Extending Test-Time Augmentation with Metamorphic Relations for Combinatorial Problems

Siwei Wei [1 2] **Xudong Zhang** [1 2] **Zhiyang Zhou** [1] **Yan Cai** [1]

Abstract

The application of machine learning methods to solve combinatorial problems has garnered considerable research interest. In this paper, we propose MAGG (**M**etamorphic **Agg**regation), a method to augment machine learning models for combinatorial problems at inference time using metamorphic relations. MAGG models metamorphic relations using directed graphs, which are then fed to a Graph Neural Network (GNN) model to improve the aggregation of predictions across transformed input instances. By incorporating metamorphic relations, MAGG essentially extends standard Test-Time Augmentation (TTA), eliminating the necessity of label-preserving transformations and expanding its applicability to a broader range of supervised learning tasks for combinatorial problems. We evaluate the proposed MAGG method on three mainstream machine learning tasks for combinatorial problems, namely Boolean Satisfiability Prediction (SAT), Decision Traveling Salesman Problem Satisfiability Prediction (Decision TSP), and Graph Edit Distance Estimation (GED). The evaluation result shows significant improvements over base models in all three tasks, corroborating the effectiveness and versatility of the proposed method.

1. Introduction

Solving combinatorial problems efficiently is a pivotal and challenging task in the field of computer science (Korte et al., 2011; Papadimitriou & Steiglitz, 1998). Recently, many works have been proposed to solve combinatorial problems using machine learning techniques (Bengio et al., 2021; Mazyavkina et al., 2021; Cappart et al., 2023). However, the intrinsic characteristics of combinatorial problems, such as their discrete nature and NP-hardness, still pose significant challenges to machine learning techniques.

Test-time augmentation (TTA) (Moshkov et al., 2020; Lyzhov et al., 2020; Shanmugam et al., 2021; Kim et al., 2020) is a technique aiming at enhancing a pretrained model at inference time. The workflow of standard TTA (defined in (Shanmugam et al., 2021)) is shown in Figure 1 (b). It turns target instance into different versions using label-preserving transformations such as rotation (see Figure 4). Model predictions on these versions of target instance are then averaged to give a better prediction. TTA has proven to provide more accurate and robust prediction than one-shot prediction (see Figure 1 (a)) in computer vision tasks.

However, standard TTA has many shortcomings, prohibiting it from applying to augment machine learning models for combinatorial problems. First, it is not easy to find label-preserving transformations for combinatorial problems, which are required by both standard TTA and conventional data augmentation methods to give unbiased predictions (Shorten & Khoshgoftaar, 2019; Duan et al., 2022). Second, there exists numerous semantic relations between instances of combinatorial problems (since combinatorial problems are formally definable), but standard TTA fails to leverage them. Third, standard TTA uses a fixed average strategy to aggregate model predictions, making it infeasible to adapt to different models.

Enlightened by works on metamorphic testing (Segura et al., 2016; Chen et al., 2018; Segura et al., 2018), we propose to extend TTA technique with metamorphic relations to amend its shortcomings. Metamorphic testing is a software testing technique that leverages the relationship among different outputs of the target system (i.e. metamorphic relation) for test. It is widely used to test deep learning based utilities (Zhang et al., 2021b; Wang & Su, 2019; Naidu et al., 2021; Dwarakanath et al., 2018). Metamorphic relations are necessary properties of the target system (see Definition 3.1). An example is given in Figure 3. Suppose one wants to test an implementation of SAT whose aim is to predict whether a boolean formula is satisfiable. Then for the two instances q and q', $SAT(q') \rightarrow SAT(q)$. If the

[1]Key Laboratory of System Software (Chinese Academy of Sciences) and State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, and University of Chinese Academy of Sciences, Beijing, China [2]Co-first authors. Correspondence to: Yan Cai <yancai@ios.ac.cn>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

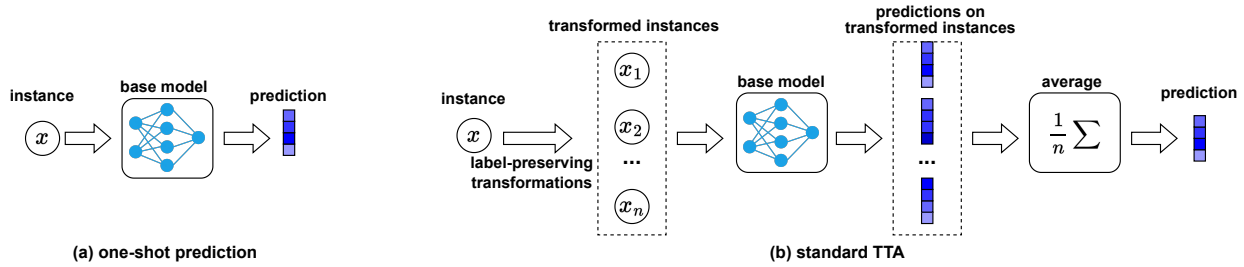

Figure 1. Workflow of (a) one-shot prediction and (b) standard TTA.

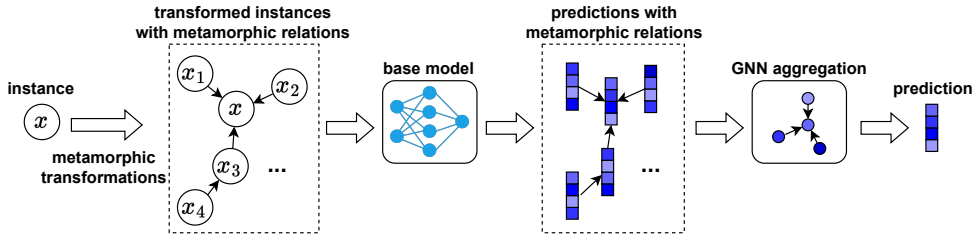

Figure 2. Workflow of MAGG.

implementation violates this relation, it can be concluded that it is faulty. Indeed, label-preserving transformations used by standard TTA can be viewed as a special kind of metamorphic relations (see Figure 4).

We believe metamorphic relations are useful for test-time augmentation of combinatorial problems, for the following reasons. First, since metamorphic relations can be used to detect errors, they should encompass information that enables models to make better predictions (see Appendix D.1). Second, metamorphic relations are prevalent in combinatorial problems, since these problems are formally definable. Third, it is not easy for a machine learning model to learn these relations directly from data (see Appendix D.2).

In this paper, we propose MAGG, a method to augment machine learning models at inference time by leveraging metamorphic relations. The workflow of MAGG is shown in Figure 2 (c). Given a problem instance x, MAGG first transforms it into several mutants using metamorphic transformations. Note in this step the metamorphic transformations do not have to be label-preserving, and they can be applied iteratively. Then, metamorphic relations among transformed instances are built, forming an Ego Metamorphic Graph (EMG) (see Definition 3.6) of x. After that, the base model (the pretrained model to augment) is executed to give predictions for transformed instances. Finally, the EMG with base model predictions as node features is fed to a Graph Neural Network to give final prediction.

The proposed MAGG essentially extends standard TTA technique in the following aspects. First, instead of requiring label-preserving transformations, MAGG is able to utilize a wide range of metamorphic relations, making it applicable to the augmentation of machine learning models for combinatorial problems. Second, by using a GNN to aggregate EMG, MAGG can utilize the information that metamorphic relations encompass to make better predictions. Third, the use of a trainable GNN instead of a fixed average strategy helps MAGG adapt to models with different characteristics. In addition, the ubiquity of metamorphic relations in combinatorial problems makes MAGG universally applicable. To the best of our knowledge, MAGG is the first TTA technique that can be applied for learning based combinatorial problem solving.

MAGG is evaluated on three mainstream machine learning tasks for combinatorial problems, namely Boolean Satisfiability Prediction (SAT), Decision Traveling Salesman Problem Satisfiability Prediction (Decision TSP), and Graph Edit Distance Estimation (GED). NeuroSAT (Selsam et al., 2019), TSP-GNN (Prates et al., 2019), and GREED (Ranjan et al., 2022) are used as base models for these tasks, respectively. MAGG achieves considerable improvements on all tasks. Specifically, it achieves maximal 15%, 4%, and 42% improvements relative to base models on the three tasks, respectively. It is surprising that MAGG, as a lightweight TTA method aiming at augmenting existing models, can achieve such a magnitude of improvement. The evaluation result shows (1) using metamorphic relations to augment pretrained models at inference time is helpful, and (2) MAGG does well in utilizing information provided by metamorphic relations.

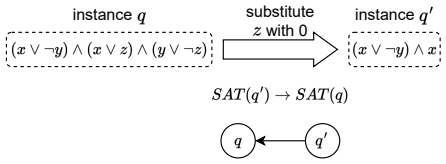

Figure 3. Metamorphic relation in SAT.

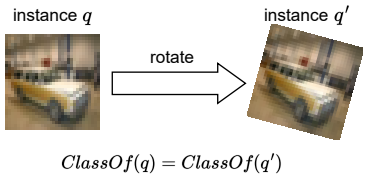

Figure 4. Label-preserving transformations can be viewed as a special kind of metamorphic relations.

In summary, the contribution of this paper is three-fold:

- We present the first attempt to apply TTA to machine learning models for combinatorial problems.
- We propose to extend TTA with metamorphic relations, and we design an effective method that models metamorphic relations with directed graphs and utilizes GNN for better aggregation.
- We conduct evaluations on three tasks, and the results show significant improvements (maximally 42%), corroborating the effectiveness of the proposed method.

2. Background and Related Work

In this section, three research areas closely related to our work, namely metamorphic testing, test-time augmentation, and machine learning for combinatorial problems, are introduced.

2.1. Metamorphic Testing

Metamorphic testing (Segura et al., 2016; Chen et al., 2018; Segura et al., 2018) is a software testing technique which aims to alleviate the test oracle problem. The field of metamorphic testing has seen remarkable growth and increased attention in recent years within the software testing and quality assurance communities. Metamorphic testing has also been applied to test deep learning models, and has proven to be fruitful and valuable (Zhang et al., 2021b; Wang & Su, 2019; Naidu et al., 2021; Dwarakanath et al., 2018).

