Contrastive Learning for Combinatorial Optimization

Mohamed Elgharbawy

May 12, 2023

Contrastive Learning for Combinatorial Optimization

by Mohamed Elgharbawy

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Joseph E. Gonzalez

Professor Joseph Gonzalez
Research Advisor

5/9/2023

(Date)

* * * * * * *

[illegible]

Professor Sanjit A. Seshia
Second Reader

5/12/2023

(Date)

Contrastive Learning for Combinatorial Optimization

by

Mohamed Elgharbawy

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Spring 2023

Abstract

Contrastive Learning for Combinatorial Optimization

by

Mohamed Elgharbawy

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Joseph E. Gonzalez, Chair

This paper presents the use of contrastive learning to improve upon the performance of Optimization Modulo Theories (OMT) solvers. OMT is a generalization of Satisfiability Modulo Theories (SMT) that requires an objective function and has numerous real-world applications, such as chip placement, worst-case execution analysis, and the Traveling Salesman Problem. OMT problems are challenging due to the requirement of finding both feasible and optimal solutions, as well as the non-convexity of constraints, making it difficult to identify the global optimal solution. The current approaches taken by OMT solvers are not problem-specific and involve reducing the problem into Integer Linear Programs (ILPs), or repeatedly making calls to SMT solvers. The presented contrastive learning approach leverages symmetries and invariances within OMT problems through optimality and feasibility-preserving transformations to better guide the search for optimal solutions. This paper builds upon Ashera, a learning-based OMT solver, and implements contrastive learning to improve downstream optimal variable assignment accuracies by over 6% on a Scheduling problem benchmark and over 5% on a Multi-Agent Traveling Salesman problem benchmark.

Contents

List of Figures

List of Tables

Acknowledgments

I would first like to thank my family for their constant support, without which my academic endeavors would not be possible. I would like to express my gratitude to Professor Joseph Gonzalez for advising me and giving me the opportunity to explore research at UC Berkeley. I am also grateful to Professor Sanjit A. Seshia for providing feedback on my thesis. I also wish to thank Justin Wong for allowing me to join his research group as an undergraduate student, which has allowed me to continue my research through my graduate career, and for constantly guiding me both as a mentor and a friend. I would also like to thank my friend and fellow 5th year masters student Chirag Sharma for helping me with my research along the way. Finally, thank you to all of my friends, colleagues, and professors at Berkeley for shaping and supporting my academic career.

Chapter 1

Introduction

Optimization Modulo Theories (OMT) are a generalization of Satisfiability Modulo Theories (SMT) [1] that require an objective function. OMT solvers have numerous real-world applications, such as chip placement [2], worst-case execution analysis [3], and the Multi-Agent Traveling Salesman Problem [4]. By requiring that OMT solutions are both feasible and optimal, finding such solutions become strictly more difficult than in SMT problems. In OMT problems, the feasible region of solutions is often defined by a set of constraints and bounds, which further complicates the search for optimal solutions. Furthermore, the non-convexity of the constraints in OMT problems can result in multiple local optima, making it challenging to identify the global optimal solution. Current approaches taken by OMT solvers either involve reducing the problem into Integer Linear Programs (ILPs) to approximate a solution, or repeatedly making calls to SMT solvers. These techniques are slow to find an optimal solution when given constraints, taking exponential time, and are not meant to be problem-specific. Despite these challenges, advancements in optimization algorithms and techniques have enabled the efficient solving of some OMT problems [5], making it a promising area for research and practical applications.

Current SMT solvers perform quite well in the industry on problems such as chip placement and route scheduling, but when given an objective function, solvers such as OptiMathSAT [6] fail to reach the same industry-leading performance. As a result, ILP approaches are used in the industry to approximate solutions. However, to achieve reasonable performance, solvers are in practice tuned against benchmarks of related problems, making them less versatile than other optimization techniques. Additionally, the accuracy of SMT solvers heavily depends on the problem representation and formulation, which can be a challenging task for complex optimization problems. Despite efforts to improve them, SMT solvers still struggle with large-scale problems and tend to be computationally expensive.

General-purpose OMT solvers are efficient at finding solutions, but in practice most applications require a solver that is specifically tuned for their specific problem setup. The performance of an OMT solver can vary greatly depending on factors such as the size of the problem, the complexity of the constraints, and the available computational resources. Therefore, it is important to have a solver that is efficient and tailored to the specific problem at hand. For instance, a company in Berkeley that is trying to analyze the most cost-efficient route of their package deliveries may want an OMT solver that is not only efficient at the Traveling Salesman Problem (TSP), but also specifically tuned to be as efficient as possible in the city of Berkeley. This has motivated the use of machine learning (ML) to learn useful heuristics for solving such problems.

