# LEARNING SYMBOLIC MODELS FOR GRAPH-STRUCTURED PHYSICAL MECHANISM

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Graph-structured physical mechanisms are ubiquitous in real-world scenarios, thus revealing underneath formulas is of great importance for scientific discovery. However, classical symbolic regression methods fail on this task since they can only handle input-output pairs that are not graph-structured. In this paper, we propose a new approach that generalizes symbolic regression to graph-structured physical mechanisms. The essence of our method is to model the formula skeleton with a message-passing flow, which helps transform the discovery of the skeleton into the search of the message-passing flow. Such a transformation guarantees that we are able to search a message-passing flow, which is efficient and Pareto-optimal in terms of both accuracy and simplicity. Subsequently, the underneath formulas can be identified by interpreting component functions of the searched message-passing flow reusing classical symbolic regression methods. We conduct extensive experiments on datasets from different physical domains, including mechanics, electricity, and thermology, and on real-world datasets of pedestrian dynamics without ground-truth formulas. The experimental results not only verify the rationale of our design but also demonstrate that the proposed method can automatically learn precise and interpretable formulas for graph-structured physical mechanisms.

## 1 INTRODUCTION

For centuries, the development of the natural sciences has been based on human intuition to abstract physical mechanisms represented by symbolic models, i.e., mathematical formulas, from experimental data recording the phenomena of nature. Among these developments, many mechanisms are naturally graph-structured (Leech, 1966), where the physical quantities are associated with individual objects (e.g., mass), pair-wise relationships (e.g., force) and the whole system (e.g., overall energy), corresponding to three types of variables on graphs: node/edge/global variables. For example, as shown in Figure 1(a), the mechanical interaction mechanism in multi-body problem corresponds to a graph with masses ($m_i$), positions ($\vec{V}_i$) as attributes of nodes, and spring constants ($k_{ij}$) as attributes of edges, which, together with the graph connectivity, yields the acceleration as output attributes of nodes; while in the case of resistor circuit, nodes and edges correspond to voltages and resistances, respectively, and these attributes define a graph-level overall power of the circuit.

In the past few years, Symbolic Regression (SR) (Sahoo et al., 2018; Schmidt & Lipson, 2009; Udrescu et al., 2020), which searches symbolic models $y = \mathcal{F}(x)$ from experimentally obtained input-output pairs $\{(x, y)\}$ with $\mathcal{F}$ being an explicit formula, has become a promising approach trying to automate scientific discovery. Traditional SR methods include genetic programming-based methods (Schmidt & Lipson, 2009; Fortin et al., 2012) working by generating candidate formulas by "evolution" (i.e., manipulations), and deep learning-based methods (Li et al., 2019; Biggio et al., 2021; Zheng et al., 2021) utilizing sequence models to generate candidate formulas. However, these methods are designed for traditional SR problems on input-output pairs $\{(x, y)\}$, without considering graph information.

To exploit the inherent graph structure in physical mechanisms, as shown in Figure 1(b), SR on graphs aims to find a formula $\mathcal{F}$ that characterizes a mapping from input $\{\mathcal{G}, X\}$ to output $y$, with $X$ and $y$ both inside graph structure $\mathcal{G}$. To perform this, we need both fine exploitation of inherent graph structures of physical mechanisms and well achievement of flexibility when it comes to diverse

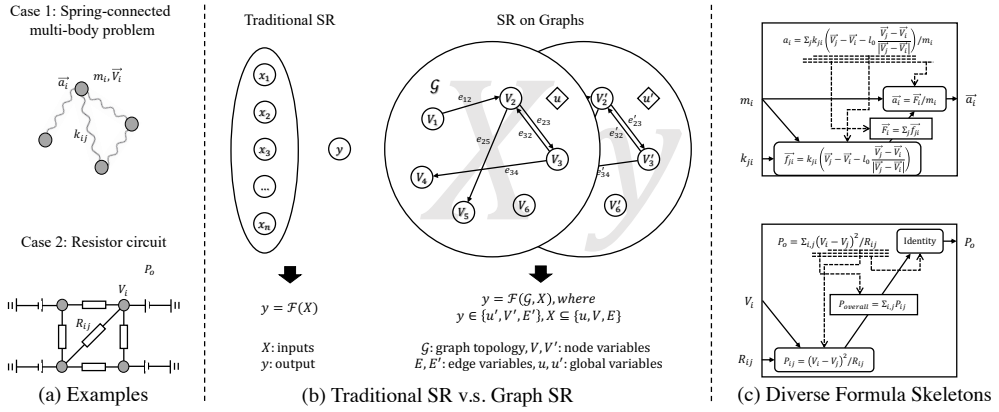



Figure 1: (a) Two examples of graph-structured physical mechanisms. (b) Illustration of traditional SR (left) and SR on graphs (right). (c) Two different formula skeletons in the two examples, where all formula components are interconnected accordingly to construct an overall formula.

forms of interaction between entities in the physical world. Recently, Graph Neural Network (GNN) is being incorporated into SR for discovering mechanisms behind particle interactions (Cranmer et al., 2020; Lemos et al., 2022). However, obvious setbacks exist that the message-passing flow of GNN, corresponding to the formula skeleton, required to be manually designed to learn the underlying mechanisms, is impractical because the formula skeletons usually remain unknown and are significantly different in diverse physical domains as shown in Figure 1(c).

To solve this problem, inspired by the correspondence between skeleton and message-passing flow in GNN, our core idea is to transform the discovery of skeleton into the search of message-passing flow, which paves the way for identifying the underneath formula by interpreting each component function in the searched message-passing flow. However, due to the coupling relationship between the skeleton and the component formula in the skeleton, neither of them can be independently identified, implying a huge highly entangled search space for both message-passing flow and component functions. To tackle this challenge, we formulate a bi-level optimization problem that searches for the message-passing flow by pruning strategy at the upper level on condition that its component functions have been optimized with deep learning (DL) at the lower level. Besides empirical accuracy, it is equally important but rather non-trivial to maintain both explicit interpretability and generalization ability in discovered formulas. We propose to search the Pareto-optimal message-passing flow between accuracy and simplicity by carefully designing a scoring function involving a complexity function of message-passing flows that optimizes in both aspects across different searching steps.

Our contributions can be summarized as the following three aspects,

- We generalize the problem of learning formulas with given skeletons (inductive bias) from graph data in Cranmer et al. (2020) by additionally learning the formula skeleton from data, which is essential for learning graph-structured physical mechanisms from diverse physical domains.
- We propose a novel method to learn graph-structured physical mechanisms from data without knowing the formula skeleton by searching the Pareto-optimal message-passing flows of GNN together with the symbolic models as components.
- We conduct experiments on five datasets from diverse physical domains, including mechanics, electricity, thermology, and two real-world datasets about pedestrian dynamics, demonstrating that our model can first automatically identify the correct skeleton, based on collected data instead of expert knowledge, and then learn the overall symbolic model for corresponding graph-structured physical mechanism.

## 2 THE PROPOSED METHOD

Before introducing the proposed method, we first formally define the the problem of symbolic regression on graphs.