The key idea of metamorphic testing is that instead of fully formalizing a system's input-output behavior, it is easier to check the relations of multiple outputs. Such relations are called metamorphic relations, which are necessary properties of target function. A prototypical example is the test of a sin function. Without knowing the exact value of $sin(x)$, one can check whether the metamorphic relation $sin(x) = sin(\pi - x)$ holds. If this relation is violated, the implementation of sin must be faulty. Additional examples of metamorphic relations are illustrated in Figures 3, 4, 6, and 7.

Inspired by the works on metamorphic testing, we propose to leverage metamorphic relations for augmenting machine learning models at inference time. This is because (1) as metamorphic relations can reveal deficiencies of machine learning models, they should contain information that can be used to remedy the deficiencies and (2) general metamorphic relations are more prevalent than label-preserving transformations, especially in the field of combinatorial problems.

2.2. Test-Time Augmentation

Test-time augmentation (TTA) (Moshkov et al., 2020; Lyzhov et al., 2020; Shanmugam et al., 2021; Kim et al., 2020; Kimura, 2021; Wang et al., 2019; Pérez et al., 2021; Chun et al., 2022; Mocerino et al., 2021) is a common practice in computer vision tasks, such as image classification. It can be viewed as a special type of data augmentation (Shorten & Khoshgoftaar, 2019). The distinctive characteristic is that TTA is performed at inference time, and it only requires a pretrained model, without the need of further training. This characteristic makes it a popular and easy-to-use technique. TTA has garnered substantial research attention in recent years, demonstrating its capability to yield more accurate and robust predictions.

Standard TTA (defined in (Shanmugam et al., 2021)) applies label-preserving transformation to target instance, yielding various versions. Then the predictions of base model on these versions are averaged to give a final prediction. Recently, many enhancements of standard TTA are put forwarded. For example, (Lyzhov et al., 2020) proposes to select useful transformations greedily, (Shanmugam et al., 2021) proposes to use trainable weighted sum instead of simple average, and (Kim et al., 2020) proposes to select suitable transformation for each instance. However, these works still use label-preserving transformations, and the enhancements are confined in computer vision tasks. Thus the potential of the idea of test-time augmentation has not been fully exploited.

Our work goes beyond standard TTA by employing general metamorphic relations instead of relying on label-preserving transformations. Metamorphic relations contain valuable information for augmentation, and are more prevalent than label-preserving transformations. Additionally, we use a trainable GNN model instead of a simple average to aggre-

gate predictions. These features make the proposed method more flexible and broadly applicable.

2.3. Machine Learning for Combinatorial Problems

Combinatorial problem solving (Korte et al., 2011; Papadimitriou & Steiglitz, 1998) is a pivotal area in computer science, and have garnered considerable research interest. Recently, there is a surge of interest to use machine learning models directly as solvers for combinatorial problems (Bengio et al., 2021; Mazyavkina et al., 2021; Cappart et al., 2023). Some works discuss the use of specific metamorphic relations in graph data and their application to SAT problems (Xu et al., 2018; Sun et al., 2022). Even though these methods have achieved considerable accomplishments, many combinatorial problems are still challenging to machine learning methods because of their discrete and NP-hard nature.

In this paper, we present the first attempt to apply TTA for combinatorial problems. We believe using metamorphic relations at inference time for augmentation is the most suitable for combinatorial problems, according to following reasons. First, it is not easy to find effective label-preserving transformations for many combinatorial problems (see Appendix D.3). Second, there exists a considerable amount of effective metamorphic relations in combinatorial problems, since combinatorial problems are formally definable. Third, non-label-preserving metamorphic relations are challenging to apply for data augmentation or unsupervised learning, as discussed in (Shorten & Khoshgoftaar, 2019; Duan et al., 2022).

3. Method

The proposed MAGG method extends test-time augmentation with metamorphic relations. The key components of MAGG are introduced in detail in the following subsections.

3.1. Representing Metamorphic Relations

We first give formal definitions of metamorphic testing related concepts, which are from (Chen et al., 2018).

Definition 3.1. (Metamorphic Relation). Let $f : X \to Y$ be a target function. A metamorphic relation is a necessary property of f over multiple inputs $\langle x_1, x_2, \cdots x_n\rangle$ and corresponding outputs $\langle f(x_1), f(x_2) \cdots f(x_n)\rangle$, which can be expressed as $R \subseteq X^n \times Y^n$.

Definition 3.2. (Source Input, Follow-up Input and Metamorphic Transformation). Consider a metamorphic relation R over $\langle x_1, x_2, \cdots x_n\rangle$ and $\langle f(x_1), f(x_2) \cdots f(x_n)\rangle$. Suppose $\langle x_{k+1}, x_{k+2}, \cdots x_n\rangle$ is constructed by applying a transformation T to $\langle x_1, x_2, \cdots x_k, f(x_1), f(x_2) \cdots f(x_k)\rangle$. We refer to $x_i, i = 1 \cdots k$ as source input, $x_j, j = k+1 \cdots n$ as follow-up input, and T as metamorphic transformation.

Definition 3.3. (Metamorphic Testing). Let m be an implementation of the target function f. Suppose we have a metamorphic relation R and the corresponding metamorphic transformation T. Metamorphic testing based on R involves the following steps:

(1) Given source input $\langle x_1, x_2, \cdots x_k\rangle$, apply m to them and obtain corresponding $\langle m(x_1), m(x_2), \cdots m(x_k)\rangle$.

(2) Apply the metamorphic transformation T to $\langle x_1, x_2, \cdots x_k, m(x_1), m(x_2) \cdots m(x_k)\rangle$, and obtain follow-up input $\langle x_{k+1}, x_{k+2}, \cdots x_n\rangle$.

(3) Apply m to follow-up input $\langle x_{k+1}, x_{k+2}, \cdots x_n\rangle$, and obtain $m(x_{k+1}), m(x_{k+2}), \cdots m(x_n)$.

(4) Check whether $\langle x_1, x_2 \cdots x_n, m(x_1), m(x_2) \cdots m(x_n)\rangle$ satisfies R. If not, then m is faulty.

For example, consider the metamorphic relation in Figure 3. The target function is SAT. The source input q is $(x \vee \neg y) \wedge (x \vee z) \wedge (y \vee \neg z)$, follow-up input q' is $(x \vee \neg y) \wedge x$, and the metamorphic transformation that transforms q to q' is substituting z with 0. The corresponding metamorphic relation is $SAT(q') \to SAT(q)$. If an implementation m gives $m(q) = 0$ and $m(q') = 1$, it can be concluded that m is faulty.

In the context of supervised learning, the target function f can be viewed as the ground truth, and the implementation m can be viewed as the machine learning model which is used to fit f.

To make things simpler, MAGG only considers binary metamorphic relations and single-input metamorphic transformations.

Definition 3.4. (Binary Metamorphic Relation). A binary metamorphic relation R is a metamorphic relation concerning only two test instances. That is, R is over $\langle x_1, x_2, f(x_1), f(x_2)\rangle$, expressed as $R \subseteq X^2 \times Y^2$.

Definition 3.5. (Single-input Metamorphic Transformation). A single-input metamorphic transformation T is a metamorphic transformation that transforms one source test instance x_1 to one follow-up test instance x_2. That is, $T : X \to X$.

The metamorphic relations and corresponding transformations shown in Figure 3 and 4 are both binary metamorphic relations (and single-input transformations).

The advantage of using binary metamorphic relations is that they can be conveniently represented in a graph. We represent a binary metamorphic relation as a directed edge, with one endpoint being the source instance and the other being the follow-up instance. Figure 3 gives such an example.

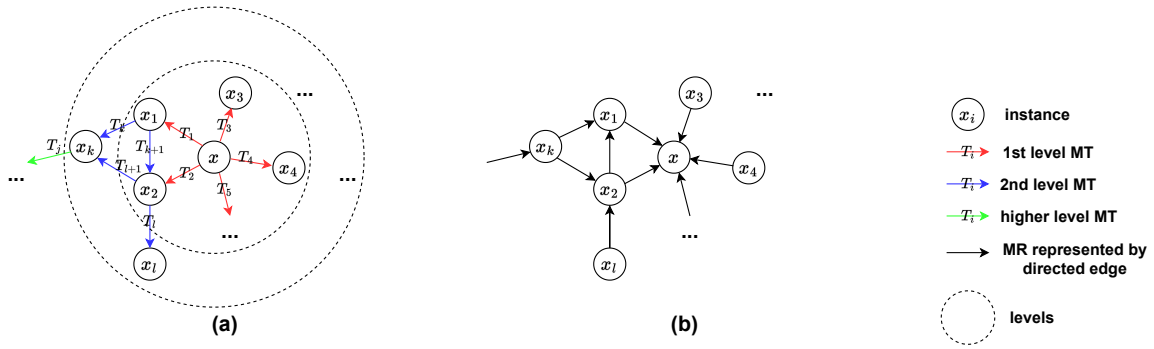

Figure 5. Construction of Ego Metamorphic Graph.

3.2. Constructing Ego Metamorphic Structure

In this section, we investigate the method for extracting information within metamorphic relations for a given instance x. We propose to construct a graph for x, defined as the Ego Metamorphic Graph of x.

Definition 3.6. (Ego Metamorphic Graph). An Ego Metamorphic Graph (EMG) is defined as a triplet $G = \langle V, E, x \rangle$, where V is the set of nodes representing instances, E is the set of directed edges representing binary metamorphic relations, $x \in V$ is the central node representing the instance of concern. It is a requisite that all nodes $y \in V$ are connected to x.

Figure 5 (b) gives an example of Ego Metamorphic Graph, where x is the central node.

For an instance x, an Ego Metamorphic Graph of x is constructed using a given set of single-input metamorphic transformations $\mathbb{T} = \{T_i\}$. The Ego Metamorphic Graph is constructed incrementally. Initially, $G = \langle \{x\}, \emptyset, x \rangle$. Each time, we sample a node u from node set V, and sample a single-input metamorphic transformation T_i from $\mathbb{T}$, then apply T_i to u. The newly generated instance and binary metamorphic relation are added to G. We refer to single-input metamorphic transformations directly applied to x as first-level metamorphic transformations, those applied to instances that are transformed directly from x as second-level metamorphic transformations, and so forth. Figure 5 gives such an example.