Our key insight is to further refine this learning by allowing the learned heuristic to effectively be tuned at a per-instance level. Another key insight is that current approaches do not analyze symmetries and invariances specific to each problem class. Exploiting symmetries in SMT problems has been shown to be successful in areas such as constraint programming and SAT solving [7]. This is due to the fact that symmetries allow a data point to become representative of far more data points, which allows us to compare and detect patterns in a lower-dimensional manifold. Hence, contrastive learning, an approach frequently used with images to learn attributes that are common between data classes, can be used in the context of OMT problems to leverage symmetries and invariances within problems to better improve the performance of current OMT solver approaches. One of the key advantages of contrastive learning is that it can help identify the underlying structures and patterns in the data, which can be used to guide the search for optimal solutions in OMT problems. By learning the common attributes and features shared between different classes of OMT problems, contrastive learning can help build more generalized and adaptive OMT solvers that can perform well across a wide range of problem instances.

Our work makes the following contributions:

- We build upon Ashera [8], a learning-based OMT solver
- We implement contrastive learning to improve downstream optimal variable assignment accuracies by over 6% on a Scheduling problem benchmark, and over 5% on a Multi-Agent Traveling Salesman problem benchmark

The rest of the paper is organized as follows: In chapter 2 we introduce necessary background and related work. In chapter 3 we elaborate on our approach and experimental design. In chapter 4, we present our experiments and results. Lastly, we conclude and present future work in chapter 5.

Chapter 2

Background and Related Work

2.1 Optimization Modulo Theories

Optimization Modulo Theories (OMT) are a generalization of Satisfiability Modulo Theories (SMT) that aim to satisfy constraints while optimizing an objective function. SMT, a generalization of Boolean Satisfiability Problems (SAT), is a problem that decides if a set of constraints are satisfiable. While SAT problems only use boolean logic, SMT problems generalize to other data types, such as real numbers, integers, data structures, and more. In this work, we focus on Linear Integer Arithmetic (LIA) [9] problems, as they guarantee decidability.

We define a *constraint* to be an inequality of the form:

$$constraint(\vec{x}) = \vec{c} \cdot \vec{x} \bowtie \vec{b} \tag{2.1}$$

where $\vec{x} \in \{0,1\}^n$, $\vec{c} \in \mathbb{Z}^n$, $\bowtie \in \{<, \leq, =, \neq, \geq, >\}$, and $\vec{b} \in \mathbb{R}$. n denotes the number of variables within the constraint.

Next, we define a *conjunction* as a logical *and* of constraints:

$$conjunction(c_1, ..., c_n) = \bigwedge_{i=0}^{n} c_i \tag{2.2}$$

where c_i is a *constraint*.

Following this, we define a *clause* in the disjunctive normal form (DNF), which is a logical *or* of our conjunctions:

$$clause(c_1, ..., c_n) = \bigvee_{i=0}^{n} c_i \tag{2.3}$$

where c_i is a *conjunction*.

Next, we define a *formula*, which is a logical *and* of our *clauses*:

$$formula(c_1, ..., c_n) = \bigwedge_{i=0}^{n} clause(c_1, ..., c_n) \tag{2.4}$$

where c_i is a *clause*.

Following this, we define our *SMT* problem:

$$\exists \, \vec{x} \text{ s.t. } formula(x_1, ..., x_n) = True \tag{2.5}$$

Finally, we define our *OMT* problem as an *SMT* problem with an objective:

$$\begin{aligned} \min_{\vec{x} \in \mathbb{R}^n} \; & C(\vec{x}) \\ \text{s.t. } & formula(\vec{x}) = True \end{aligned} \tag{2.6}$$

where $C(\vec{x})$ is our objective function.

2.2 Graph Neural Networks for Combinatorial Optimization

Graph Neural Networks (GNNs) are a class of neural networks that are able to process data in the form of graphs. In these graphs, the nodes represent entities, while the edges represent the relationships between them. GNNs learn a set of functions to propagate information between nodes in a graph to update their representations. GNNs have numerous applications, such as recommendation systems [10], drug discovery [11], and social network analysis [12].

Graph Convolutional Neural Networks (GCNs) [13] are a form of GNNs that can be interpreted as a generalization of Convolutional Neural Networks (CNNs). GCNs use convolutional operations to aggregate information from a node's neighbors to update the node's representation. GCNs and variations upon its implementation have been shown to have state-of-the-art performance on graph-related tasks and can be computationally efficient [14].

The application of GNNs and GCNs has been a relatively recent breakthrough in the approach of solving combinatorial optimization problems [15]. GNNs have an inductive bias that allows them to encode combinatorial and relational input effectively, as they are permutation-invariant and can handle input sparsity. In practice, GNNs and GCNs have been shown to be effective at chip placement [16], where the objective is to optimize the power, performance, and area of a chip by mapping the nodes of a netlist (which represents the desired chip) onto a bounded 2D space called a chip canvas. Such innovations show promise in the field of using GNNs to tackle combinatorial optimization, but only begin to explore its potential.

2.3 Label-Preserving Augmentations

There are label-preserving augmentations (LPAs) that can be applied to SAT and SMT problems [17] that maintain both satisfiability and unsatisfiability, as well as optimality:

Positive/Negative Scalar Multiplier (PM/NM). Multiplying a *constraint* by a positive scalar multiplier does not change its satisfiability nor its optimal solution. Multiplying a *constraint* by a negative scalar multiplier has the same property, provided the operator $\bowtie$ is negated (e.g. $\bowtie \in \{<, \leq, =, \neq, \geq, >\} \rightarrow \neg \bowtie \in \{\geq, >, \neq, =, <, \leq\}$, respectively).