**Definition 1 (Variables on Graphs)** *The topology of the graph is denoted as $\mathcal{G}$, and its variables include $\{\mathbf{V}, \mathbf{E}, \mathbf{u}\}$, where $\mathbf{V}$ denotes the set of node-level variables, $\mathbf{E}$ denotes the set of edge-level*

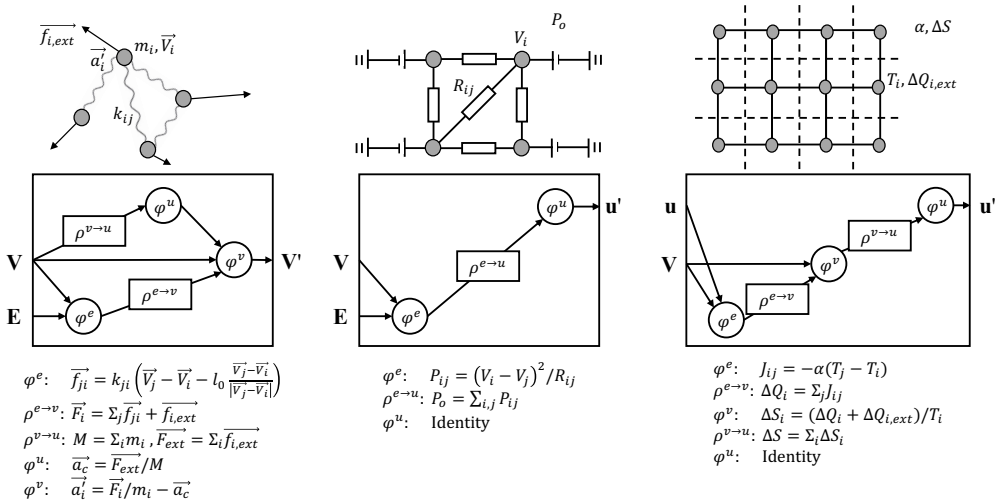



Figure 2: Examples of three different graph-structured physical mechanisms and their corresponding message-passing flows. Notice here the three examples differ from each other greatly in the layouts of message-passing flows. Every component of the message-passing flows represents a certain function (listed at the bottom) on the graph abstracted from the physical scenarios given in the first row.

*variables and* $\mathbf{u} \in \mathbb{R}^{n_u}$ *denotes a global variable which interacts with elements in* $\mathbf{V}$ *and* $\mathbf{E}$ *through topology* $\mathcal{G}$*. Specifically,* $\mathbf{v}_i \in \mathbb{R}^{n_v}$ *is the variable associated with the i-th node while* $\mathbf{e}_{ij} \in \mathbb{R}^{n_e}$ *is the variable associated with the edge connecting the i-th and j-th nodes.* $n_u$*,* $n_v$ *and* $n_e$ *denote the dimensions of the global variable, node variable and edge variables.*

**Definition 2 (Symbolic Regression on Graphs)** *Given a set of* $\{(\mathcal{G}_i, X_i, \mathbf{y}_i)\}$*, where* $X_i \subset \{\mathbf{V}, \mathbf{E}, \mathbf{u}\}$*,* $\mathbf{y}_i \in \{\mathbf{V}', \mathbf{E}', \mathbf{u}'\}$*, and variables with prime denotes output variables, we aim to find an accurate and compact formula* $\mathcal{F}(\cdot)$ *that fits* $\mathbf{y} = \mathcal{F}(\mathcal{G}, X)$*.*

### 2.1 Model Formula Skeleton with Message-Passing Flow

As shown in Figure 1(c), a formula $\mathcal{F}$ describing graph-structured physical mechanisms can always be decomposed into several formula components, each representing its association with individual node, pair-wise relationships between nodes or the whole system. As illustrated in the specific diagram, these components are interconnected according to the variable dependency, which is termed as "*skeleton*". Moreover, differences between two examples also indicate an existence of diverse skeletons underlying different physical mechanisms.

The key insight is that the skeleton has a strong correspondence with the message-passing flows in GNN, which differs a lot in various physical scenarios, including mechanics (Sanchez-Gonzalez et al., 2020), electricity (Zhang et al., 2019), thermology (Chamberlain et al., 2021). The message-passing flows can be diverse by cascading multiple blocks, changing/removing some functions and adjusting the embedding sizes. A block of full GNN contains the updating of edges, nodes and graph representation respectively as follows (Battaglia et al., 2018),

$$\begin{aligned} \mathbf{e}'_{ij} &= \phi^e\left(\mathbf{e}_{ij}, \mathbf{v}_i, \mathbf{v}_j, \mathbf{u}\right) & \overline{\mathbf{e}}'_i &= \rho^{e\to v}\left(E'_i\right) \\ \mathbf{v}'_i &= \phi^v\left(\overline{\mathbf{e}}'_i, \mathbf{v}_i, \mathbf{u}\right) & \overline{\mathbf{e}}' &= \rho^{e\to u}\left(E'\right) \\ \mathbf{u}' &= \phi^u\left(\overline{\mathbf{e}}', \overline{\mathbf{v}}', \mathbf{u}\right) & \overline{\mathbf{v}}' &= \rho^{v\to u}\left(V'\right) \end{aligned} \tag{1}$$

where $\phi(\cdot)$ denotes the message functions and $\rho(\cdot)$ denotes the aggregation functions. We provide three examples in different physical scenarios, as shown in Figure 2, to illustrate the well-defined analogy between message-passing flows and skeletons.

**Example 1 (Mechanics: Multi-body Kinematics)** *In this problem we aim to find particles' acceleration in the center-of-mass frame. We have an edge update function* $\phi^e$ *since particle pairs determine string forces, a global update function* $\phi^u$ *corresponding to the Center-of-Mass Theorem, while the aggregation function* $\rho^{e\to v}$ *of edges is based on the independent action principle of force.*

Table 1: Analogy between graph-structured formulas and GNN.

| computation steps of graph-structured formulas | architecture of message-passing networks |
|---|---|
| # intermediate variables | embedding size |
| dependent variables | message-passing connection |
| formulas with (multi-dimensional) values as input | message function $\phi$ |
| formulas with set as input | aggregation function $\rho$ |

**Example 2 (Electricity: Resistor Circuit)** *The objective of this problem is to find the overall power of a given resistor circuit. An edge update function $\phi^e$ is corresponding to the computation of single-resistor power utilizing Joule's Law, and an aggregation $\rho^{e\to u}$ from edge to global appears for summation to get overall power.*

**Example 3 (Thermology: Heat Conduction)** *The objective of this problem is to compute the entropy production rate. The edge update function $\phi^e$ corresponds to Fourier's Law of Heat Conduction, and the node update function $\phi^v$ corresponds to the Clausius entropy expression, followed by an aggregation $\rho^{v\to u}$ that sums up individual entropy production rates.*

### 2.2 Transforming into the Task of Message-Passing Flow Searching

The message-passing flows of GNN correspond to explicit meanings in the symbolic calculation for graph-structured mechanisms, which is summarized in Table 1. This strong resemblance inspired us to devise a transformation of the primitive SR task on graphs into a relatively more practical task of searching message-passing flows. There are two stages in our model: message-passing flow searching and message-passing flow-based SR. Specifically, at stage 1, we need to search the message-passing flow as the formula skeleton, and then at stage 2, we need to symbolize components into formulas and cascade them according to the skeleton to get the final graph-structured mechanisms.

With the above transformation, it is clear that we need to solve the following bi-level the optimization problem in stage 1, i.e.,

$$\{\mathbf{P}^*, \mathbf{M}^*\} = \arg\min_{\mathbf{M},\mathbf{P}} s(\mathbf{P}; \mathbf{M}), \tag{2}$$

where $\mathbf{M}$ denotes the message-passing flows, $\mathbf{P}$ denotes the parameters in the DL components, $\mathbf{M}^*$ and $\mathbf{P}^*$ denote the Pareto-optimal one, and $s(\cdot)$ gauges how well the finally learned formula obtained by $\mathbf{M}$ and $\mathbf{P}$ performs. However, there are two core challenges to learning formulas on graph: (i) considering simplicity and accuracy simultaneously for graph SR is difficult; (ii) the discrete search space composed of skeletons and the component formulas is prohibitively huge.