3.3. Aggregating Predictions

Given an Ego Metamorphic Graph $G = \langle V, E, x \rangle$, we believe that the predictions of the base model on V and the graph structure are helpful for improving the prediction of x. In this paper, we propose to use a trainable aggregation model to learn how to leverage them from data. The advantages of using a trainable aggregation model are as follows: (1) Using a trainable aggregation model frees people from writing hard-coded aggregation rules for various metamorphic relations and tasks, especially when the appropriate aggregation rule is not evident. (2) Using a trainable aggregation model makes it feasible to adapt to the features of different base models. A base model can be aggressive or conservative, and can predict more accurately on some instances than on others. A trainable aggregation model can provide targeted aggregation strategies for it.

We choose to use Graph Neural Network (GNN) as the aggregation model. The benefits of using GNN are as follows: (1) Since the ego metamorphic structure of a target instance is represented as a graph, it is natural to use a GNN to leverage it. (2) GNN can adapt to varying Ego Metamorphic Graphs, making it feasible to train and test on instances with different ego metamorphic structures.

Specifically, in this paper a simplified Message Passing Neural Network (MPNN) model (Gilmer et al., 2017) is adopted. We design this model to be as simple as possible to demonstrate that the effectiveness of the proposed MAGG method comes mostly from the incorporation of metamorphic relations rather than aggregation model. The message passing phase runs for T times, where T is the maximum level of metamorphic transformations (see Section 3.2). Node states $h_u^t \in \mathbb{R}^{d_t}$ are updated using messages $m_u^t \in \mathbb{R}^{d_t}$ which are aggregated from their neighbors. The update rules are

$$m_u^{t+1} = \frac{1}{|N(u)|} \sum_{v \in N(u)} M_t(h_u^t)$$

$$h_u^{t+1} = U_t(h_u^t, m_u^{t+1})$$

where $N(u)$ is the set of neighbors of u, that is, the nodes with a directed edge to u in a directed graph. The initial node state (or node feature) $h_u^0 \in \mathbb{R}^{\mathbb{C}}$ is the prediction of base model on instance u ($\mathbb{C}$ is the number of classes). The state of central node x at time T, i.e. $h_x^T \in \mathbb{R}^{\mathbb{C}}$, is used as final prediction.

3.4. Training the Aggregation Model

A portion of the training set is used to train the aggregation model, as we observed that the base models we use do not seem to overfit. For each instance-label pair in the selected portion of training set, an Ego Metamorphic Graph is generated for this instance, with base model predictions as node features. In this way, a graph classification task is constructed upon which the aggregation model is trained.

4. Evaluation

In this Section, we evaluate the proposed MAGG method on machine learning tasks for combinatorial problems, assessing its ability to enhance base models' performance at inference time. Three tasks are selected, namely Boolean Satisfiability Prediction (SAT), Decision Traveling Salesman Problem Satisfiability Prediction (Decision TSP), and Graph Edit Distance Estimation (GED). The details of each task, including base models, datasets, evaluation metrics, and maximal improvements, are summarized in Table 1. All experiments are conducted on a Ubuntu 22.04.2 LTS machine with Intel Xeon Gold 6338 processor and NVIDIA A800 80GB PCIe GPU. The code of the experiments is available at `https://github.com/MetamorphicAgg/MAgg`.

4.1. Boolean Satifiability Prediction

4.1.1. Task Description and Base Model

The Boolean Satisfiability Prediction task (SAT) (Gu et al., 1996) aims to predict the satisfiability of a boolean formula presented in conjunctive normal form (CNF). Recently, many machine learning based methods have been proposed to directly solve SAT (Selsam et al., 2019; Ozolins et al., 2022; Bünz & Lamm, 2017) or provide assistance in solving it (Zhang et al., 2021a; Zhang & Zhang, 2021).

NeuroSAT (Selsam et al., 2019) is used as base model. It is a seminal model in learning-based SAT prediction, trained with a single-bit of supervision. The authors of NeuroSAT sample SAT instances of n variables and corresponding satisfiability labels from a distribution $SR(n)$. NeuroSAT is trained using instances sampled from $SR(U(10, 40))$ and tested on $SR(40), SR(80) \cdots$ to show its generalization ability. More details are given in Appendix A.3.

We train a NeuroSAT model using the same setup as in (Selsam et al., 2019), and get similar results as reported (see Appendix A.3).

4.1.2. Evaluation Setup

Metamorphic Relation. A family of metamorphic transformations $\mathbb{T}$ is used, where $\mathbb{T} = \{T_{i,v} \,|i = 1, 2, \cdots n, v = 0, 1\}$. $T_{i,v}$ replaces the i-th variable in an instance with boolean value v, yielding a transformed instance. An example of $T_{i,v}$ and the corresponding metamorphic relation is shown in Figure 3.

Two settings, namely *MAgg(n)* and *MAgg(n,m)*, are used to generate Ego Metamorphic Graphs. In the *MAgg(n)* setting, only first-level metamorphic transformations (refer to Figure 5) are used, and n metamorphic transformations from $\mathbb{T}$ are sampled for each instance. In the *MAgg(n, m)* setting, first and second-level metamorphic transformations are used. We sample n first-level metamorphic transformations for each instance, and sample m second-level metamorphic transformations for each first level transformation.

Aggregation Model. A simplified MPNN model (refer to Section 3.3) is used as aggregation model. The message function M_t is set to the identity function. The state update function U_t is set to $U_t(h_u^t, m_u^{t+1}) = \sigma_t(W_t \cdot (h_u^t \,\|\, m_u^{t+1}))$, i.e. a fully connected layer operating on the concatenation of node state and message. In the *MAgg(n)* setting, $\sigma_0 = id$, and in the *MAgg(n, m)* setting, $\sigma_0 = softmax, \sigma_1 = id$. The hidden dimensions are set to $d_t = 2$ and the same weight W_t is used for each layer. Note for *MAgg(n)*, the aggregation model is essentially linear.

Training the Aggregation Model. For each $n \in \{40, 80, 120\}$ and settings *MAgg(10)*, *MAgg(10,10)*, 5000 instances are sampled from the training set (which follows $SR(n)$), and an Ego Metamorphic Graph is constructed for each instance. The aggregation model is trained on the corresponding set for 100 epochs with a 0.001 learning rate.

Testing. The performance of the proposed method is evaluated for each $n \in \{40, 80, 120\}$. The test set contains 5000 instances sampled from $SR(n)$. Accuracy is used as evaluation metric. The process of building Ego Metamorphic Graphs and training the aggregation model is repeated for 10 times, and the mean accuracy is reported.

4.1.3. Classification Result

In Table 2, we compare the classification accuracy of base model (the NeuroSAT column) and base model augmented with MAGG in two settings (the MAgg(10) and MAgg(10,10) columns). For every n, NeuroSAT augmented with MAGG considerably outperforms NeuroSAT. Maximally, MAGG achieves 0.13 accuracy improvement.

This result shows that leveraging metamorphic relations at test time can significantly improve the performance of base model . Also, the metamorphic transformations used are not label-preserving, thus can not be leveraged by standard TTA. The significant accuracy improvement in the *MAgg(10,10)* setting compared to *MAgg(10)* shows the helpfulness of leveraging indirect metamorphic relations of the central instance. This underscores the superiority of constructing a multi-hop Ego Metamorphic Graph over solely considering one-step transformed instances.

Table 1. Summary of the selected tasks and the results.

Task	**Base Model**	**Datasets**	**Metric**	**Maximal Improvements of** MAGG
SAT	NeuroSAT	SR(40), SR(80), SR(120)	accuracy	15% relative to base model
Decision TSP	TSP-GNN	TSP(0.01), TSP(0.02)	accuracy	4% relative to base model
GED	GREED	AIDS, IMDB, LINUX	RMSE	42% relative to base model

Table 2. Classification result for SAT in accuracy.

Dataset	NeuroSAT	MAgg(10)	MAgg(10,10)
SR(40)	0.8444	0.9548	**0.9757**
SR(80)	0.7268	0.7936	**0.8533**
SR(120)	0.6270	0.6412	**0.6643**

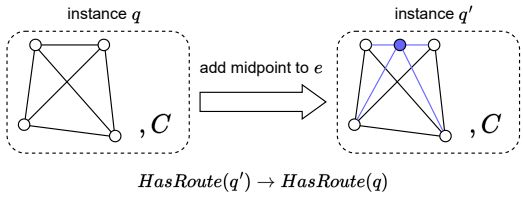

Figure 6. Metamorphic relation in decision TSP.

It is surprising that the aggregation model in the *MAgg(10)* setting, which is essentially linear, can achieve such a magnitude of accuracy improvement. We analyze the effect of the *MAgg(10)* aggregation model in Appendix A.7, and find that using base model predictions on transformed instances as features makes the satisfiable and unsatisfiable instances linearly separable, which underscores the value of metamorphic relations.

4.2. Decision TSP Satifiability Prediction

4.2.1. Task Description and Base Model

An instance of Decision Traveling Salesman Problem (Decision TSP) (Zambito, 2006) is a pair $\langle G, C \rangle$, where G is a weighted complete graph and C is target cost. Given an instance, the Decision TSP Satifiability Prediction task aims at predicting whether there exists a Hamiltonian cycle whose cost is less than or equal to C. Many machine learning-based methods have been proposed for solving TSP (Prates et al., 2019; Vinyals et al., 2015; Joshi et al., 2019).

For this task, TSP-GNN (Prates et al., 2019) is used as base model. It is a GNN based model trained to predict the satisfiability of Decision TSP instances. TSP-GNN is trained on 2d Eucildean Decision TSP instances. The authors of TSP-GNN sample 2d Eucildean Decision TSP instances and corresponding satisfiability labels from a distribution, which we denote as $TSP(dev)$, to train and test TSP-GNN. The dev argument controls the difficulty of Decision TSP instances. The lower dev is, the harder the Decision TSP instances are. The TSP-GNN model is trained on $TSP(0.01)$, $TSP(0.02)$, $TSP(0.05)$ and $TSP(0.10)$, respectively, and tested using instances sampled from the corresponding distribution. More details are given in Appendix B.3.

We train a TSP-GNN model using the same setup as in (Prates et al., 2019) for $TSP(0.01)$ and $TSP(0.02)$, and get similar results as reported (see Appendix B.3).