Original	**PM**
min $2x_0 - 3x_1$ Subject to: $c_1 : (3x_0 + 2x_1 \leq 1) \vee (x_0 - 3x_1 > 2)$ $c_2 : (x_0 - 3x_1 \leq 2) \vee (5x_0 - x_1 > 4)$	min $2x_0 - 3x_1$ Subject to: $c_1 : (12x_0 + 8x_1 \leq 4) \vee (4x_0 - 12x_1 > 8)$ $c_2 : (2x_0 - 6x_1 \leq 4) \vee (10x_0 - 2x_1 > 8)$

Table 2.1: Positive Scalar Multiplier. c_1 is multiplied by 4, and c_2 is multiplied by 2.

Unit Propagation (UP). A unit clause contains only one literal. If a problem p contains a unit literal ℓ, we can remove all clauses in p containing ℓ and delete $\neg\ell$ from all other clauses.

Original	**UP**
min x_0 Subject to: $c_1 : x_0 \leq 0$ $c_2 : (x_0 > 0 \wedge x_1 \leq 0) \vee (x_0 \leq 0 \wedge x_1 \leq 3)$	min x_0 Subject to: $c_2 : x_1 \leq 0$

Table 2.2: Unit Propagation. c_1 is pure, so we delete c_1, as well as $(x_0 \leq 0 \wedge x_1 \leq 3)$ and $x_0 > 0$ from c_2.

Add Unit Literal (AU). This is the inverse of UP. Construct a new unit clause from a new literal ℓ, add $\neg\ell$ to some disjunctions, and add ℓ to conjunctions.

Original	**AU**
min x_0 Subject to: $c_1 : x_0 \leq 0$ $c_2 : (x_0 \leq 0 \land x_1 \leq 3)$	min x_0 Subject to: $c_1 : x_0 \leq 0$ $c_2 : (x_0 \leq 0 \land x_1 \leq 3 \land x_1 \leq 1) \lor (x_1 > 1)$ $c_3 : x_1 \leq 1$

Table 2.3: Add Unit Literal. We construct and add *constraint* $\ell = (x_1 \leq 1)$ inside the first conjunction of c_2, as well as a constraint c_3, and add $\neg\ell = (x_1 > 1)$ to the disjunction of c_2.

Subsumed Clause Elimination (SC) If a clause is a subset of another clause, i.e. $c_1 \subset c_2$, then removing c_2 will not change the satisfiability nor optimality of the problem.

Original	**SC**
min x_0 Subject to: $c_1 : x_0 \leq 0$ $c_2 : (x_0 \leq 0) \lor (x_0 \leq 2 \land x_1 \leq 3)$	min x_0 Subject to: $c_1 : x_0 \leq 0$

Table 2.4: Subsumed Clause Elimination. $c_1 \subset c_2$, so we eliminate c_2.

Clause Resolution (CR) Adding the resolvent of two clauses containing complementary ltierals (e.g. $(x_1 \lor x_2) \otimes (\neg x_2 \lor x_3) = (x_1 \lor x_3)$) does not change satisfiability or optimality.

Original	**CR**
min x_0 Subject to: $c_1 : x_0 \leq 0$ $c_2 : (x_0 > 0) \lor (x_1 \leq 2 \land x_2 \leq 3)$	min x_0 Subject to: $c_1 : x_0 \leq 0$ $c_2 : (x_0 > 0) \lor (x_1 \leq 2 \land x_2 \leq 3)$ $c_3 : (x_1 \leq 2 \land x_2 \leq 3)$

Table 2.5: Clause Resolution. Add $c_3 = c_1 \otimes c_2$.

2.4 Contrastive Learning

Contrastive learning [18] is a machine learning technique that aims to learn low-dimensional representations of data by contrasting between samples of the same and different classes.

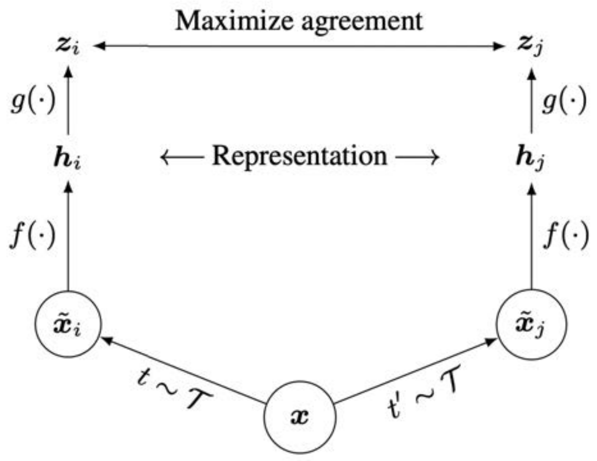


Figure 2.1: SimCLR Framework [19]. Transformations t and t' are applied to a data point x, passed through a base encoder network $f(\cdot)$, then through a projection head $g(\cdot)$. The output embeddings z_i and z_j are passed through a contrastive loss function to maximize agreement.