### 2.3 Searching Message-Passing Flows $\mathbf{M}$ (Formula Skeletons)

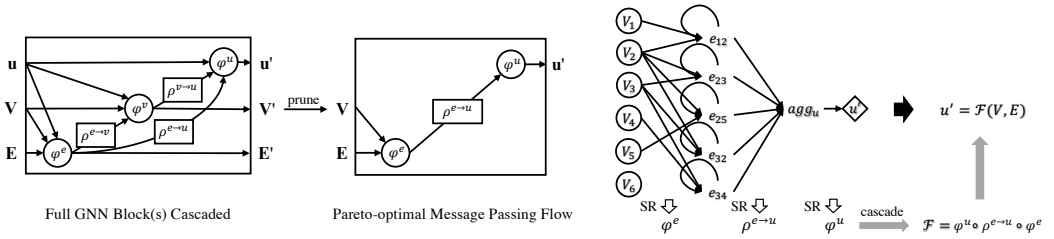
To deal with the first challenge, we are motivated to change equation 2 into

$$\mathbf{M}^* = \arg\min_{\mathbf{M}} s(\mathbf{P}^*; \mathbf{M}), \tag{3}$$

$$\text{s.t.} \quad \mathbf{P}^* = \arg\min_{\mathbf{P}} l(\mathbf{P}; \mathbf{M}), \tag{4}$$

where equation 3 and equation 4 are two optimization problems in upper level and lower level, $s(\mathbf{P}^*; \mathbf{M}) = l(\mathbf{P}^*; \mathbf{M}) + \lambda c(\mathbf{M})$ is the score taking both simplicity and accuracy into consideration, $l(\cdot)$ denotes the error loss of predicting outputs (see details in Appendix B.2), $\lambda$ is the weight, and $c(\cdot)$ denotes the complexity of message-passing flow. The design of complexity $c(\cdot)$ is flexible (see details in Appendix B.3), and we calculate the complexity as follows, (i) for each layer, the complexity can be calculated as the product of the embedding size of this layer and the number of inputs in this layers; (ii) the whole complexity of the message-passing flow can be calculated as the summation of the complexity of each layer. The optimization at the lower level is solved by training parameters $\mathbf{P}$ of a DL model given the structure $\mathbf{M}$, while the optimization at the upper level w.r.t. $\mathbf{M}$ is difficult because $\mathbf{M}$ forms a huge discrete search space including # blocks, # message-passing layers, connections and embedding sizes.

Another insight facilitating the dealing with the second challenge is that if the message-passing flow is a super-structure of the ground-truth one, i.e., redundant computations are done, results in merely




(a) Message-Passing Flow Searching (b) Message-Passing Flow-based SR

Figure 3: The framework of our model. We partition the task into two stages: (a) Message-Passing Flow Searching and (b) Message-Passing Flow-based SR. At the first stage, full GN blocks with several layers (one layer shown) are pruned into a compact but effective, i.e. Pareto optimal, message-passing flow while at the second stage, we perform symbolic regression on every component, which is also a function, of the yielded message-passing flow. The final result is given by fusing SR results together based on the message-passing flow.

subtle variations of the loss, but if the message-passing flow is a sub-structure of the ground-truth, i.e., some necessary computations are missing, the loss jumps up with a change of magnitude.

Such an observation together with equation 3 facilitates us to first search an initial message-passing flow that is the super-structure of the ground-truth and then learn to prune the message-passing flows to get both compact and expressive message-passing flows. The framework of our model is shown in Figure 3, which sequentially search # blocks, # layers, connections and embedding sizes in a hierarchical way and the four steps in detail are listed as follows.

**Step 1: Search Message-Passing Blocks.** First, we need to find a message-passing flow that the ground-truth message-passing flow is its sub-structure. To achieve this goal, we first stack several full message-passing blocks. By optimizing the score in equation 3, we find the Pareto-optimal number of message-passing blocks, where we take the number of blocks as the complexity in equation 3 and RMSE between the predicted acceleration and the ground-truth as the loss.

**Step 2: Search Message-Passing Layers.** As mentioned, a full message-passing block contains three layers corresponding to updates of edge, node, and graph representation. However, not all of them are necessary for obtaining the output. To find the most compact layers, we try to delete each layer to see whether the score in equation 3 increases or decreases, where we define the complexity as the number of layers and RMSE as the loss. Our pruning-based searching method is based on the unique insight in SR, which is much more efficient than brute force search. Specifically, our method can significantly decrease the computational cost from $O(2^n)$ to $O(n)$, where $n$ is the number of initial layers.

**Step 3: Search Necessary Inputs.** We further filter out the useless inputs for each layer. Specifically, we adopt a similar strategy to the previous searching step: try to delete each input to see whether the score in equation 3 will rise or drop, where the complexity is the number of connections and the loss is RMSE. Similar to step 2, our model can significantly decrease the computational cost from $O(2^n)$ to $O(n)$, where $n$ is the number of initial inputs.

**Step 4: Search Embedding Sizes.** To ensure that the embedding in each layer are compact and with explicit physical meanings, we use the score given in equation 3 to find the Pareto-optimal embedding size for each embedding, where the complexity is defined as the embedding size and the loss is defined by RMSE. We try to reduce the embedding size to find the embedding size with the highest score, while we fix other embedding sizes as a large enough number to ensure the information bottleneck can only be caused by the embedding size we are searching.

### 2.4 The Learning Procedure

After obtaining the message-passing flow $\mathbf{M}^*$ and the parameters of DL component function $\mathbf{P}^*$ at the first stage, we follow Cranmer et al. (2020) to symbolize each DL component into formulas, and then cascade them according to the skeleton represented in $\mathbf{M}^*$ into a whole formula at the second stage. For aggregation functions $\rho$ corresponding to set functions, (i) we choose several commonly used aggregators as candidates, including *sum*, *mean* and *max*, while other aggregators can be generated by them, and select the maximum operator to symbolize it replacing the softmax function, (ii) we perform SR on input-output pairs from trained GNN component functions, (iii) we

fine-tune all constants in the overall function (given by cascading component functions), thereby avoiding accumulation of errors.

# 3 EVALUATION

## 3.1 EXPERIMENT SETUP

**Dataset.** We conduct experiments on both traditional physical datasets and real-world pedestrian datasets. In terms of traditional physical scenario, we utilize five datasets in different physical domains to demonstrate that our model has the ability to rediscover the well-known graph-structure physical mechanisms, as introduced in Section 2.1. We provide two cases of mechanics scenarios, one of electricity scenario, and two of thermology scenarios. For both mechanics and thermology scenarios, there are two selected cases with different complexity, where the content listed in parentheses are associated with the more complex scenarios. For the simple case in mechanics scenario The detailed information about formulas and data generation is reported in Appendix A.1. In terms of real-world pedestrian scenario, we conduct experiments on two real-world datasets including unidirectional and bidirectional flow in corridors, and the details are introduced in Appendix A.4.

**Metrics.** Given the same input, we use coefficient of determination $R^2$, indicating the proportion of the variation of the output that can be predicted from the input variables. Specifically, it is calculated by the output of distilled formula and the output of the ground-truth formula to measure whether the learned formula is accurate enough. $R^2$ can be calculated as $R^2 = 1 - \sum_i (y_i - \hat{y}_i)^2 / \sum_i (y_i - \bar{y})^2$, where $\bar{y} = \sum_i y_i / n$.

**Comparing Methods.** We compare our model with learning symbolic models from deep learning with inductive bias (SymDL) (Cranmer et al., 2020) to demonstrate that our model is flexible in more scenarios and a variant of our model that uses a full graph network without pruning searching (FullGN) for the ablation study (1-layer full GNN that removes non-exist inputs and non-required outputs). The message-passing flows of SymDL and FullGN are as shown in Appendix A.2.

**Parameter Settings.** We set learning rate as $10^{-4}$, tolerance in early stopping as $10$, $\#$layers and embedding size in MLP as $4$ and $16$, the max number of epoch as $20000$ and the weight $\lambda$ as $0.1$. The choice of parameter $\lambda$ is analysed in Appendix A.3.

**Implementation.** We implement our model in Python using Pytorch library and optimize all the models by Adam optimizer (Kingma & Ba, 2015). We use parallel symbolic regression in Python (PySR)[1] (Cranmer, 2020) to extract formulas from each message functions $\phi$. Our codes are in `https://anonymous.4open.science/r/GraphSR-654C`.

**Obtaining the Final Formula.** Due to the randomness of DL training and the genetic algorithm, we train a model ten times with different seeds to get ten formulas and select the Pareto-optimal formula according to the score (see Appendix B.4) as the final formula.