4.2.2. Evaluation Setup

Metamorphic Relation. As the TSP-GNN model is trained with 2d Eucildean Decision TSP instances, the chosen metamorphic transformations should preserve the 2d Eucildean property. To this end, a family of metamorphic transformations $\mathbb{T}$ is used, where $\mathbb{T} = \{T_e \mid e \; is \; the \; 50\% \; edges \; with \; the \; least \; weights\}$. T_e adds a node which is at the midpoint of e and adds corresponding edges, yielding a transformed instance. We select e from the 50% edges with least weights to ensure a close relationship in satisfiability between the two Decision TSP instances. Suppose T_e transforms 2d Eucildean Decision TSP instance q to q', because of the triangular inequality in Euclidean space, we have the metamorphic relation $HasRoute(q') \rightarrow HasRoute(q)$, as shown in Figure 6.

The *MAgg(n)* setting (refer to Section 4.1.2) is used to generate Ego Metamorphic Graphs.

Aggregation Model. The same aggregation model as in Section 4.1.2 is adopted.

Training the Aggregation Model. For each $dev \in \{0.01, 0.02\}$, 2048 instances are sampled from the training set (which follows $TSP(dev)$), and an Ego Metamorphic Graph is constructed for each instance. We train the aggregation model on the corresponding set for 100 epochs, with a 0.0001 learning rate.

Testing. The performance of the proposed method is evaluated for $dev \in \{0.01, 0.02\}$. The test set contains 2048 instances sampled from $TSP(dev)$. Accuracy is used as evaluation metric. The process of building Ego Metamorphic Graphs and training the aggregation model is repeated for 10 times, and the mean accuracy is reported.

Table 3. Classification result for Decision TSP in accuracy.

Dataset	TSP-GNN	MAgg(10)
TSP(0.01)	0.6562	**0.6812**
TSP(0.02)	0.8101	**0.8321**

4.2.3. Classification Result

In Table 3, we compare the classification accuracy of base model (the TSP-GNN column) and base model augmented with MAGG (the MAgg(10) column). The result shows a notable accuracy improvement after augmentation with MAGG on both datasets, corroborating that utilizing metamorphic relations at inference time is effective.

4.3. Graph Edit Distance Estimation

4.3.1. Task Description and Base Model

Graph Edit Distance (GED) (Gao et al., 2010) is a measure of graph similarities, defined as the minimum number of steps needed to transform one graph into another using a predefined set of edit operations. An instance of GED estimation comprises two graphs, denoted as $\langle G_1, G_2 \rangle$, and the model is required to estimate the GED of the two graphs. Many machine learning based methods have been proposed for effective GED estimation (Ranjan et al., 2022; Bai et al., 2019; 2020; Li et al., 2019).

For this task, GREED (Ranjan et al., 2022) is used as base model. It is an efficient Siamese Graph Neural Network architecture for graph similarity measure such as GED. The authors of GREED conduct experiments on GED estimation using three datasets, namely AIDS, IMDB and LINUX, which all consist of real-world graphs. For each of the three datasets, the GREED Model is trained using a portion of graph pairs and tested using others. More details are given in Appendix C.3.

The trained checkpoints of GREED are available online, so we simply use them.

4.3.2. Evaluation Setup

Metamorphic Relation. A family of metamorphic transformations $\mathbb{T}$ is used, where $\mathbb{T} = \{T_{i,op} \mid i = 1, 2, v \in Op\}$. For $q = \langle G_1, G_2 \rangle$, $T_{i,op}$ performs a graph edit operation op to G_i, yielding a new graph pair q'. Since the operation op already takes one step, we have the metamorphic relation $GED(q) \leq GED(q') + 1$, as shown in Figure 7.

The *MAgg(n)* setting (refer to Section 4.1.2) is used to generate Ego Metamorphic Graphs.

Aggregation Model. A simplified MPNN model (see Section 3.3) is used as aggregation model. The message function M_t is set to the identity function. To better handle regression task, we normalize and then discretize the base model predictions into 200 bins, representing them using one-hot vectors. For the LINUX and IMDB datasets, the state update function U_t is set to $U_t(h_u^t, m_u^{t+1}) = MLP(h_u^t \parallel m_u^{t+1})$, i.e. a multi-layer perceptron operating on the concatenation of node state and message. The hidden dimensions of the MLP are $200, 100, 50$ and the activation function is $ReLU$. For the AIDS dataset, a linear state update function is used to alleviate overfitting.

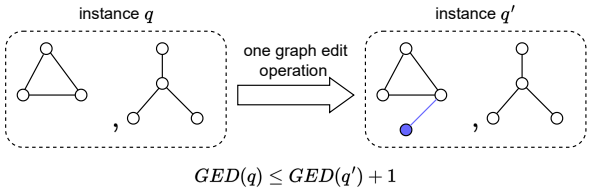

Figure 7. Metamorphic relation in GED.

Table 4. Regression result for GED in RMSE (lower is better).

Dataset	GREED	MAgg(50)
AIDS	0.7957	0.7994
LINUX	0.4151	**0.2409**
IMDB	6.7341	**6.3107**

Training the Aggregation Model. We use 50% of the training set, and construct an Ego Metamorphic Graph for each instance. The aggregation model is trained for 500 epochs with a 0.0001 learning rate and a 0.5 dropout.

Testing. The Root Mean Square Error (RMSE) is used as evaluation metric, as in (Ranjan et al., 2022). The RMSE is defined as $RMSE = \sqrt{\frac{1}{n}\sum_i (y_i - \hat{y}_i)^2}$, where $\hat{y}_i$ is the GED ground-truth and y_i is the estimation of model. The process of building Ego Metamorphic Graphs and training the aggregation model is repeated for 10 times, and the mean RMSE is reported.

4.3.3. Regression Result

In Table 4, we compare the RMSE score of base model (the GREED column) and base model augmented with MAGG (the MAgg(50) column). On IMDB and LINUX datasets, there are considerable RMSE improvements after augmenting base model using MAGG. And a 42% improvement relative to base model is achieved on LINUX dataset. This result shows using metamorphic relations for test-time augmentation is helpful. However, on AIDS dataset, MAGG provides no improvements, and we think this is because graphs have various node labels in AIDS dataset. Because of the existence of node labels, a random graph edit operation is more likely to increase the GED, making the metamorphic relation less effective.

5. Conclusion

Using machine learning techniques to help solve combinatorial problems is an emerging research field. However, the use of TTA in this field remains unexplored. In this paper, we propose MAGG, which uses metamorphic relations in target problems to enhance the corresponding machine learning models at inference time. The key extension of MAGG over standard TTA is the utilization of metamorphic relations, which helps MAGG (1) to be applicable to many combinatorial problems where there are sufficient metamorphic relations while few label-preserving transformations (2) to be able to leverage the semantic information encoded in metamorphic relations to achieve better augmentation performance. Also, the use of Graph Neural Network model to aggregate base model predictions helps the MAGG method to be highly adaptable to different combinatorial problems and different models. The evaluation result shows significant improvements over base models, underscoring the importance of incorporating metamorphic relations and effectiveness of the proposed MAGG method.

Acknowledgements

We thank anonymous reviewers for their insights on improving this work. This work is supported in part by the National Key Research and Development Program of China (No. 2022YFB3104004), National Natural Science Foundation of China (NSFC) (Grant No. 61932012, 62232016), the Key Research Program of Frontier Sciences, CAS (Grant No. ZDBS-LY-7006), the Youth Innovation Promotion Association of the Chinese Academy of Sciences (YICAS) (Grant No. Y2021041).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bai, Y., Ding, H., Bian, S., Chen, T., Sun, Y., and Wang, W. Simgnn: A neural network approach to fast graph similarity computation. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pp. 384–392, 2019.

Bai, Y., Ding, H., Gu, K., Sun, Y., and Wang, W. Learning-based efficient graph similarity computation via multi-scale convolutional set matching. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):3219–3226, Apr. 2020. doi: 10.1609/aaai.v34i04.5720. URL https://ojs.aaai.org/index.php/AAAI/article/view/5720.

Bengio, Y., Lodi, A., and Prouvost, A. Machine learning for combinatorial optimization: a methodological tour d¡¯horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

Bünz, B. and Lamm, M. Graph neural networks and boolean satisfiability. *arXiv preprint arXiv:1702.03592*, 2017.

Cappart, Q., Chételat, D., Khalil, E. B., Lodi, A., Morris, C., and Veličković, P. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24(130):1–61, 2023.

Chen, T. Y., Kuo, F.-C., Liu, H., Poon, P.-L., Towey, D., Tse, T., and Zhou, Z. Q. Metamorphic testing: A review of challenges and opportunities. *ACM Computing Surveys (CSUR)*, 51(1):1–27, 2018.

Chun, S., Lee, J. Y., and Kim, J. Cyclic test time augmentation with entropy weight method. In *Uncertainty in Artificial Intelligence*, pp. 433–442. PMLR, 2022.

Duan, H., Vaezipoor, P., Paulus, M. B., Ruan, Y., and Maddison, C. Augment with care: Contrastive learning for combinatorial problems. In *International Conference on Machine Learning*, pp. 5627–5642. PMLR, 2022.

Dwarakanath, A., Ahuja, M., Sikand, S., Rao, R. M., Bose, R. J. C., Dubash, N., and Podder, S. Identifying implementation bugs in machine learning based image classifiers using metamorphic testing. In *Proceedings of the 27th ACM SIGSOFT international symposium on software testing and analysis*, pp. 118–128, 2018.

Gao, X., Xiao, B., Tao, D., and Li, X. A survey of graph edit distance. *Pattern Analysis and applications*, 13:113–129, 2010.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Gu, J., Purdom, P. W., Franco, J., and Wah, B. W. Algorithms for the satisfiability (sat) problem: A survey. *Satisfiability problem: Theory and applications*, 35:19–152, 1996.

Joshi, C. K., Laurent, T., and Bresson, X. An efficient graph convolutional network technique for the travelling salesman problem. *arXiv preprint arXiv:1906.01227*, 2019.

Kim, I., Kim, Y., and Kim, S. Learning loss for test-time augmentation. *Advances in Neural Information Processing Systems*, 33:4163–4174, 2020.

