We conducted an ablation study on the temperature hyperparameter (τ) in our contrastive learning framework. Using the LCG+NeuroSAT+GraphSAGE backbone, we train models on both easy-level and medium-level instances while varying the value of τ from 0.1 to 0.8. The results are presented in Table 22 and Table 23, indicating that the specific value of τ has a limited impact on the final performance. However, it does influence the convergence rate of the contrastive learning process.

Table 22: Performance of graph models trained using different temperature (τ). The models are trained on easy-level instances. 'Converged Epoch' denotes the training epoch where the contrastive loss ceases to decrease.

Temperature	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall	Converged Epoch
0.1	79.7	63.2	70.8	93.3	75.3	71.0	63.9	73.9	16
0.2	79.5	63.0	70.8	93.2	75.0	70.7	63.9	73.7	18
0.3	79.6	63.2	70.7	93.2	75.2	71.1	63.7	73.8	20
0.4	79.7	63.1	70.8	93.5	75.5	71.2	63.9	74.0	19
0.5	79.7	63.1	70.6	93.1	75.4	70.9	63.9	73.8	19
0.6	79.9	63.1	70.9	93.1	75.4	71.0	64.2	73.9	20
0.7	79.7	63.2	70.9	93.3	75.4	71.0	63.8	73.9	22
0.8	79.6	63.2	70.8	93.2	75.1	71.0	63.6	73.8	24

Table 23: Performance of graph models trained using different temperature (τ). The models are trained on medium-level instances. 'Converged Epoch' denotes the training epoch where the contrastive loss ceases to decrease.

Temperature	k-Clique	k-Domset	k-Vercov	k-Color	k-Indset	Matching	Automorph	Overall	Converged Epoch
0.1	72.6	64.2	66.5	86.0	70.0	71.7	65.0	70.9	24
0.2	72.4	64.5	66.6	85.8	70.2	71.5	64.8	70.8	23
0.3	72.9	64.0	66.8	85.9	69.9	71.6	64.7	70.8	21
0.4	72.7	64.0	66.7	86.0	70.4	72.0	64.9	71.0	18
0.5	72.8	64.1	66.7	85.9	70.1	71.7	64.8	70.9	18
0.6	72.7	64.0	66.7	85.7	70.0	71.8	64.7	70.8	20
0.7	72.5	64.1	66.5	85.8	70.0	71.6	64.6	70.7	23
0.8	72.6	64.1	66.5	85.9	70.1	71.6	64.8	70.8	24

H Large Language Model Usage

In this paper, large language models (LLMs) are only used to find and correct grammatical errors.