It aims to reduce the Euclidean distance between similar samples in the low-dimension representation space, while pushing dissimilar samples apart. This style of learning can both be supervised and unsupervised.

SimCLR [19] is a famous self-supervised framework for unsupervised contrastive learning. The framework creates positive image pairs through random transformations to the anchor image, such as cropping, flipping, and changing colors. These transformations create a diverse and large set of images for learning. SimCLR was able to improve upon previous state-of-the-art linear classifiers, matching the performance of a supervised ResNet-50. Further more, when fine-tuned on 1% of labels, SimCLR was able to outperform AlexNet with 100x fewer labels.

Chapter 3

Approach and Experimental Design

We seek to demonstrate that applying LPAs allows us to learn embeddings that better generalize. We desire to demonstrate that utilizing representation learning using contrast over transformations will lead to desirable embeddings for downstream tasks.

3.1 Graph Representation for OMT Problems

We translate an OMT problem into a graph using the methodology found in Gasse et al [20]. In each $formula$, we encode all of the variables, *conjunctions*, and *clauses*. Then, for each *constraint* $\vec{c} \cdot \vec{x} \bowtie \vec{b}$, we add an edge between each variable node x_i and the *constraint*, with weight a_i. $\vec{b}$ is encoded as an attribute for the *constraint* node. *Conjunctions* are encoded as a tree over their corresponding *constraints*. *Disjunctions* are encoded as a tree over the *conjunction* nodes. Finally, we have a cost node connected to all of the variable nodes with weights c_i.

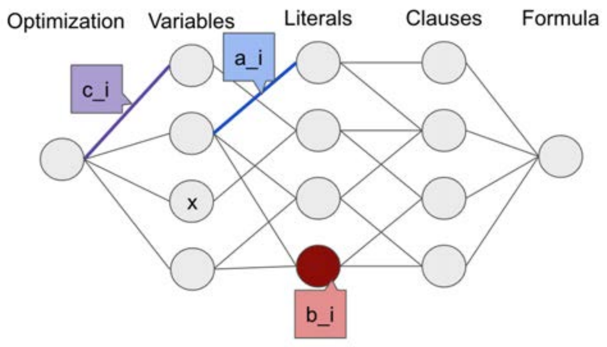


Figure 3.1: Graph representation. The OMT problem is represented a graph with edges connecting variable, constraint, and cost nodes.

3.2 Multi-Agent Traveling Salesman Problem

The Multi-Agent Traveling Salesman problem (MATSP) is a generalization of the famous Traveling Salesman Problem (TSP), where multiple salesmen need to visit a number of waypoints exactly once and return to their initial position, all while minimizing the sum of travel times t_i when a waypoint is visited:

$$\min \sum_{i=0}^{N} t_i \tag{3.1}$$

Due to space constraints, the following constraints are applied to our version of the problem:

- **Visited:** All waypoints must be visited by at least one vehicle.
- **Deterministic:** After visiting a waypoint, a vehicle visits at most one more waypoint immediately afterwards.
- **Ordering:** The starting waypoint s has an order $o_s = 0$. For all subsequent waypoints w visited in order o_w, the previously visited waypoint p_w must have order $o_{p_w} = o_w - 1$. This prevents tours that do not include the starting waypoint s.
- **Weight Constraint:** The sum of the weights of the vehicles is less than a given value M.
- **Visit Time:** For all waypoints w, the visit time t_w if vehicle v visits it is at least $t_{p_w} + \tau_{v,p,w}$, where t_{p_w} is the time when the preceding waypoint was visited and the τ is the travel time from p_w to w by vehicle v.
- **Exclusion:** If a vehicle v is traveling from w to w' from t_w to $t_{w'}$, there cannot be a waypoint w'' that is visited by v while it is traveling.

3.3 Scheduling Problem

The Scheduling problem involves finding an optimal placement of tasks to resources, as well as assigning start times to tasks. Additionally, there are constraints on resources and dependencies between tasks that must be satisfied. The goal is to maximize slack, which is the buffer time before a deadline that a task is expected to complete. Finding feasible solutions is non-trivial, and finding an optimal one is even more difficult.

Consider a set of N tasks, $T = \{t_i | i \in [1, N]\}$, and a dependency matrix M, where $M_{ij} = 1$ if t_i must complete before t_j, and 0 otherwise. Additionally, we have a set of deadlines and expected runtimes, d and e, respectively. We also have resources requirements and placements. Let $r_i = 1$ if t_i requires a GPU, and 0 otherwise.

We denote N_G and N_C as the number of GPUs and CPUs, respectively. We seek to optimize with respect to s_i and p_i, which denote the start time and placement of t_i, respectively. Let $p_i = k \in [1, N_G]$ if p_i is placed on the k^{th} GPU of N_G, and $(k - N_G)^{th}$ CPU if it is in $[N_G, N_G + N_C]$.