Kimura, M. Understanding test-time augmentation. In *International Conference on Neural Information Processing*, pp. 558–569. Springer, 2021.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

Korte, B. H., Vygen, J., Korte, B., and Vygen, J. *Combinatorial optimization*, volume 1. Springer, 2011.

Li, Y., Gu, C., Dullien, T., Vinyals, O., and Kohli, P. Graph matching networks for learning the similarity of graph structured objects. In *International conference on machine learning*, pp. 3835–3845. PMLR, 2019.

Lyzhov, A., Molchanova, Y., Ashukha, A., Molchanov, D., and Vetrov, D. Greedy policy search: A simple baseline for learnable test-time augmentation. In *Conference on Uncertainty in Artificial Intelligence*, pp. 1308–1317. PMLR, 2020.

Mazyavkina, N., Sviridov, S., Ivanov, S., and Burnaev, E. Reinforcement learning for combinatorial optimization: A survey. *Computers & Operations Research*, 134: 105400, 2021.

Mocerino, L., Rizzo, R. G., Peluso, V., Calimera, A., and Macii, E. Adaptta: adaptive test-time augmentation for reliable embedded convnets. In *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6. IEEE, 2021.

Moshkov, N., Mathe, B., Kertesz-Farkas, A., Hollandi, R., and Horvath, P. Test-time augmentation for deep learning-based cell segmentation on microscopy images. *Scientific reports*, 10(1):5068, 2020.

Naidu, P., Gudaparthi, H., and Niu, N. Metamorphic testing for convolutional neural networks: Relations over image classification. In *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, pp. 99–106. IEEE, 2021.

Ozolins, E., Freivalds, K., Draguns, A., Gaile, E., Zakovskis, R., and Kozlovics, S. Goal-aware neural sat solver. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, 2022.

Papadimitriou, C. H. and Steiglitz, K. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

Pérez, J. C., Alfarra, M., Jeanneret, G., Rueda, L., Thabet, A., Ghanem, B., and Arbeláez, P. Enhancing adversarial robustness via test-time transformation ensembling. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 81–91, 2021.

Prates, M., Avelar, P. H. C., Lemos, H., Lamb, L. C., and Vardi, M. Y. Learning to solve np-complete problems: a graph neural network for decision tsp. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence and Thirty-First Innovative Applications of Artificial Intelligence Conference and Ninth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'19/IAAI'19/EAAI'19. AAAI Press, 2019. ISBN 978-1-57735-809-1. doi: 10.1609/aaai.v33i01.33014731. URL https://doi.org/10.1609/aaai.v33i01.33014731.

Ranjan, R., Grover, S., Medya, S., Chakaravarthy, V., Sabharwal, Y., and Ranu, S. Greed: A neural framework for learning graph distance functions. *Advances in Neural Information Processing Systems*, 35:22518–22530, 2022.

Segura, S., Fraser, G., Sanchez, A. B., and Ruiz-Cortés, A. A survey on metamorphic testing. *IEEE Transactions on software engineering*, 42(9):805–824, 2016.

Segura, S., Towey, D., Zhou, Z. Q., and Chen, T. Y. Metamorphic testing: Testing the untestable. *IEEE Software*, 37(3):46–53, 2018.

Selsam, D., Lamm, M., Bünz, B., Liang, P., de Moura, L., and Dill, D. L. Learning a SAT solver from single-bit supervision. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HJMC_iA5tm.

Shanmugam, D., Blalock, D., Balakrishnan, G., and Guttag, J. Better aggregation in test-time augmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1214–1223, 2021.

Shorten, C. and Khoshgoftaar, T. M. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

Sun, Z., Zhang, W., Mou, L., Zhu, Q., Xiong, Y., and Zhang, L. Generalized equivariance and preferential labeling for gnn node classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 8395–8403, 2022.

Vinyals, O., Fortunato, M., and Jaitly, N. Pointer networks. *Advances in neural information processing systems*, 28, 2015.

Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S., and Vercauteren, T. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 338:34–45, 2019.

Wang, S. and Su, Z. Metamorphic testing for object detection systems. *arXiv preprint arXiv:1912.12162*, 2019.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2018.

Zambito, L. The traveling salesman problem: a comprehensive survey. *Project for CSE*, 4080, 2006.

Zhang, W., Sun, Z., Zhu, Q., Li, G., Cai, S., Xiong, Y., and Zhang, L. Nlocalsat: boosting local search with solution prediction. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021a. ISBN 9780999241165.

Zhang, Z. and Zhang, Y. Elimination mechanism of glue variables for solving sat problems in linguistics. In *The Asian Conference on Language*, pp. 147–167, 2021.

Zhang, Z., Wang, P., Guo, H., Wang, Z., Zhou, Y., and Huang, Z. Deepbackground: Metamorphic testing for deep-learning-driven image recognition systems accompanied by background-relevance. *Information and Software Technology*, 140:106701, 2021b.

A. Boolean Satisfiability Prediction

A.1. Formal Definition

Definition A.1. (Boolean Formula and Conjunctive Normal Form). A boolean formula ϕ is an expression constructed from boolean variables and three logical operators which are conjunction ($\wedge$), disjunction ($\vee$) and negation ($\neg$). A boolean formula in conjunctive normal form (CNF) is a conjunction of clauses, where a clause is a disjunction of literals and a literal is a boolean variable or its negation.

Definition A.2. (Assignment and Satisfiability). An assignment τ is a map from boolean variables to boolean values. A boolean formula ϕ is satisfiable if there exists an assignment under which ϕ evaluates to $True$. Otherwise ϕ is unsatisfiable.

Definition A.3. (Boolean Satisfiability Problem). Given a formula ϕ in CNF, the Boolean Satisfiability Problem (SAT) aims to determine whether ϕ is satisfiable.

A.2. Proof of Metamorphic Relation

Proposition A.4. *Suppose ϕ is a boolean formula, and ϕ' is obtained by assigning a boolean value v to a boolean variable x_k in ϕ. Then, if ϕ' is satisfiable, then ϕ is satisfiable.*

Proof. Because ϕ' is satisfiable, there exists an assignment τ' such that ϕ' evaluates to $True$ under τ'. We define an assignment τ of ϕ as follows. For varaibles x_i which are also varaibles of ϕ', $\tau(x_i) := \tau'(x_i)$. And for x_k, $\tau(x_k) := v$. Because ϕ' is obtained by assigning a boolean varaible x_k to v in ϕ and τ' is a satisfiable assignment for ϕ', τ is a satisfiable assignment for ϕ. That is, ϕ is satisfiable. □

A.3. Base Model Details

NeuroSAT (Selsam et al., 2019) is used as base model, and we briefly introduce the details of the NeuroSAT model in this section.

A.3.1. Problem Encoding

In NeuroSAT architecture, an SAT instance is encoded as a heterogeneous graph. Specifically, there are two types of nodes: (1) one node for every clause, and (2) one node for every literal. Also, there are two types of edges: (1) one edge between literal and the clause it appears in, and (2) two edges between one literal and its negation.

A.3.2. Model Architecture

NeuroSAT adopts an Neural Message Passing architecture (Gilmer et al., 2017). The inference of NeuroSAT consists of many iterations. At each iteration, the nodes representing clauses first collect messages from their neighbors and update their embeddings, the nodes representing literals then collect messages from their neighbors and update their embeddings. The update rules are given by MLPs and LSTMs. Finally, the average of literal votes are computed to give satisfiability prediction.

A.3.3. Training Hyperparameters

The NeuroSAT model is trained on $SR(U(10, 40))$ where SAT instances of 10 to 40 variables are uniformly sampled. The instances are continuously sampled during training. The dimension of literal embeddings is $d = 128$. For each MLP, 3 hidden layers are used. The message passing is performed for 26 iterations. The model is trained using ADAM optimizer, with a $2 \cdot 10^{-5}$ learning rate. The model is trained with batches, where each batch contains 12000 nodes.

A.3.4. Reproducing Results

We train the NeuroSAT model using the same settings as in the original paper. (Selsam et al., 2019) reports a 0.85 accuracy on $SR(40)$, and our trained NeuroSAT model achieves a 0.8444 accuracy on the same dataset, which is close to the reported accuracy.

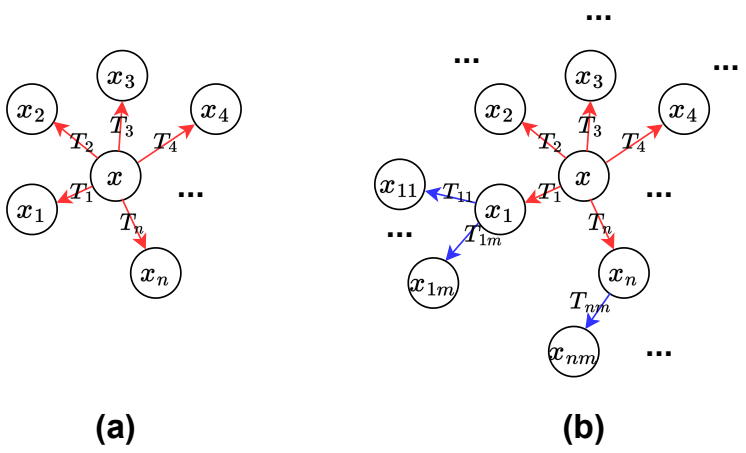

Figure 8. Visualization of MAgg setting

A.4. Dataset Details

The dataset used in the SAT task is $SR(n)$, proposed in (Selsam et al., 2019). $SR(n)$ is a distribution over random SAT instances with n variables. It generates a pair of SAT instances with one satisfiable while the other not, which two differ by negating one literal. $SR(n)$ works as follows: It first create clauses by randomly choosing k variables without replacement and negating each one with 50% probability. This process continues until adding a specific clause makes the SAT problem unsatisfiable (checked using an SAT solver). Negating a single literal in the final clause results in a satisfiable SAT instance. Finally, the pair of SAT instances is generated.

The base model is trained using $SR(U(10, 40))$ in (Selsam et al., 2019), and we use $SR(40), SR(80)$ and $SR(120)$ for evaluation.

A.5. Ego Metamorphic Graph Generation Setting

We visualize the two settings used to generate Ego Metamorphic Graph for each instance. Figure 8 (a) shows the MAgg(n) setting, in which only first level metamorphic transformations are used. Figure 8 (b) shows the MAgg(n, m) setting, in which only first and second level metamorphic transformations are used.