## 3.2 PLAUSIBILITY COMPARISON WITH BASELINES

Table 2: The applicability in different scenarios.

| Method | Mechanics (simple) | (complex) | Electricity | Thermology (simple) | (complex) |
|---|---|---|---|---|---|
| SymDL | √ | √ | × | × | × |
| Ours | √ | √ | √ | √ | √ |

Table 3: The running times (mins) for two stages in different scenarios.

| Scenario | | Mechanics (simple) | (complex) | Electricity | Thermology (simple) | (complex) |
|---|---|---|---|---|---|---|
| SymDL | | 69.91 | 70.82 | – | – | – |
| FullGN | | 86.25 | 113.59 | 59.23 | 67.21 | 69.23 |
| Ours | Stage 1 | 11.59 | 18.42 | 11.26 | 11.37 | 12.68 |
| | Stage 2 | 68.75 | 93.12 | 28.96 | 30.88 | 45.27 |
| | Total | 80.34 | 111.54 | 40.22 | 42.25 | 57.95 |

[1]https://github.com/MilesCranmer/PySR

| Method | Mechanics (simple) | | Mechanics (complex) | |
|---|---|---|---|---|
| | Learned formulas | R2 | Learned formulas | R2 |
| **SymDL (Cranmer, 2020)** | $a_i[0] = k_{ij}/m_i \sum_j ((C_1\Delta x_{ij} + C_2\Delta y_{ij}) + (C_3\Delta x_{ij} + C_4\Delta y_{ij})/r)$<br>$a_i[1] = k_{ij}/m_i \sum_j ((C_1\Delta x_{ij} + C_2\Delta y_{ij}) + (C_3\Delta x_{ij} + C_4\Delta y_{ij})/r)$ | 1.000 | $a_i[0] = \sum_j ((C_1 m_i + m_j)(\Delta x_{ij} + C_2\Delta y_{ij})/r + m_i m_j(C_3\Delta x_{ij} + \Delta y_{ij})) + C_4 F_{i,ext,x}/m_i$<br>$a_i[1] = \sum_j ((C_1 m_i + m_j)(\Delta x_{ij} + C_2\Delta y_{ij})/r * C_5 - m_i m_j(C_3\Delta x_{ij} + \Delta y_{ij})) + C_4 F_{i,ext,y}/m_i$ | 0.203 |
| **FullGN** | $a_i[0] = k_{ij}/m_i \sum_j ((C_1\Delta x_{ij} + C_2\Delta y_{ij}) + (C_3\Delta x_{ij} + C_4\Delta y_{ij})/r)$<br>$a_i[1] = k_{ij}/m_i \sum_j ((C_1\Delta x_{ij} + C_2\Delta y_{ij}) + (C_3\Delta x_{ij} + C_4\Delta y_{ij})/r)$ | 1.000 | $a_i[0] = \sum_j (C_1(\Delta x_{ij} + C_2\Delta y_{ij})/(m_i r) + (m_j\Delta x_{ij}/\Delta y_{ij}) + C_4 F_{i,ext,x}/m_i$<br>$a_i[1] = \sum_j (C_1(\Delta x_{ij} + C_2\Delta y_{ij})/(m_i r) + (\Delta x_{ij} + C_3\Delta y_{ij})/r) + C_4 F_{i,ext,y}/m_i$ | 0.529 |
| **Ours** | $a_i[0] = k_{ij}/m_i \sum_j ((C_1\Delta x_{ij} + C_2\Delta y_{ij}) + (C_3\Delta x_{ij} + C_4\Delta y_{ij})/r)$<br>$a_i[1] = k_{ij}/m_i \sum_j ((C_1\Delta x_{ij} + C_2\Delta y_{ij}) + (C_3\Delta x_{ij} + C_4\Delta y_{ij})/r)$ | 1.000 | $a_i[0] = k_{ij}/m_i \sum_j ((C_1\Delta x_{ij} + C_2\Delta y_{ij}) + (C_3\Delta x_{ij} + C_4\Delta y_{ij})/r) + C_5 f_{i,ext,x}/m_i + C_6 \sum_i f_{i,ext,x} / \sum_i m_i$<br>$a_i[1] = k_{ij}/m_i \sum_j ((C_1\Delta x_{ij} + C_2\Delta y_{ij}) + (C_3\Delta x_{ij} + C_4\Delta y_{ij})/r) + C_5 f_{i,ext,y}/m_i + C_6 \sum_i f_{i,ext,y} / \sum_i m_i$ | 0.917 |

| Method | Electricity | | Thermology (simple) | | Thermology (complex) | |
|---|---|---|---|---|---|---|
| | Learned formulas | R2 | Learned formulas | R2 | Learned formulas | R2 |
| **FullGN** | $P_o = C_1 \sum_{i,j} (V_i - V_j)(V_i/R + C_2) + \sum_i \sum_j (V_i V_j + C_2 V_i^2 + C_3)/R$ | 0.587 | $\Delta S = C_1 \sum_{i,j} C_2 (T_j * T_i) + C_3 T_i + \sum_i \sum_j \alpha(T_i - C_4 T_i)$ | 0.318 | $\Delta S = C_1 \sum_{ij} C_3 \alpha (T_j/T_i + C_4 T_i * \Delta Q_{i,ext}) + C_6 \Delta Q_{i,ext}/T_i$ | 0.189 |
| **Ours** | $P_o = C_1 \sum_{i,j} (C_2 V_i - C_3 V_j)^2/R$ | 1.000 | $\Delta S = C_1 \sum_{i,j} C_2 \alpha (T_j - T_i)/T_i$ | 1.000 | $\Delta S = C_1 \sum_i \left( C_2 \sum_j C_3 \alpha (T_j - T_i) + C_4 \Delta Q_{i,ext} \right)/T_i$ | 0.993 |

Figure 4: Correctness comparison based on learned formulas and $R^2$ in five scenarios. The formulas learned by our model are the same as the ground-truth formulas

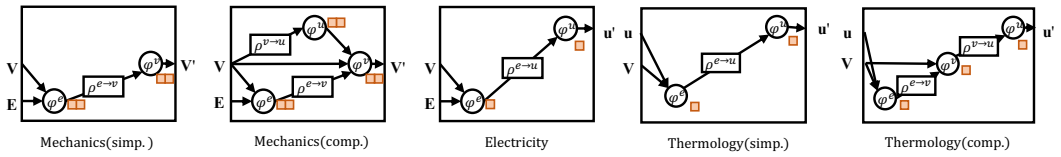



Figure 5: Learned message-passing flows by our model.

We first compare the applicability of our method with the SOTA baseline (Cranmer et al., 2020). As listed in Table 2, our method can be applied to all five cases from three physical scenarios including mechanics, electricity and thermology, while the baseline fails in the last two scenarios due to incorrect message-passing flow, because their designed message-passing flow is specifically designed for Newton-force interaction in simple mechanical scenario and not flexible enough for other scenarios.

We design two cases in mechanics scenarios: calculating the acceleration and the relative acceleration in the center-of-mass frame. SymDL is designed for handling formula discovery at the simple case, with a specified message-passing flow. Comparatively, our method moves forward to a more general and challenging setting without specifying a message-passing flow representing formula skeleton. To ascertain the correctness of the learned formulas, besides the baseline, we further introduced a variant of our method without searching the message-passing flow, i.e., directly using the full message-passing block in the first stage. As listed in Figure 4, in the simple mechanics case, our model achieved the same performance with two baselines. In the complex case, our model outperformed the SOTA baseline and the variant of our model by a large margin w.r.t. $R^2$ metric. Specifically, these two competitors both failed and got a rather low $R^2$, while our method managed to rediscover the correct formula with $R^2 = 0.917$, indicating the advantage of searching correct messaging-passing flow. For our model, the difference between two formulas for two cases is about the latter two terms corresponding to the additional message-passing flows $\mathbf{V} \rightarrow \mathbf{V}'$ and $\mathbf{V} \rightarrow \mathbf{u}' \rightarrow \mathbf{V}'$, which cannot by handled by SymDL. The formulas learned by baselines are wrong for lack of necessary dependencies, which fail to have physical meaning and differ largely from the ground truth.