We seek to minimize the following cost objective:

$$\min \sum_{i=0}^{N} d_i - (s_i + e_i) \tag{3.2}$$

We enforce the following constraints:

- **Basic constraints:** $\forall i,\ s_i \geq 0$ and $p_i > 0$.
- **Finishing before the deadline:** $\forall i,\ s_i + e_i \leq d_i$.
- **Placement constraints:** $\forall i$, if $r_i = 1$, then $1 \leq p_i \leq N_G$. Otherwise, when $r_i = 0$, $1 \leq p_i \leq N_G + N_C$.
- **Dependency respecting:** $\forall i, j$, if $M_{ij} = 1$, $s_i + e_i \leq s_j$.
- **Exclusion:** $\forall i, j, p_i = p_j \Rightarrow (s_i + e_i \leq s_j \vee s_j + e_j \leq s_i)$.

3.4 Contrastive Learning

The goal of our contrastive learner is to exploit symmetries and invariances in OMT problems. Intuitively, this should be useful for both MATSP and Scheduling problems. In MATSP, cities being further apart by a constant factor, as well as cities in different orders should result in the same optimal solution. In the context of Scheduling, the order of the tasks/GPUs do not matter. Additionally, increasing the expected runtimes while proportionally delaying the the deadline should result in the same optimal solution.

Loss Function

To perform contrastive learning, we will apply the LPAs as described in section 2.3. Additionally, we will use the InfoNCE loss function [21]. Given a set $X = \{x_1, ..., x_N\}$ of N random samples containing one positive sample from $p(x_{t+k}|c_t)$ and $N-1$ negative samples from the proposal distribution $p(x_{t+k})$, we optimize:

$$\mathcal{L}_N = -\mathbb{E}_X \left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)} \right] \tag{3.3}$$

InfoNCE loss is a version of cross-entropy loss with respect to classifying the positive sample correctly, where $\frac{f_k}{\sum_X f_k}$ is the prediction of the model.

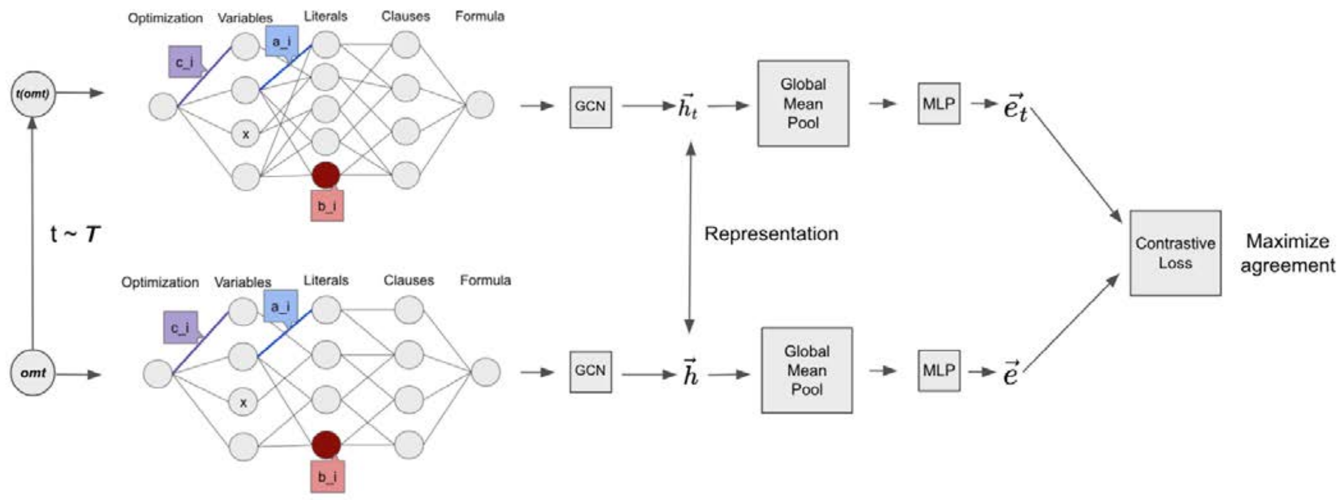


Figure 3.2: Contrastive Learning Architecture

Architecture

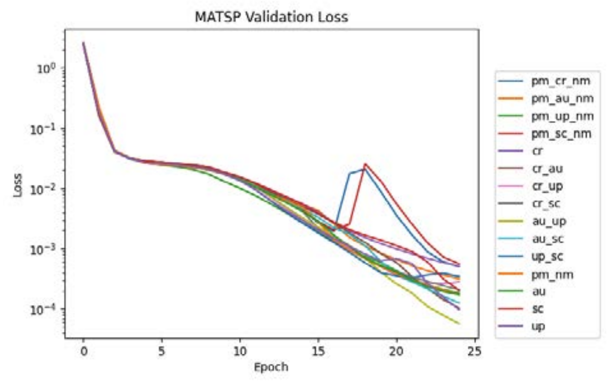
During training, an OMT problem is sampled from within a batch. A transformation t is sampled randomly from a pool of T transformations, and applied to the OMT problem to generate a positive key. Both problems are fed through a GCN model, which outputs hidden representations $\vec{h}$ and $\vec{h_t}$ of the variables within the problem. A global mean pool is then applied to $\vec{h}$ and $\vec{h_t}$, where they are then each fed through an MLP to output embeddings $\vec{e}$ and $\vec{e_t}$. Finally, these embeddings are passed through our contrastive loss function, InfoNCE, where the goal is to maximize their agreement.