A.6. Impact of Number of Samples

In this section, we analyze the effect of the number of metamorphic transformations samples.

A.6.1. Setting

We use the trained aggregation model in MAgg(10), MAgg(10, 10) settings, and change the number of metamorphic transformation samples fed to the aggregation model.

A.6.2. Results

The changes of classification accuracy in regard to the number of samples are shown on Figure 9 and 10.

Figure 9 shows the effect of the number of first level metamorphic transformation samples (denoted as n_1, the x-axis) on classification accuracy in the MAgg(10) setting. We observed that even though the aggregation models are trained with 10 samples, they can well generalize to larger number of samples, yielding better classification result. Also, the marginal effect of number of samples is decreasing, and only a small number of samples can yield satisfactory results.

Figure 10 shows the effect of the number of first level metamorphic transformation samples (denoted as n_1, the x-axis) and the number of first level metamorphic transformation samples (denoted as n_2, shown in the legend) on classification

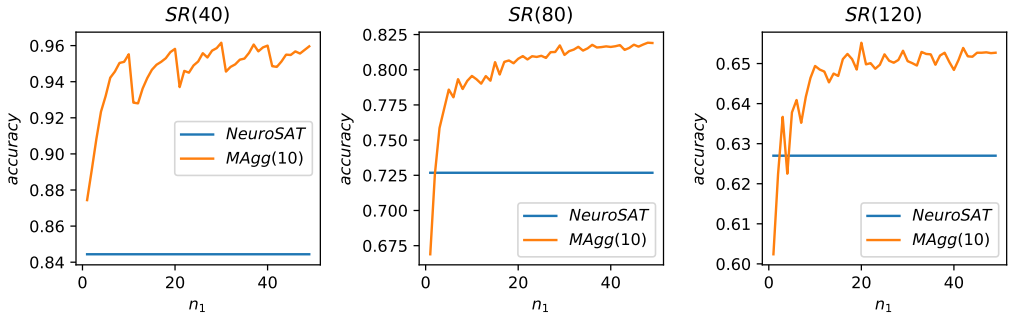

Figure 9. Effect of the number of first level metamorphic transformation samples

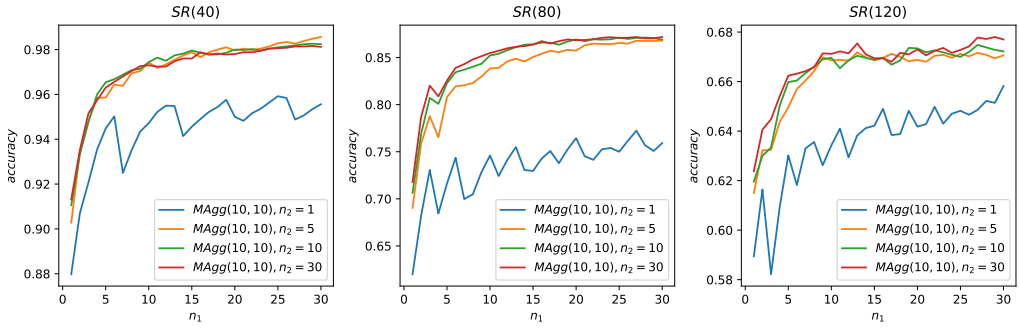

Figure 10. Effect of the number of first and second level metamorphic transformation samples

accuracy in the MAgg(10, 10) setting. Similar as the MAgg(10) setting, the aggregation model can generalize to larger number of samples with better classification accuracy. The observation on marginal effect of the number of samples also holds in MAgg(10, 10) setting.

A.7. Explaining the Effectiveness

In this section, we analyze the reason why the MAGG method is so effective in the SAT task. For simplicity, we consider the MAgg(n) setting. In MAgg(n) setting, for each instance x, n metamorphic transformations as shown in Figure 3 are sampled, yielding n mutants $x_1, x_2 \cdots x_n$. We assume that the number of samples n is large enough. The aggregation model uses base model's predictions on $x, x_1, \cdots x_n$ to give a better prediction on x. Because all mutants have same status and SAT is a binary classification task, only the portion of satisfiable predictions matters, which we denote α. That is, $\alpha := \frac{|\{x_i | BaseModel(x_i) = True\}|}{n}$.

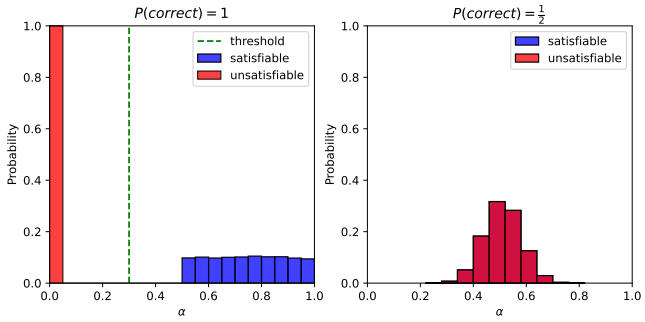

Figure 11. The α distributions in two edge cases

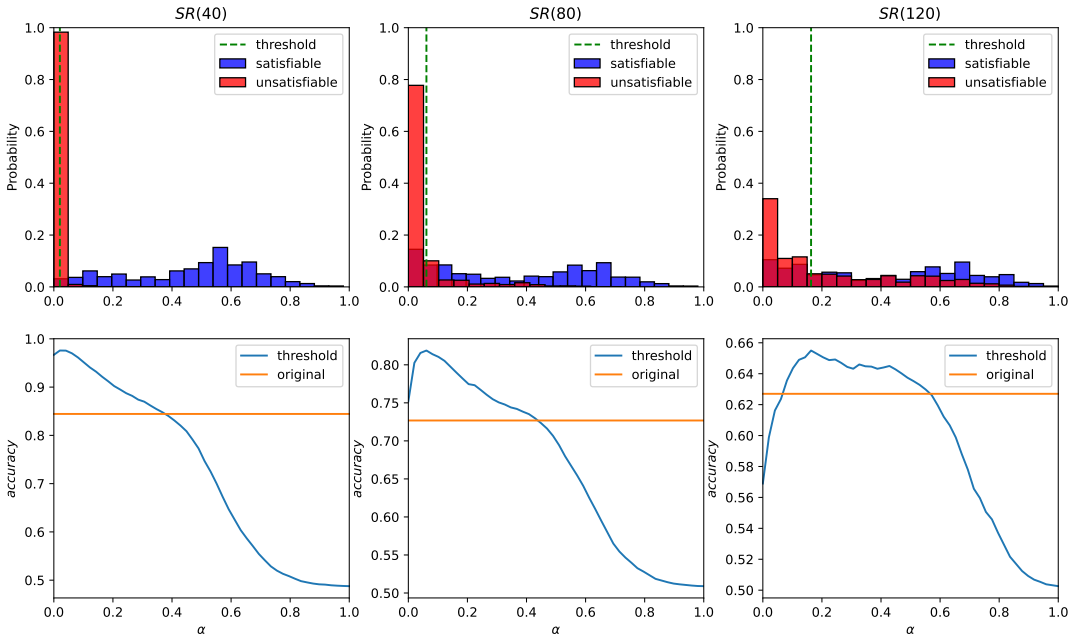

Figure 12. The α distributions on $SR(40)$, $SR(80)$ and $SR(120)$

Suppose the base model always make correct predictions. If x is satisfiable, then at least a half of x_is are satisfiable (since for one variable, at least one assignment is satisfiable). If x is unsatisfiable, then all x_is are unsatisfiable. Then the distribution of α under the two cases can be illustrated as in Figure 11 left. The satisfiable and unsatisfiable distributions can be well separated using a threshold.

Suppose the base model makes random independent predictions and it predict $True$ and $False$ with equal probability. Then the distributions of α when x is satisfiable or not will be the same, as in Figure 11 right.

Now, suppose the base model make independent predictions, and the probability of correct is $\frac{1}{2} < p < 1$. If p approaches 1, the α distribution will be similar to Figure 11 left. If p approaches $\frac{1}{2}$, the α distribution will be similar to Figure 11 right. Now for $\frac{1}{2} < p < 1$, the α distribution will be in some intermediate state between Figure 11 left and right. Figure 12 shows the α distribution of NeuroSAT on $SR(40)$, $SR(80)$ and $SR(120)$, with a decreasing probability of correct prediction. It can be observed that using a threshold on α to classify satisfiable and unsatisfiable instances can yield better classification accuracy.

A.8. Comparison with Standard TTA

In this section, we compare the performance of the proposed MAGG with standard TTA.

A.8.1. SETTING

The metamorphic transformation used by MAGG (see Figure 3) is not label-preserving, thus can not be used by Standard TTA. To apply Standard TTA for SAT prediction, we use the Add Unit Literal (AU) transformation (see (Duan et al., 2022)). The AU transformations uses a new literal l. It adds one clause containing only l, and add $\neg l$ to some other clauses, as depicted in Figure 13. The satisfiability of SAT instance will not change after apply the AU transformation. For each instances, 10 AU transformations are sampled. The process of sampling AU transformations is repeated for 10 times, and the mean and standard deviation are reported.