Our problem is different from SymDL, requiring prior knowledge of the formula skeleton for designing the deep learning architecture, which is almost impossible to know in new real-world scenarios. For the rest three cases that the SOTA baseline cannot handle, as listed in Figure 4, the performance gain over the variant using full message-passing flow indicates that optimizing Pareto-optimal score is essential in obtaining correct formulas, which is less subject to redundant

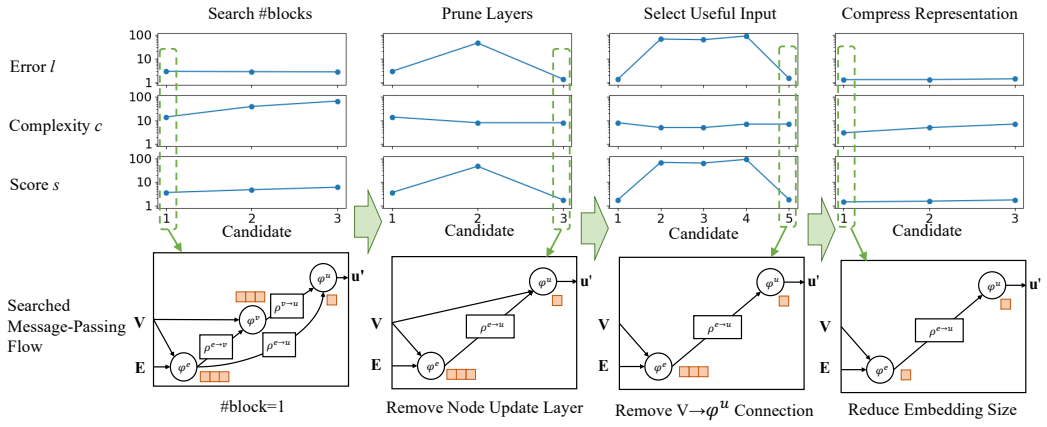



Figure 6: The searching process of message-passing flows in four steps in the circuit scenario.

message-passing flows that hinder the subsequent SR process, including unnecessary inputs and redundant computation steps. The detailed searching process of the electricity case is analyzed in Section 3.3. For the complex case of thermology, it can be observed that the learned formula successfully captures the effect of external conducted heat compared with the simple case, while other baselines fail to have physical meanings due to unnecessary inputs and redundant computation steps. Although the change is slight (whether exist external heat conduction), both the skeleton and the entire formula are quite different and so is the entire formula.

To be more clear, the learned message-passing flows by our model are as shown in Figure 5. Besides, the time cost of each independent part is shown in Table 6, where we can observe that searching message-passing flows only takes a small part of the whole procedure, and our model's running times are similar to SymDL and usually shorter than FullGN. Furthermore, the learning curves are reported in Appendix A.5.

### 3.3 Qualitative Results for Understanding the Searching Process

We show the searching process of message-passing flow in Figure 6, where the upper row shows the learning curve in terms of error (RMSE), complexity and a score, which is a weighted summation of error and complexity. From Figure 6, we observe that if the message-passing flow is a sub-structure of the ground-truth message-passing flow, the performance will drop significantly, and on the other hand, message-passing flows with redundant layers/inputs/embedding sizes will have a similar performance, echoing the rationale of our pruning strategy. The core idea is to search the most compact message-passing that are expressive enough and the four-step searching process is as follows, (i) the model tried the number of blocks as $1 \sim 3$, finding similar errors with a rise in complexity, so it opted for 1; (ii) from the searched message-passing flow at the previous stage, it tried to delete every layer associated with $\phi$. It turned out that deleting the edge layer would cause a huge error increase, so the edge layer was preserved, after which it tried to delete node layer and found that the score decreased (error does not change a lot and the complexity decrease), so it decided to delete node layer; (iii) similar to the previous stage, it tried to delete each input, and found that only deleting the $V \rightarrow \phi^u$ connection would not cause error increase, so this connection was deleted. (iv) finally, it tried to compress each representation and found that when the embedding size was 1, the score minimized, so 1 was chosen as the embedding size. After the whole process, the message-passing flows including embedding (intermediate variable), functions and topology have explicit physical meanings, paving the way for symbolic regression.

## 4 Related Works

**Symbolic Regression (SR).** Distilling elegant symbolic expressions from vast experimental data has always been the mainstream method used for findings of new formulas and the verification of hypotheses throughout the history of physics. SR is a classic topic (Schmidt & Lipson, 2009; Petersen et al., 2020; Biggio et al., 2021; Guimerà et al., 2020) that tries to emulate the process, to learn an explicit symbolic model that can describe a the projection from input $X$ to the output $y$ as accurately

as possible while maintaining its compactness. Traditional methods of discovering formulas from data are mostly based on genetic programming(GP) (Schmidt & Lipson, 2009; Koza, 1994; Worm & Chiu, 2013). The main idea of GP-based SR methods lies in the "evolution" of symbolic expressions through processes including selection, crossover, and mutation. From generation to generation, the expressions become increasingly better at fitting the data given. Hitherto, there have been promising results yielded by GP-based SR methods such as the famous commercial SR method Eureqa (Dubčáková, 2011), Burlacu et al. (2020) and Virgolin et al. (2019), etc. More recently, with the development of DL techniques, DL methods for symbolic regression are introduced (Zheng et al., 2021; Qian et al., 2021) with better expressive ability than GP. Martius & Lampert (2016) proposed a model named EQL and extracted the symbolic formulas with a neural network with symbolic models as building blocks. Kusner et al. (2017) managed to eschew the problem of discrete optimization by converting discrete data into a parse tree. AI Feynman (Udrescu & Tegmark, 2020; Udrescu et al., 2020) implement the philosophy of divide-and-conquer and exploits the modularity of the function by splitting it into sub-functions and perform regression on each module separately. The partition of functions are achieved with a trained neural network. SymDL (Cranmer et al., 2020) exploited the inductive biases of the correspondence of physical mechanisms (kinematics, especially) to GNN structure and established the method PySR to tackle the problem. They first revealed the link between GNNs and physical mechanisms. Different from these works, our model can learn graph-structured physical mechanism without requiring the information of formula skeletons, which can be hardly obtained in new physical scenarios for discovery.

**Graph Neural Network (GNN).** GNN (Kipf & Welling, 2017; Veličković et al., 2018; Gilmer et al., 2017) is powerful for learning representations for graph-structured data. GNN has been widely used for physical systems by capturing the interacting mechanisms, such as simulating mechanical system (Sanchez-Gonzalez et al., 2020; Huang et al., 2021; Sanchez-Gonzalez et al., 2018), designing circuits (Zhang et al., 2019; Ren et al., 2020), simulating heat conduction (Chamberlain et al., 2021; Xhonneux et al., 2020). GNNs can be viewed in a message-passing manner (Battaglia et al., 2018; Veličković, 2022; Bronstein et al., 2017), and most of them can be summarized as message-passing among three levels: edge/node/graph level, while the message-passing flows and message/aggregation functions can be customized very differently based on the specific characteristics of applications. Furthermore, there are some works (You et al., 2020; Yoon et al., 2020; Cai et al., 2021; Gu et al., 2021) that propose search space and search algorithms for the architecture of message-passing network for a specific task. From the perspective of graph-structured formulas, the message-passing flow corresponds to the computational steps with different level variables as inputs and output. Different from them, inspired by symbolic regression (Udrescu & Tegmark, 2020), we propose to search the Pareto-optimal message-passing flows, which is both accurate and simple, and can benefit learning of symbolic models.

**Pareto-optimal Search.** The previous pareto-optimal solutions proposed in NAS area (Lomurno et al., 2021; Lu et al., 2020; Dong et al., 2018) focus on finding the model architecture with both high prediction accuracy and low inference latency, which does not meet requirements for solving graph SR problem. Instead, our proposed method is based on a novel insight in SR scenario: the performance would be similar when the message-passing flow (skeleton) is a super-structure of the ground-truth one, while the performance degrades a lot if it is a sub-structure of the ground-truth one.

## 5 Conclusion

In this paper, we generalize the problem in Cranmer et al. (2020) by learning the formula skeleton rather than manually designing, which is crucial for learning formulas in a new physical area without much prior knowledge. We propose a new SR method that first transforms the discovery of formula skeleton to the search of the Pareto-optimal message-passing flow with accuracy and compactness, and then symbolizes its message functions to obtain underneath formula. We conduct experiments on five datasets from three different physical domains, including mechanics, electricity and thermology, and demonstrate that our method can consistently learn plausible formulas describing the graph-structured physical mechanism. Furthermore, to show that our model is practical for learning unknown formulas in real world, we conduct experiments on two real-world datasets about pedestrian dynamics (see Appendix A.4), which learns different formulas with explicit physical meanings for different scenarios more precise than mainstream empirical formulas.