The GCN consists of a convolutional layer, a ReLU activation function, and a second convolutional layer. The variable hidden representations are extracted from the output of the GCNs and are our $\vec{h}$ and $\vec{h_t}$. A global mean pool is then applied to each variable individually, and the results are concatenated and fed through an MLP. The MLP consists of a linear layer, a ReLU activation function, followed by a second linear layer. The outputs of the MLPs become our embeddings $\vec{e}$ and $\vec{e_t}$.

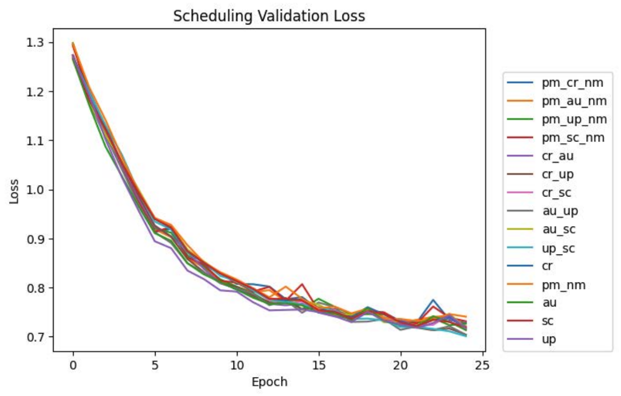
LPA Selection

The model was trained on a small subset of the MATSP and Scheduling data. Of the five different transformations mentioned in section 2.3 (positive/scalar multiplication is treated as a singular transformation class), $\binom{5}{2} = 10$ combinations were tested, alongside only applying one transformation at a time, for a total of 15 different tests per problem. Applying no transformations was used as a baseline, and resulted in a final epoch validation loss of 9.78×10^{-5} for MATSP, and 0.721 for Scheduling.

For MATSP, table 3.1 demonstrates that Unit Propagation alone, as well as combined



(a) MATSP Validation Loss (Log Scale)



(b) Scheduling Validation Loss

Figure 3.3: LPA Combination Tests on a Small Dataset

	pm/nm	**au**	**sc**	**up**	**cr**
pm/nm	1.74×10^{-4}	3.13×10^{-4}	5.58×10^{-4}	1.83×10^{-4}	5.03×10^{-4}
au	3.13×10^{-4}	1.72×10^{-4}	1.25×10^{-4}	5.71×10^{-5}	1.02×10^{-4}
sc	5.58×10^{-4}	1.25×10^{-4}	2.01×10^{-4}	3.44×10^{-4}	2.07×10^{-4}
up	1.83×10^{-4}	5.71×10^{-5}	3.44×10^{-4}	9.61×10^{-5}	2.81×10^{-4}
cr	5.03×10^{-4}	1.02×10^{-4}	2.07×10^{-4}	2.81×10^{-4}	5.05×10^{-4}

Table 3.1: MATSP transformation combinations on a small dataset. Each cell represents the validation loss on the final epoch of training. Green cells have a lower final validation loss than the baseline of 9.78×10^{-5}.

	pm/nm	**au**	**sc**	**up**	**cr**
pm/nm	0.741	0.721	0.731	0.727	0.728
au	0.721	0.713	0.717	0.704	0.719
sc	0.731	0.717	0.719	0.701	0.714
up	0.727	0.704	0.701	0.717	0.704
cr	0.728	0.719	0.714	0.704	0.718

Table 3.2: Scheduling transformation combinations on a small dataset. Each cell represents the validation loss on the final epoch of training. Green cells have a lower final validation loss than the baseline of 0.721.

with Add Unit Literal, were able to have a lower final epoch validation loss compared to the baseline metric. For Scheduling, table 3.2 demonstrates that the majority of transformations had a lower validation loss compared to the baseline metric. Positive/Scalar Multiplication seemed to perform equally or slightly worse across the board.

Hence, we will use Unit Propagation and Add Unit Literal for the experiments in chapter 4.

3.5 Learning Problem Formulation

Our downstream task for evaluating the performance of our model is to find an optimal assignment of variable values for a given OMT problem. Additionally, we would like to demonstrate that OMT problems and their transformations can be embedded into a lower-dimensional space through contrastive learning, such that problems of the same class are close to each other, while problems of different classes are further apart.

Given a graph encoding G, we would like to learn a function f that estimates a probability distribution over all potential values of each variable. Since we are in the realm of LIA problems, we treat integer variable values as independent classes, and train the model to classify each variable. We use a GCN to accomplish the goal of optimal assignments, as well as effective contrastive embeddings. Our GCN learns embeddings for each variable, and then splits into two different heads: one head to produce a value for each variable, and one head to output a pooled embedding over all of the variables, as detailed in section 3.4. We then optimize the following loss function:

$$\mathcal{L} = \underbrace{-\sum_{i=1}^{m} x_i^* \log p(x_i|G)}_{\text{Cross-entropy loss}} - \underbrace{\mathbb{E}_X \left[\log \frac{f_k(x_{t+k}, c_t)}{\sum_{x_j \in X} f_k(x_j, c_t)} \right]}_{\text{InfoNCE loss}} \tag{3.4}$$

The first term represents the cross-entropy loss of our variable value prediction, where x_i^* is the optimal assignment of the i-th variable and $\log p(x_i|G)$ is the probability of the assignment being generated by the function f. The second term represents the InfoNCE loss as detailed in section 3.4. Combined, the goal of this loss function is to improve upon the first goal of predicting optimal assignments by optimizing over our contrastive variable embeddings.