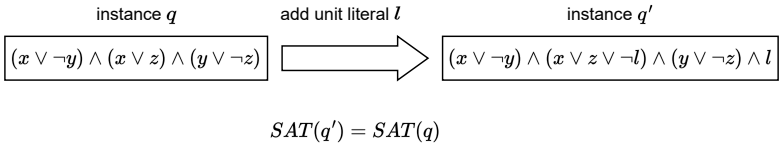

Figure 13. Label-preserving transformation AU in SAT

Table 5. Classification result for SAT in accuracy

Dataset	NeuroSAT	MAgg(10)	STTA(10)
SR(40)	0.8444	0.9548 ± 0.0026	0.8431 ± 0.0006
SR(80)	0.7268	0.7936 ± 0.0020	0.7302 ± 0.0022
SR(120)	0.6270	0.6412 ± 0.0027	0.6287 ± 0.0019

Table 6. Classification result for SAT in accuracy

Model	NeuroSAT	MAgg(10)	MAgg_GCN(10)	MAgg(10, 10)	MAgg_GCN(10, 10)
SR(40)	0.8444	0.9548 ± 0.0026	0.9559 ± 0.0025	0.9757 ± 0.0021	0.9838 ± 0.0015
SR(80)	0.7268	0.7936 ± 0.0020	0.7920 ± 0.0023	0.8533 ± 0.0035	0.8430 ± 0.0025
SR(120)	0.6270	0.6412 ± 0.0027	0.6446 ± 0.0035	0.6643 ± 0.0026	0.6626 ± 0.0020

A.8.2. RESULTS

In Table 5, we compare the classification accuracy of base model (the NeuroSAT column), base model augmented with MAGG (the MAgg(10) column), and base model augmented with standard TTA (the STTA(10) column). For all n, the classification accuracy of Standard TTA on $SR(n)$ is almost the same as NeuroSAT, and far inferior to MAGG. We speculate that this is because (1) the AU transformation is not effective (it makes instances more complex and harder for the base model to classify), and (2) the fixed average aggregation used by Standard TTA can not adapt to base model's characteristics. Thus, the leverage of non-label-preserving metamorphic relations and a trainable aggregation model are necessary.

A.9. Ablation Study

A.9.1. SETTING

The ablation study is conducted by replacing the simplified MPNN aggregation model with other Graph Neural Network architectures. The Graph Convolutional Network (GCN) (Kipf & Welling, 2017) is used for substitution. The state update rule of GCN is,

$$h_u^{t+1} = \sigma\Big(\sum_{v \in N(u) \cup \{u\}} \frac{1}{\sqrt{d_u \cdot d_v}} h_v^t W^t\Big)$$

where d_u is the degree of node u, and W^t is the parameter matrix at time step t, and σ is the activation function. The process of building Ego Metamorphic Graph and training the aggregation model is repeated for 10 times, and the mean and standard deviation are reported.

A.9.2. RESULTS

In Table 6, we compare the classification accuracy of base model (the NeuroSAT column), base model augmented with MAGG (the MAgg(10) and MAgg(10,10) columns), and base model augmented with MAGG in which GCN is used as aggregation model (the MAgg_GCN(10) and MAgg_GCN(10,10) columns). For all n, the performance of GCN aggregation model is close to the simplified MPNN aggregation model on $SR(n)$. We speculate that this is because the base model predictions on transformed instances matter more than predictions on original instances, and the two aggregation models leverage predictions on transformed instances in the same way.

B. Decision TSP Satisfiability Prediction

B.1. Formal Definition

Definition B.1. (Hamilton Cycle). Given an undirected graph $G = \langle V, E \rangle$, a Hamilton cycle in G is a cycle (a path in which only the first and last vertices are equal) that visits each vertex $v \in V$ exactly once.

Definition B.2. (Traveling Salesman Problem). Given a complete undirected weighted graph $G = \langle V, E, w \rangle$ where $w : E \to \mathbb{R}$ is the edge weight function. The Traveling Salesman Problem (TSP) aims to determine the Hamilton cycle with minimal cost, where the cost of a path is defined as the sum of its edge weights.

Definition B.3. (Decision Traveling Salesman Problem). An instance of decision Traveling Salesman Problem (Decision TSP) is a pair $\langle G, C \rangle$, where G is a complete undirected weighted graph and C is the target cost. Decision TSP aims to determine whether there exists a Hamilton cycle whose cost is lower than or equal to C. If so, we call this instance satisfiable, otherwise unsatisfiable.

Definition B.4. (2D Euclidean Decision TSP). The 2D Euclidean Decision TSP is a special type of Decision TSP, where the target graph G is a 2D Euclidean graph. That is, the vertices of G are points on Euclidean plane, and the weight function w is defined as the distance function in Euclidean plane.

Note that the weight function in 2D Euclidean Decision TSP satisfies triangular inequality, that is, $\forall u, v, t \in V, w(\langle u, v \rangle) \leq w(\langle u, t \rangle) + w(\langle t, v \rangle)$

B.2. Proof of Metamorphic Relation

Proposition B.5. *Suppose $\langle G, C \rangle$ is a 2D Euclidean TSP instance, and $\langle G', C \rangle$ is obtained by add a midpoint and corresponding edges to G in Euclidean plane. Then, if G' has a Hamilton cycle whose cost is $\leq C$, then G has a Hamilton cycle whose cost is $\leq C$.*

Proof. Suppose G' is obtained by adding a midpoint t. If G' has a Hamilton cycle $\tau' := u_0, u_1 \cdots, t = u_i, \cdots u_n = u_0$, such that $cost(\tau') := \sum_{i=0}^{n-1} w(\langle u_i, u_{i+1} \rangle) \leq C$. Then define a Hamilton cycle $\tau := u_0, u_1 \cdots, u_{i-1}, u_{i+1}, \cdots u_n = u_0$ of G which is obtained by removing t in τ'. Because of the triangular inequality $w(\langle u_{i-1}, u_{i+1} \rangle) \leq w(\langle u_{i-1}, t \rangle) + w(\langle t, u_{i+1} \rangle)$, $cost(\tau) \leq cost(\tau') \leq C$. That is, G has a Hamilton cycle whose cost is $\leq C$. □

B.3. Base Model Details

TSP-GNN (Prates et al., 2019) is used as base model, and we briefly introduce the details of the TSP-GNN model in this section.

B.3.1. Model Architecture

TSP-GNN uses a message passing architecture where messaging and updating functions are modeled using neural network. In addition to node embeddings, TSP-GNN uses edge embeddings to encode the information of edge weights, and uses vertex-edge adjacency rather than vertex-vertex adjacency. The target cost C is encoded to initial edge embedding using concatenation. The message passing is performed for many iterations. At each iteration, the nodes first collect messages from their neighbors and update their embeddings, the edges then collect messages from their sources and targets and update their embeddings. The messaging and updating functions are given by MLPs and RNNs. Finally, the edge embeddings are aggregated to give satisfiability prediction.

B.3.2. Training Hyperparameters

The TSP-GNN Model is trained on $TSP(0.01), TSP(0.02), TSP(0.05)$ and $TSP(0.10)$, respectively. For each dev, there is one trained TSP-GNN model. The model is then trained with Adam optimizer with binary cross entropy loss. The model is trained using batches. The dimension of node and edge embeddings are set to $d = 64$. Every MLP uses three layers with dimensions $64, 64, 64$ and $ReLU$ activation. The message passing is performed for 32 iterations. The models for all devs are trained for 1000 epochs.

Table 7. Classification result for Decision TSP in accuracy

Dataset	TSP-GNN	MAgg(10)	MAgg_GCN(10)
TSP(0.01)	0.6562	0.6812 ± 0.0039	0.6815 ± 0.0021
TSP(0.02)	0.8101	0.8321 ± 0.0029	0.8309 ± 0.0028

B.3.3. Reproducing Results

We train the TSP-GNN model using the same settings as in the original paper. (Prates et al., 2019) reports 0.66 accuracy on $TSP(0.01)$ and 0.80 accuracy on $TSP(0.02)$. Our trained TSP-GNN model achieves 0.6562 accuracy on $TSP(0.01)$ and 0.8101 accuracy on $TSP(0.02)$, which are close to the reported results.

B.4. Dataset Details

The dataset used in the Decision TSP task is $TSP(dev)$, proposed in (Prates et al., 2019). $TSP(dev)$ is a distribution over random 2d Euclidean Decision TSP instances, where the difficulty of instances is controlled by dev. The smaller dev is, the harder the generated instances are. This distribution contains the following steps:
(1) Sampling $n\ U(20, 40)$ points in a $\frac{\sqrt{2}}{2} \times \frac{\sqrt{2}}{2}$ square.
(2) Computing the edge weight function using Euclidean distance.
(3) Using a TSP solver to solve the TSP instance, yielding optimal cost C.
(4) Constructing a pair of decision TSP instances: $\langle G, (1+dev) \cdot C \rangle$ and $\langle G, (1-dev) \cdot C \rangle$, note that the former is satisfiable while the latter is not.

TSP-GNN is trained on $TSP(0.01)$, $TSP(0.02)$, $TSP(0.05)$ and $TSP(0.10)$ respectively, in (Prates et al., 2019). In this paper, we use $TSP(0.01)$ and $TSP(0.02)$ for evaluation.

B.5. Ablation Study

B.5.1. Setting

The setting is the same as in Section A.9.1.

B.5.2. Results

In Table 7, we compare the classification accuracy of base model (the NeuroSAT column), base model augmented with MAgg (the MAgg(10) column), and base model augmented with MAgg in which GCN is used as aggregation model (the MAgg_GCN(10) column). For all dev, the performance of GCN aggregation model is close to the simplified MPNN aggregation model on $TSP(dev)$. We speculate that this is because the base model predictions on transformed instances matter more than predictions on original instances, and the two aggregation models leverage predictions on transformed instances in the same way.

C. Graph Edit Distance Estimation

C.1. Formal Definition

Definition C.1. (Graph Isomorphism). Given two graphs $G_1 = \langle V_1, E_1 \rangle, G_2 = \langle V_2, E_2 \rangle$, an isomorphism $i : V_1 \to V_2$ is a bijection between V_1 and V_2, such that (1) $\forall \langle u, v \rangle \in E_1, \langle i(u), i(v) \rangle \in E_2$ and (2) $\forall \langle u', v' \rangle \in E_2, \langle i^{-1}(u'), i^{-1}(v') \rangle \in E_1$. If there exits an isomorphism for G_1, G_2, we say G_1, G_2 are isomorphic.

Definition C.2. (Graph Edit Operation). A Graph Edit Operation is a function that transforms a graph to another. In this paper, only the following set of graph edit operations are considered:
(1) Add a node.
(2) Delete an isolated node.
(3) Add an edge.
(4) Delete an edge.
(5) Relabel a node (if the graph is labeled).

Table 8. Dataset statistics for GED

Dataset	# Pairs(train)	# Pairs(val)	# Pairs(test)
AIDS	285600	28000	78400
LINUX	600000	40000	160000
IMDB	1380000	60000	360000

Definition C.3. (Graph Edit Sequence). A graph edit sequence (GES) s for G_1, G_2 is a sequence of graph edit operations, such that after these operations G_1, G_2 are isomorphic. The set of graph edit sequences for G_1, G_2 is denoted as $GES(G_1, G_2)$.