## REFERENCES


Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. *arXiv preprint arXiv:2106.06427*, 2021.

Maik Boltes and Armin Seyfried. Collecting pedestrian trajectories. *Neurocomputing*, 100:127–133, 2013.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Bogdan Burlacu, Gabriel Kronberger, and Michael Kommenda. *Operon C++: An Efficient Genetic Programming Framework for Symbolic Regression*, pp. 1562–1570. Association for Computing Machinery, New York, NY, USA, 2020. ISBN 9781450371278. URL `https://doi.org/10.1145/3377929.3398099`.

Shaofei Cai, Liang Li, Jincan Deng, Beichen Zhang, Zheng-Jun Zha, Li Su, and Qingming Huang. Rethinking graph neural architecture search from message-passing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6657–6666, 2021.

Ben Chamberlain, James Rowbottom, Maria I Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. Grand: Graph neural diffusion. In *International Conference on Machine Learning*, pp. 1407–1418. PMLR, 2021.

Miles Cranmer. Pysr: Fast & parallelized symbolic regression in python/julia, September 2020. URL `http://doi.org/10.5281/zenodo.4041459`.

Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in Neural Information Processing Systems*, 33, 2020.

Jin-Dong Dong, An-Chieh Cheng, Da-Cheng Juan, Wei Wei, and Min Sun. Dpp-net: Device-aware progressive search for pareto-optimal neural architectures. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 517–531, 2018.

Renáta Dubčáková. Eureqa: software review, 2011.

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner Gardner, Marc Parizeau, and Christian Gagné. Deap: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, 13(1):2171–2175, 2012.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Yu-Chao Gu, Li-Juan Wang, Yun Liu, Yi Yang, Yu-Huan Wu, Shao-Ping Lu, and Ming-Ming Cheng. Dots: Decoupling operation and topology in differentiable architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12311–12320, 2021.

Roger Guimerà, Ignasi Reichardt, Antoni Aguilar-Mogas, Francesco A Massucci, Manuel Miranda, Jordi Pallarès, and Marta Sales-Pardo. A bayesian machine scientist to aid in the solution of challenging scientific problems. *Science advances*, 6(5):eaav6971, 2020.

Dirk Helbing and Peter Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51 (5):4282, 1995.

Wenbing Huang, Jiaqi Han, Yu Rong, Tingyang Xu, Fuchun Sun, and Junzhou Huang. Equivariant graph mechanics networks with constraints. In *International Conference on Learning Representations*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

Matt J. Kusner, Brooks Paige, and José Miguel Hernández-Lobato. Grammar variational autoencoder. In Doina Precup and Yee Whye Teh (eds.), *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1945–1954. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/kusner17a.html.

JW Leech. The feynman lectures on physics. *Physics Bulletin*, 17(10):367, 1966.

Pablo Lemos, Niall Jeffrey, Miles Cranmer, Shirley Ho, and Peter Battaglia. Rediscovering orbital mechanics with machine learning. *arXiv preprint arXiv:2202.02306*, 2022.

Li Li, Minjie Fan, Rishabh Singh, and Patrick Riley. Neural-guided symbolic regression with asymptotic constraints. *arXiv preprint arXiv:1901.07714*, 2019.

Eugenio Lomurno, Stefano Samele, Matteo Matteucci, and Danilo Ardagna. Pareto-optimal progressive neural architecture search. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 1726–1734, 2021.

Zhichao Lu, Ian Whalen, Yashesh Dhebar, Kalyanmoy Deb, Erik D Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Multiobjective evolutionary design of deep convolutional neural networks for image classification. *IEEE Transactions on Evolutionary Computation*, 25(2):277–291, 2020.

Georg Martius and Christoph H. Lampert. Extrapolation and learning equations, 2016. URL https://arxiv.org/abs/1610.02995.

Brenden K Petersen, Mikel Landajuela Larma, Terrell N Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2020.

Zhaozhi Qian, Krzysztof Kacprzyk, and Mihaela van der Schaar. D-code: Discovering closed-form odes from observed trajectories. In *International Conference on Learning Representations*, 2021.

Haoxing Ren, George F Kokai, Walker J Turner, and Ting-Sheng Ku. Paragraph: Layout parasitics and device parameter prediction using graph neural networks. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6. IEEE, 2020.

Subham Sahoo, Christoph Lampert, and Georg Martius. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, pp. 4442–4450. PMLR, 2018.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pp. 4470–4479. PMLR, 2018.

Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter Battaglia. Learning to simulate complex physics with graph networks. In *International Conference on Machine Learning*, pp. 8459–8468. PMLR, 2020.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *science*, 324(5923):81–85, 2009.

Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. *Advances in Neural Information Processing Systems*, 33, 2020.

Petar Veličković. Message passing all the way up. *arXiv preprint arXiv:2202.11097*, 2022.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *International Conference on Learning Representations*, 2018.

Marco Virgolin, Tanja Alderliesten, and Peter A. N. Bosman. Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pp. 1084–1092, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361118. doi: 10.1145/3321707.3321758. URL `https://doi.org/10.1145/3321707.3321758`.

Tony Worm and Kenneth Chiu. Prioritized grammar enumeration: symbolic regression by dynamic programming. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 1021–1028, 2013.

Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. Continuous graph neural networks. In *International Conference on Machine Learning*, pp. 10432–10441. PMLR, 2020.

Minji Yoon, Théophile Gervet, Bryan Hooi, and Christos Faloutsos. Autonomous graph mining algorithm search with best speed/accuracy trade-off. In *2020 IEEE International Conference on Data Mining (ICDM)*, pp. 751–760. IEEE, 2020.

Jiaxuan You, Zhitao Ying, and Jure Leskovec. Design space for graph neural networks. *Advances in Neural Information Processing Systems*, 33:17009–17021, 2020.

Guo Zhang, Hao He, and Dina Katabi. Circuit-gnn: Graph neural networks for distributed circuit design. In *International conference on machine learning*, pp. 7364–7373. PMLR, 2019.

Wenqing Zheng, Tianlong Chen, Ting-Kuei Hu, and Zhangyang Wang. Symbolic learning to optimize: Towards interpretability and scalability. In *International Conference on Learning Representations*, 2021.

# A EXPERIMENTS

## A.1 DATA GENERATION

In different scenarios, we devised different inputs and computed the theoretical outcome according to the known mechanisms.

In the scenario of *Mechanics*, we randomly (standard normal distribution) set the $(x, y)$ coordinates of the particles in 2-D cases. We assign the masses of the particles randomly according to the lognormal distribution. The original lengths of the springs are all set to 1. In the complex case, external forces with all dimensions following the standard normal distribution are exerted. Graph topology is picked randomly. We then compute the accelerations of each particles with Hooke's Law, the independent action principle of force and Newton's Second Law of Motion.

In the scenario of *Electricity*, we randomly give a topology on the graph and set the electric potential of nodes following the standard normal distribution. Resistances of resistors on edges are chosen uniformly randomly from 0.01 to 1.01 to avoid extremely large power outputs. We then compute the power of each edges (resistors) according to Joule's Law and add them up to reach the overall power of the resistor circuit.

In the scenario of *Thermology*, the graph topology is given as a 'grid', echoing the core idea of Finite Element Analysis. We set the temperature of each node randomly between 0 and 1, the thermal conductivity randomly from 1 to 3 globally. We then compute the discrete laplacian on the grid and the heat flow according to Fourier's Law of Heat Conduction. With the heat flow and the temperature of each nodes, we compute their entropy production rate separately and add them up to reach the overall entropy production rate.

The basic information of our used datasets are listed in Table 4.

Table 4: Basic Information about Datasets.

| Domain | Mechanics | Electricity | Thermology |
|---|---|---|---|
| Scenario | Spring-connected particles | Resistor circuit | Thermal system |
| Topology | Unoriented graph | Unoriented graph | Grid |
| Node Input | Mass, position, (force) | Electric potential | Temperature, (external heat) |
| Edge Input | Spring constant | Resistance | Thermal conductivity |
| Global Input | None | None | None |
| Output | Acceleration (relative to mass center) | Overall power | Overall entropy production |
| # Graphs | 100 | 100 | 50 |
| # Nodes | 5 | 5 | 9 |
| # X-Y pairs | 100000 | 100000 | 100000 |

## A.2 BASELINE DETAILS

The message-passing flows of baselines, SymDL and FullGN, are shown in Figure 7.