Chapter 4

Experiments and Results

For each problem, we compare the training loss, validation loss, and downstream optimal variable assignment prediction accuracy between the baseline model and the contrastive model. We run experiments on subsets of our data based on the problem: the number of waypoints per cluster for MATSP, and the number of tasks for Scheduling.

4.1 Multi-Agent Traveling Salesman Problem

We find that the training performance is roughly equal between both approaches as shown in table 4.1, while the downstream optimal variable assignment accuracies improve upon the baseline using the contrastive learning approach as shown in table 4.2. The downstream optimal variable assignment accuracies improve by up to 5%, such as in the 4 and 5 waypoints per cluster cases.

# Waypoints Per Cluster	**Baseline Training Loss**	**Baseline Validation Loss**	**Contrastive Training Loss**	**Contrastive Validation Loss**
3	0.2334	0.2282	0.2283	0.2278
4	0.2199	0.2193	0.2259	0.2265
5	0.2344	0.2182	0.2378	0.2354
6	0.2697	0.2564	0.2068	0.2106
7	0.1939	0.1939	0.1946	0.1955

Table 4.1: Comparison of MATSP training performance based on the number of waypoints per cluster.

# Waypoints Per Cluster	Baseline Accuracy	Contrastive Accuracy
3	89.78%	91.24%
4	87.2%	92.42%
5	88.04%	93.36%
6	89.43%	92.38%
7	91.68%	94.14%

Table 4.2: Comparison of MATSP downstream optimal variable prediction accuracy based on the number of waypoints per cluster.

4.2 Scheduling Problem

We find that the training performance is roughly equal between both approaches as shown in table 4.3, while the downstream optimal variable assignment accuracies improve upon the baseline in most cases using the contrastive learning approach, aside from the 12 tasks experiment as shown in table 4.4. The downstream optimal variable assignment accuracies improve by up to 6.25%, such as in the 5 tasks case. Improving the accuracy for 11 tasks is significant to note, as state-of-the-art approaches do not trivially find optimal solutions for this setup.

Number of Tasks	Baseline Training Loss	Baseline Validation Loss	Contrastive Training Loss	Contrastive Validation Loss
5	0.5425	0.4079	0.5475	0.3892
6	0.5922	0.5091	0.5818	0.5073
7	0.597	0.5413	0.5937	0.5407
8	0.6768	0.6232	0.6783	0.6259
9	0.7052	0.6417	0.7101	0.649
10	0.7835	0.7096	0.7761	0.7138
11	0.789	0.7256	0.8113	0.7347
12	0.781	0.7826	0.7838	0.7882

Table 4.3: Comparison of Scheduling training performance based on the number of tasks.

Number of Tasks	**Baseline Accuracy**	**Contrastive Accuracy**
5	68.75%	75%
6	68.42%	73.68%
7	63.64%	68.18%
8	64%	68%
9	60.71%	64.29%
10	61.29%	64.52%
11	58.82%	61.76%
12	62.16%	62.16%

Table 4.4: Comparison of Scheduling downstream optimal variable prediction accuracy based on the number of tasks.

Chapter 5

Conclusion and Future Work

Our work presents a contrastive learning-based approach for combinatorial optimization, and in the context of downstream optimal variable assignment, demonstrates an over 6% accuracy increase for a Scheduling problem benchmark, and an over 5% accuracy increase for a Multi-Agent Traveling Salesman problem benchmark, when compared to a baseline that does not utilize this approach. We demonstrate that contrastive learning is able to learn variable embeddings on a per-problem basis, hence improving our model's accuracy to predict optimal variable assignments for OMT problems. We contribute benchmarks of problem families including Scheduling and Multi-Agent Traveling Salesman for evaluating learning-based OMT solvers.

Future work may include modifying the contrastive learning architecture to allow for multiple transformations to be compared to each other, as opposed to only comparing the embeddings of a problem and its respective transformation. Such an approach may reduce the contrastive loss by allowing the model to learn more complex relationships between lower-dimensional embeddings of equivalent OMT problem transformations, and hence allow better embeddings for each variable to be learned. Additionally, future work may include exploring more complex transformations that maintain satisfiability and optimality.

Furthermore, different neural network architectures, such as Graph Attention Networks (GATs) [22], can be explored with the goal of producing better variable embeddings. In GATs, each node learns to assign attention weights to its neighbors based on their representations, and then combines these representations by weighting them with attention scores. This process is repeated over multiple layers, allowing the network to learn increasingly complex patterns in the graph. GATs have been shown to achieve state-of-the-art performance on various benchmarks, and may further improve the performance of the benchmarks presented in this paper.