Definition C.4. (Graph Edit Distance). The Graph Edit Distance (GED) between G_1, G_2 is defined as the minimal length of graph edit sequences between G_1, G_2. That is, $GED(G_1, G_2) := \min_{s \in GES(G_1,G_2)} length(s)$, where $length(s)$ is defined as the number of graph edit operations in s.

C.2. Proof of Metamorphic Relation

Proposition C.5. *Suppose G_1, G_2 is a pair of graphs, and G_1', G_2' is obtained by performing one step graph edit operation op. Then, $GED(G_1, G_2) \leq GED(G_1', G_2') + 1$.*

Proof. Suppose the graph edit sequence for G_1', G_2' with minimal length is $s' \in GES(G_1', G_2')$. We construct s by concatenating *op* to s', then $s \in GES(G_1, G_2)$ and $length(s') = length(s) + 1$. So G_1, G_2 have a graph edit sequence of length $GED(G_1', G_2') + 1$. That is, $GED(G_1, G_2) \leq GED(G_1', G_2') + 1$. □

C.3. Base Model Details

GREED (Ranjan et al., 2022) is used as base model, and we briefly introduce the details of the GREED model in this section.

C.3.1. Model Architecture

GREED uses a Siamese Graph Neural Network architecture to embed graph into vector space, and the distance between embeddings of two graphs are used to compute GED. The Siamese Graph Neural Network architecture consists of two identical GNN with same parameters. The GNN structure first uses a pre-MLP to reduce node features to a desired dimension. Then, a Graph Isomorphism Network (GIN) structure is used to extract graph structure information. After that, the hidden representations in GIN are concatenated and fed to a post-MLP to yield an embedding of input graph. The L2 distance of two embedding vectors is produced to give an estimation of GED.

C.3.2. Training Hyperparameters

The GREED model is trained on AIDS, LINUX and IMDB, respectively. The number of GIN layers is set to 8 and the hidden dimensions are set to $d = 64$. The model is trained until there is less than 0.05% change in validation loss over a number of epochs.

C.4. Dataset Details

Three datasets are used in the GED task, namely AIDS, LINUX and IMDB.

- The AIDS dataset is composed of graphs that are built from the AIDS antiviral screen database. The graphs in this dataset depict molecular compounds with Hydrogen atoms omitted. In these representations, atoms are nodes, and chemical bonds are edges. Note the graphs are labeled.
- The LINUX dataset consists program dependence graphs. A graph represents a function, wherein nodes are statements and edges are dependency between statements. Note the graphs are unlabeled.
- The IMDB dataset consists of ego-networks of actors/actresses. The nodes are actors/actresses, and the edges are co-appearance relations. Note the graphs are unlabeled.

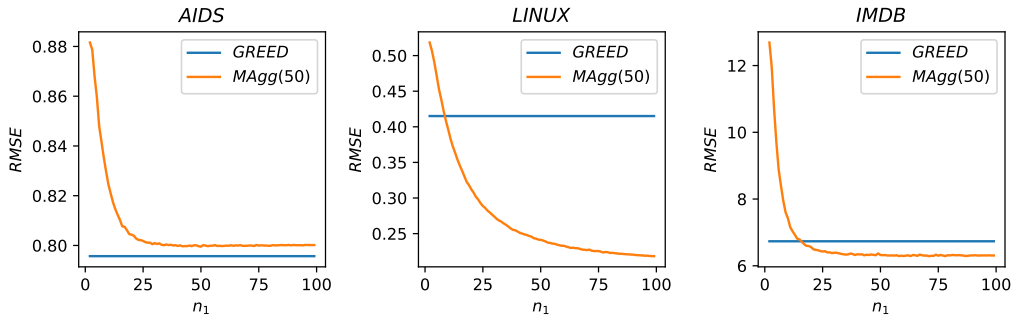

Figure 14. Effect of the number of first level metamorphic transformation samples

Table 9. Regression result for GED in RMSE (lower is better)

Dataset	GREED	MAgg(50)	MAgg_GCN(50)
AIDS	0.7957	0.7994 ± 0.0002	0.8184 ± 0.0024
LINUX	0.4151	0.2409 ± 0.0030	0.5709 ± 0.0017
IMDB	6.7341	6.3107 ± 0.0204	6.6896 ± 0.0011

For the GED task, graph pairs are constructed using graphs in each dataset, for training, validation, and testing. The statistics are shown in Table 8. GED computation algorithms are used to provide groundtruth.

GREED is trained on AIDS, LINUX, and IMDB, in (Ranjan et al., 2022). In this paper, we uses all the three datasets for evaluation.

C.5. Impact of Number of Samples

In this section, we analyze the effect of the number of metamorphic transformations samples.

C.5.1. Setting

We use the trained aggregation models on MAgg(50) setting, and change the number of metamorphic transformation samples fed to the aggregation model.

C.5.2. Results

The changes of RMSE in regard to the number of samples are shown on Figure 14, in which the number of first level metamorphic transformations is denoted as n_1. In AIDS dataset, the aggregation model does not help to improve RMSE, and the change of number of samples still can not help with it. In LINUX and IMDB dataset, the aggregation models have learned to aggregate predictions. It can be observed that even though the aggregation models are trained with 50 samples, they can well generalize to larger number of samples, yielding better regression result. Also, the marginal effect of number of samples is decreasing.

C.6. Ablation Study

C.6.1. Setting

The setting is the same as in Section A.9.1.

C.6.2. Results

In Table 9, we compare the RMSE score of base model (the NeuroSAT column), base model augmented with MAGG (the MAgg(50) column), and base model augmented with MAGG in which GCN is used as aggregation model (the MAgg_GCN(50) column). The performance of GCN aggregation model is worse than the simplified MPNN aggregation

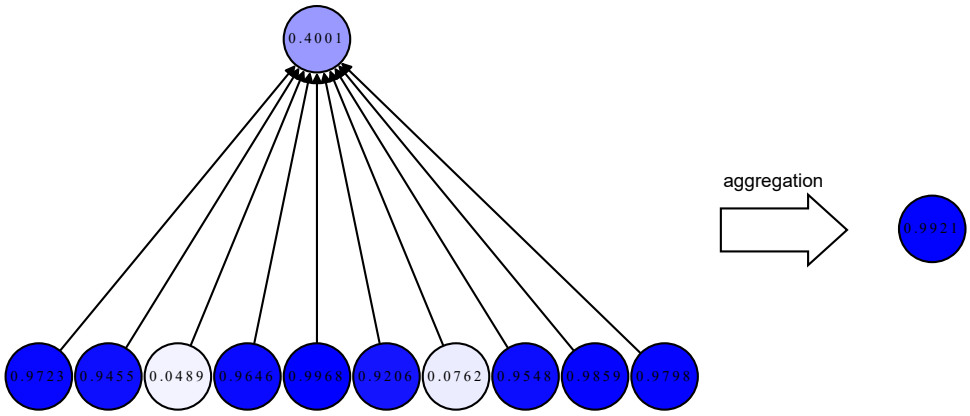

Figure 15. An example in which MAGG corrects NeuroSAT's prediction

model, and is even worse than the base model on some datasets. We speculate that this is because (1) the predictions of central instances should be handled separately and (2) the use of MLP as the state update function is necessary.

D. Detailed Explanations

D.1. MRs Encompass Helpful Information

Since metamorphic relations can be used to detect errors in models, they should encompass information that enables the model to make better predictions. For example, consider the metamorphic relation in Figure 3. Suppose one transforms instance q into multiple instances q_i using metamorphic relation in Figure 3. Consider the case when the base model predicts $SAT(q) = 0$, while for most of q_i, it predicts $SAT(q_i) = 1$. Then according to the metamorphic relation, it is more likely that $SAT(q) = 1$. Thus one can correct the base model's prediction using its predictions on transformed instances.

Figure 15 gives an example in which MAGG corrects a prediction of NeuroSAT (Selsam et al., 2019) using metamorphic relations. On the left hand side, the nodes denote instances, where q is on top and q_is are on the bottom. The digits within the nodes denote NeuroSAT's predictions, with their values illustrated using colors. Aggregated prediction by MAGG is shown on the right hand side. NeuroSAT predicts q to be unsatisfiable, while predicting most of the q_is to be satisfiable. This violates the metamorphic relation. MAGG corrects NeuroSAT's prediction on q.

D.2. MRs Are Hard to Learn from Data

It is not easy for a machine learning model to learn metamorphic relations directly from data, according to following reasons. First, metamorphic testing has been applied to test deep learning based utilities (Zhang et al., 2021b; Wang & Su, 2019; Naidu et al., 2021; Dwarakanath et al., 2018), and has proven to be effective. The effectiveness of metamorphic testing shows that machine learning models do not always learn metamorphic relations well. Second, a training set with fixed size is not likely to contain enough groups of instances that follow the metamorphic relations. For example, consider the metamorphic relation in Figure 3 and the training set sampled from $SR(U(10, 40))$. For a pair of instances $\langle q, q' \rangle$ to follow this relation, q' must be obtained by substituting one variable in q. However, there is a low probability that such two instances are both sampled from $SR(U(10, 40))$, thus the model trained using $SR(U(10, 40))$ is not likely to learn it. During evaluation, we found metamorphic relations in Figure 3, 6, 7 are easily violated by the corresponding base models.

D.3. Effective Label-Preserving Transformations Are Rare in Combinatorial Problems

It is not easy to find effective label-preserving transformations for many combinatorial problems. For computer vision tasks, there is a large amount of well studied effective label-preserving transformations, such as flipping, rotation, translation, etc (Shorten & Khoshgoftaar, 2019). However, in the area of combinatorial problem solving, such effective label-preserving transformations are rare. For SAT, (Duan et al., 2022) proposes six label-preserving transformations, namely Unit Propagation (UP), Add Unit Literal (AU), Pure Literal Elimination (PL), Subsumed Clause Elimination (SC), Clause Resolution (CR), and Variable Elimination (VE). However, UP, PL, SC and CR are only applicable to specific SAT instances, and AU

and VE make instances much more complex, harder for the base model to predict. Thus none of these label-preserving transformations is effective. For Decision TSP and GED, it is hard to find any non-trivial label-preserving transformations.