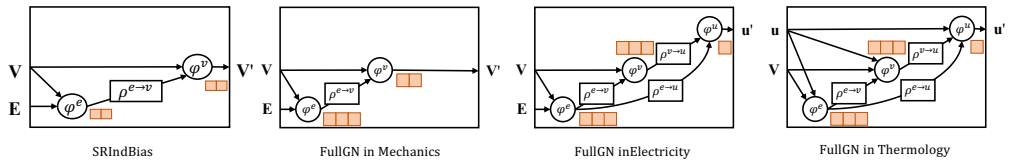



Figure 7: message-passing flows of SymDL and FullGN.

## A.3 PARAMETER SENSITIVITY

One of the most critical parameters in our model is the weight $\lambda$ that balances the complexity and errors. We normalize the input-output pairs to make the outputs have variance 1, and then the errors

are in similar magnitudes, and besides, the complexities of different skeletons are also with similar magnitudes, so the best value of $\lambda$ is similar for different datasets. Here, we test different values of $\lambda$ on three diverse scenarios, including the circuit scenario and two mechanical scenarios. For each value of $\lambda$, we use ten different seeds to train the model to test whether our model can learn the correct message-passing flows and the correct formulas. We choose the best formula among ten formulas, so all of these values of $\lambda$ can allow us to find the correct formula in different scenarios. Among them, we choose $\lambda = 0.1$ for achieving the highest success rate.

Table 5: Success Rate in the Circuit Scenarios.

| Value of $\lambda$ | | 0.025 | 0.05 | 0.1 | 0.2 | 0.4 |
|---|---|---|---|---|---|---|
| Circuit | learning message-passing flows | 3/10 | 9/10 | 10/10 | 10/10 | 8/10 |
| | learning formulas | 3/10 | 9/10 | 10/10 | 10/10 | 8/10 |
| Mechanics (Simple) | learning message-passing flows | 2/10 | 8/10 | 9/10 | 6/10 | 1/10 |
| | learning formulas | 2/10 | 7/10 | 8/10 | 6/10 | 1/10 |
| Mechanics (Complex) | learning message-passing flows | 1/10 | 6/10 | 7/10 | 5/10 | 2/10 |
| | learning formulas | 1/10 | 5/10 | 6/10 | 5/10 | 1/10 |

### A.4 Experiments on Formula Discovery for Real-world Pedestrian Dynamics

To better show how our model discovers unknown graph-structured physical mechanisms in the real world, we conduct carefully-designed experiments of our model on formula discovery for pedestrian dynamics. We first formulate this problem. We aim to find a formula that can approximately describe the relationship between acceleration $a$ and the velocity $v$, pedestrian position $x$ and destination position $x_{dest}$. The graph $\mathcal{G}$ describe the interaction relationship, which is constructed as follows, when the two pedestrians' distance is less than $R$, they are connected, and otherwise, they are unconnected. Formally, the problem can be described as finding a formula $\mathcal{F}$ that fits $a = \mathcal{F}(\mathcal{G}, x, v, x_{dest})$.

For pedestrian dynamics, a well-known manually-designed model is social force model Helbing & Molnar (1995), in which the pedestrian dynamics are driven by two factors: (a) a pedestrian is attracted by his/her destination with a force $\mathbf{F}_{Di} = (v_{di}\mathbf{e}_i - \mathbf{v}_i)/\tau$, $\mathbf{e}_i = \frac{\mathbf{x}_d - \mathbf{x}_i}{\|\mathbf{x}_d - \mathbf{x}_i\|}$, where $v_{di}$ is the value of desired velocity, $\mathbf{v}_i$ is the current velocity, $\tau$ is the relaxation time, $\mathbf{e}_i$ is the unit vector toward the destination; (b) a pedestrian is repulsed by the nearby ones with a force $\mathbf{F}_{ij} = A_i \exp(-r_{ij}/B_i)\,\mathbf{e}_{ij}^n$, where $\mathbf{F}_{ij}$ is the repulsive force, $r_{ij}$ is the distance between pedestrian $i$ and $j$. The join force is $\mathbf{F}_i = \mathbf{F}_{Di} + \sum_{j \in \mathcal{N}_i} \mathbf{F}_{ij}$, where $\mathcal{N}_i$ means the set of pedestrians whose distance to pedestrian $i$ is less than 5 meters. Social force model is widely used as the foundation of many commercial softwares such as viswalk[2] and anylogic[3]. In this paper, we assume that mass of pedestrian as 1, and thus $a_i = F_i/m = F_i$.

However, on one hand, social force model is manually designed which may have discrepancies with the real-world pedestrian dynamics. On the other hand, usually, different scenarios are with very different pedestrian interaction mechanisms, which cannot be precisely modeled by one single model. So it is meaningful to learn data-driven formulas to describe different pedestrian interaction mechanisms.

We conduct experiments on two real-world datasets of crowd trajectories: several experimental datasets from studies about pedestrian dynamics Boltes & Seyfried (2013)[4], including the following scenarios, (i) **Unidirectional flow in corridor**: a group of people go through a corridor in the same direction, as shown in Figure 8(a); (ii) **Bidirectional flow in corridor**: a group of people go through a corridor in the opposite directions, as shown in Figure 8(b). The learned formulas and the corresponding physical meanings are reported in Figure 9, which demonstrate that our model can learn different skeletons and the formulas that are more precise than social force model with explicit

[2] https://www.myptv.com/en/mobility-software/pedestrian-simulation-software-ptv-viswalk
[3] https://www.anylogic.com/features/libraries/pedestrian-library/
[4] https://ped.fz-juelich.de/database/doku.php

physical meanings. The performance comparison are reported in Table 6, where we can observe that our model has about 10% improvement compared to social force model.

(a) Corridor, Unidirectional Flow. (b) Corridor, Bidirectional Flow.

Figure 8: Representative snapshot of two pedestrian datasets.

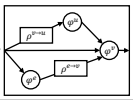
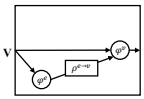
| Scenario | Unidirectional flow in corridor | Bidirectional flow in corridor |
|---|---|---|
| **Searched Message-passing Flow** |  |  |
| **Searched Skeleton** | $\varphi^v\left(x_i, \sum_{j\in N(i)} \varphi^e(x_i,x_j), \sum_{j\in V} \varphi^u(x_j)\right)$ | $\varphi^v\left(x_i, \sum_{j\in N(i)} \varphi^e(x_i,x_j)\right)$ |
| **Learned Formula (Difference with Social Force)** | $\varphi^e$: $ke^{-\lambda r}\,\mathrm{sign}(v_n)$ (normal component)<br>$\varphi^e$: $\delta$ (tangential component)<br>$\varphi^u$: $v_{mean} = \mathrm{mean}(v_i)$<br>$\varphi^v$: $\sum_{j\in N(i)} \varphi^e + (\mathrm{v} - \lambda_1 \boldsymbol{v}_0 - \lambda_2 \varphi^u)/\tau$ | $\varphi^e$: $\mathrm{k}_1\, e^{-\lambda r}\mathrm{sign}(v_n)$ (normal component)<br>$\varphi^e$: $\delta$ (tangential component)<br>$\varphi^v$: $\sum_{j\in N(i)} \varphi^e(x_i,x_j) + (\mathbf{v} - \lambda \boldsymbol{v}_0)/\tau$ |
| **Physical Meanings** | **1. There is no repulsive force when the front is faster than the back**<br>**2. Tend to dodge to the right**<br>**3. Herding: Tend to follow the average speed of the crowd** | **1. There is no repulsive force when the front is faster than the back**<br>**2. Tend to dodge to the right** |

Figure 9: Learned formulas for pedestrian dynamics in two scenarios.