Bibliography

[1] Clark Barrett, Roberto Sebastiani, Sanjit A. Seshia, and Cesare Tinelli. "Satisfiability Modulo Theories". In: *Handbook of Satisfiability*. Ed. by Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh. Second. IOS Press, 2021. Chap. 33, pp. 1267–1329.

[2] Azalia Mirhoseini et al. "Chip placement with deep reinforcement learning". In: *arXiv preprint arXiv:2004.10746* (2020).

[3] Má rton Búr, Kristóf Marussy, Brett H. Meyer, and Dániel Varró. "Worst-case Execution Time Calculation for Query-based Monitors by Witness Generation". In: *ACM Transactions on Embedded Computing Systems* 20.6 (Oct. 2021), pp. 1–36. DOI: `10.1145/3471904`. URL: `https://doi.org/10.1145%2F3471904`.

[4] Omar Cheikhrouhou and Ines Khoufi. "A comprehensive survey on the Multiple Traveling Salesman Problem: Applications, approaches and taxonomy". In: *Computer Science Review* 40 (May 2021), p. 100369. DOI: `10.1016/j.cosrev.2021.100369`. URL: `https://doi.org/10.1016%2Fj.cosrev.2021.100369`.

[5] Ignacio E. Grossmann et al. "Recent advances in mathematical programming techniques for the optimization of process systems under uncertainty". In: *Computers & Chemical Engineering* 91 (2016). 12th International Symposium on Process Systems Engineering & 25th European Symposium of Computer Aided Process Engineering (PSE-2015/ESCAPE-25), 31 May - 4 June 2015, Copenhagen, Denmark, pp. 3–14. ISSN: 0098-1354. DOI: `https://doi.org/10.1016/j.compchemeng.2016.03.002`. URL: `https://www.sciencedirect.com/science/article/pii/S0098135416300540`.

[6] Roberto Sebastiani and Patrick Trentin. "OptiMathSAT: A Tool for Optimization Modulo Theories." In: *Proc. International Conference on Computer-Aided Verification, CAV 2015*. Vol. 9206. LNCS. Springer, 2015.

[7] David Déharbe, Pascal Fontaine, Stephan Merz, and Bruno Woltzenlogel Paleo. "Exploiting Symmetry in SMT Problems". In: *Automated Deduction – CADE-23*. Ed. by Nikolaj Bjørner and Viorica Sofronie-Stokkermans. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 222–236. ISBN: 978-3-642-22438-6.

[8] Justin Wong et al. *Ashera; Neural Guided Optimization Modulo Theory*. Tech. rep. UCB/EECS-2023-103. EECS Department, University of California, Berkeley, May 2023. URL: `http://www2.eecs.berkeley.edu/Pubs/TechRpts/2023/EECS-2023-103.html`.

[9] Martin Bromberger, Thomas Sturm, and Christoph Weidenbach. *Linear Integer Arithmetic Revisited*. 2020. arXiv: `1503.02948 [cs.LO]`.

[10] Shiwen Wu et al. *Graph Neural Networks in Recommender Systems: A Survey*. 2022. arXiv: `2011.02260 [cs.IR]`.

[11] Kehang Han, Balaji Lakshminarayanan, and Jeremiah Liu. *Reliable Graph Neural Networks for Drug Discovery Under Distributional Shift*. 2021. arXiv: `2111.12951 [cs.LG]`.

[12] Alex Davies and Nirav Ajmeri. *Realistic Synthetic Social Networks with Graph Neural Networks*. 2022. arXiv: `2212.07843 [cs.SI]`.

[13] Thomas N. Kipf and Max Welling. *Semi-Supervised Classification with Graph Convolutional Networks*. 2017. arXiv: `1609.02907 [cs.LG]`.

[14] Wei-Lin Chiang et al. "Cluster-GCN". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery &amp; Data Mining*. ACM, 2019. DOI: `10.1145/3292500.3330925`. URL: `https://doi.org/10.1145%5C%2F3292500.3330925`.

[15] Quentin Cappart et al. *Combinatorial optimization and reasoning with graph neural networks*. 2022. arXiv: `2102.09544 [cs.LG]`.

[16] Azalia Mirhoseini et al. "A graph placement methodology for fast chip design". In: *Nature* 594 (2021), pp. 207–212.

[17] Haonan Duan et al. *Augment with Care: Contrastive Learning for Combinatorial Problems*. 2022. arXiv: `2202.08396 [cs.LG]`.

[18] Raia Hadsell, Sumit Chopra, and Yann Lecun. "Dimensionality Reduction by Learning an Invariant Mapping". In: Feb. 2006, pp. 1735–1742. ISBN: 0-7695-2597-0. DOI: `10.1109/CVPR.2006.100`.

[19] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: `2002.05709 [cs.LG]`.

[20] Maxime Gasse et al. "Exact combinatorial optimization with graph convolutional neural networks". In: *arXiv preprint arXiv:1906.01629* (2019).

[21] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. *Representation Learning with Contrastive Predictive Coding*. 2019. arXiv: `1807.03748 [cs.LG]`.

[22] Petar Veličković et al. *Graph Attention Networks*. 2018. arXiv: `1710.10903 [stat.ML]`.