Table 6: R2 of social force model and our learned formulas for two pedestrian scenarios.

| Scenario | Unidirectional Flow in Corridor | Bidirectional Flow in Corridor |
|---|---|---|
| Social Force | 0.71 | 0.68 |
| Ours | 0.78 | 0.81 |

### A.5 Learning Curves

We first plot the learning curves of three cases w.r.t. different message-passing flows in Figure 10, corresponding to the one lacking the necessary message-passing connection, the ground-truth one, and the one with a redundant message-passing connection, respectively. We find that the ground-truth message-passing flow and the one with redundant message-passing connection have only little variations in performances after the loss function converges. However, the performance of message-passing flow lacking necessary connection decreases significantly.

We further test the impact of the embedding size on the performance, where the learning curves are shown in Figure 11. From this, we can see that when the embedding size is less than a certain number 2, the performance decreases significantly, while the performance is similar when the embedding size is not less than 2.

Furthermore, we test whether the softmax function can learn the ground-truth aggregator. As we can see in Figure 12, the learning curve with softmax and with the ground-truth aggregator is quite similar, suggesting that the softmax function is able to learn the choice of the aggregator. Furthermore, we verify that the learned aggregator is a “summation”, which is the same as the ground truth.

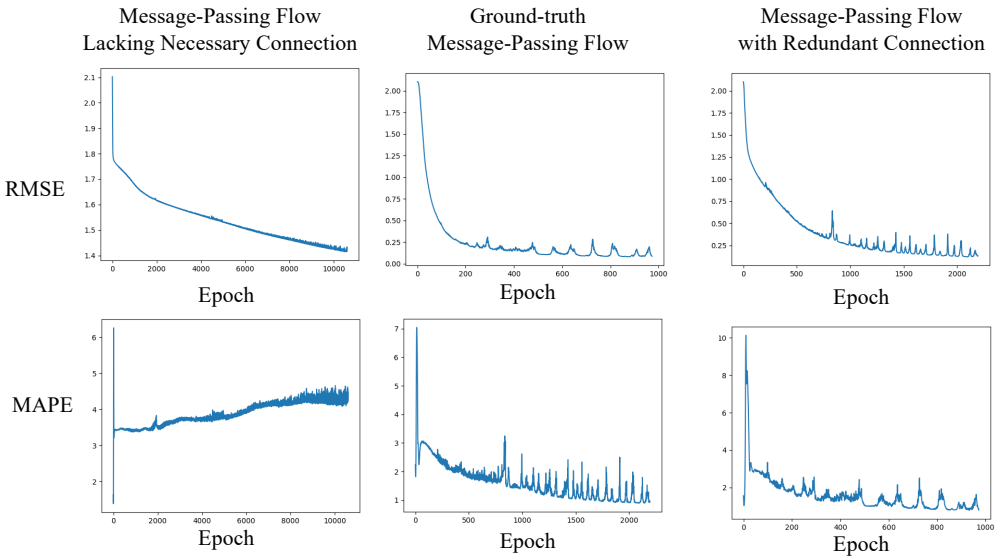



Figure 10: Learning curves w.r.t. message-passing flows.

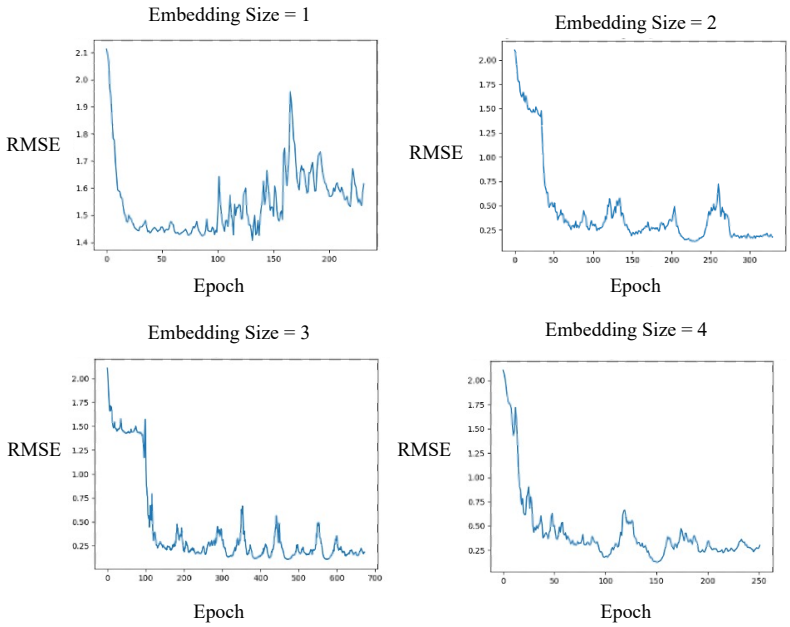



Figure 11: Learning curves w.r.t. embedding sizes.

# B METHOD DETAILS

## B.1 DETAILS OF SYMBOLIZING THE AGGREGATION/MESSAGE FUNCTIONS

**Symbolize Aggregation Function $\rho$.** We choose several commonly used aggregators as candidates, including *sum*, *mean* and *max*, while other candidate aggregators such as *min* can be achieved by

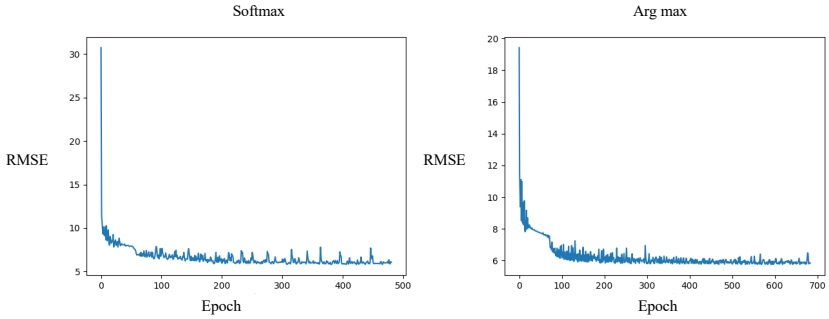



Figure 12: Learning curves with or without softmax.

$min(x) = -max(-x)$, and root mean square can be represented by $(mean(x^2))^{\frac{1}{2}}$. Harmonic mean and l-norm can also be decomposed into two $\phi$ functions and a *mean* aggregators. These operations can be learned by $\phi$.

**Symbolize Message Function $\phi$.** After training the GNN model, we use the tool of symbolic regression to extract a symbolic model with correspondence to message and update functions. Specifically, for message functions in neural network, we recorded their input-output pairs and implement classic symbolic tools to symbolize them.

**Retrain and Fine-tune Constants.** Finally, to get rid of accumulated errors, we set all constants in the entire formulas as parameters to be trained and fine-tune them to get the final graph-structured mechanism represented by symbolic models. Specifically, after extracting the message-passing flows, we replace the MLP in message/update functions by the corresponding formulas. And we set the constants in the formulas as parameters in the deep model to optimize (all the operations in symbolic models are differentiable).

### B.2 Details of Errors

In this paper, we use *RMSE* as the measure of error. *RMSE* can be calculated as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}},$$

where $\hat{y}_i$ is the $i$-th predicted value and $y_i$ is the $i$-th ground-truth value and $i = 1, \cdots, n$.

### B.3 Details of Complexity

The design of complexity measurement for message-passing flows is flexible. In this paper, we calculate the complexity as follows, (i) for each layer, the complexity can be calculated as the product of the embedding size of this layer and the number of inputs in this layers; (ii) the whole complexity of the message-passing flow can be calculated as the summation of the complexity of each layer. To illustrate, the complexity of four message-passing flows in the Figure 6 can be calculated as $2 \times 3 + 2 \times 3 + 2 \times 1 = 14$, $2 \times 3 + 2 \times 1 = 8$, $2 \times 3 + 1 \times 1 = 7$ and $2 \times 1 + 1 \times 1 = 3$. As we can see, the complexity consistently decreases in the searching process.

### B.4 Details of Score

The score of a graph-structured formula is $s = l + \lambda c$, where $l$ is RMSE, $c$ is the complexity of the graph-structured formulas, which is defined as the complexity of message-passing flow multiplying the average complexity of component formulas.