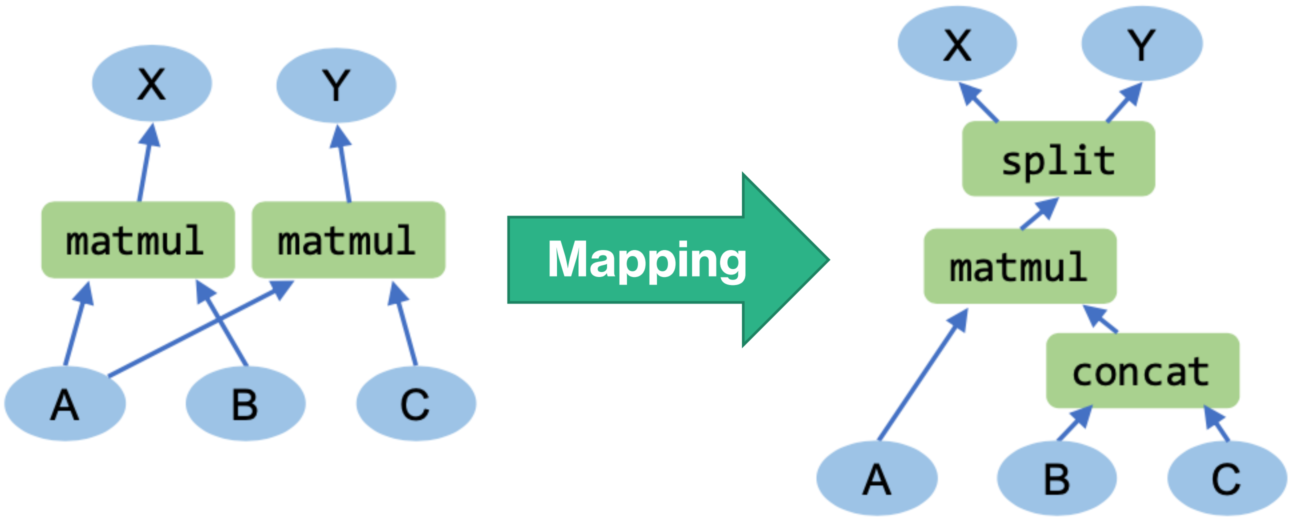
TASO: Optimizing Deep Learning Computation with Automatic Generation of Graph Substitutions

Zhihao Jia, Oded Padon, James Thomas, Todd Warszawski, Matei Zaharia, Alex Aiken

Presented by: Samuil Stoychev

Graph Substitutions

- DNNs are represented as **computation graphs** combining multiple **operators**.
- A **substitution** is a **mapping** from a **source graph** to a **target graph**.
- We want a target graph that is **functionally equivalent** and that has **better performance**.



Source graph

Target graph

Existing Solutions

- All use a greedy **rule-based** approach.
- Substitution heuristics manually designed by **human experts**.

Existing Approach

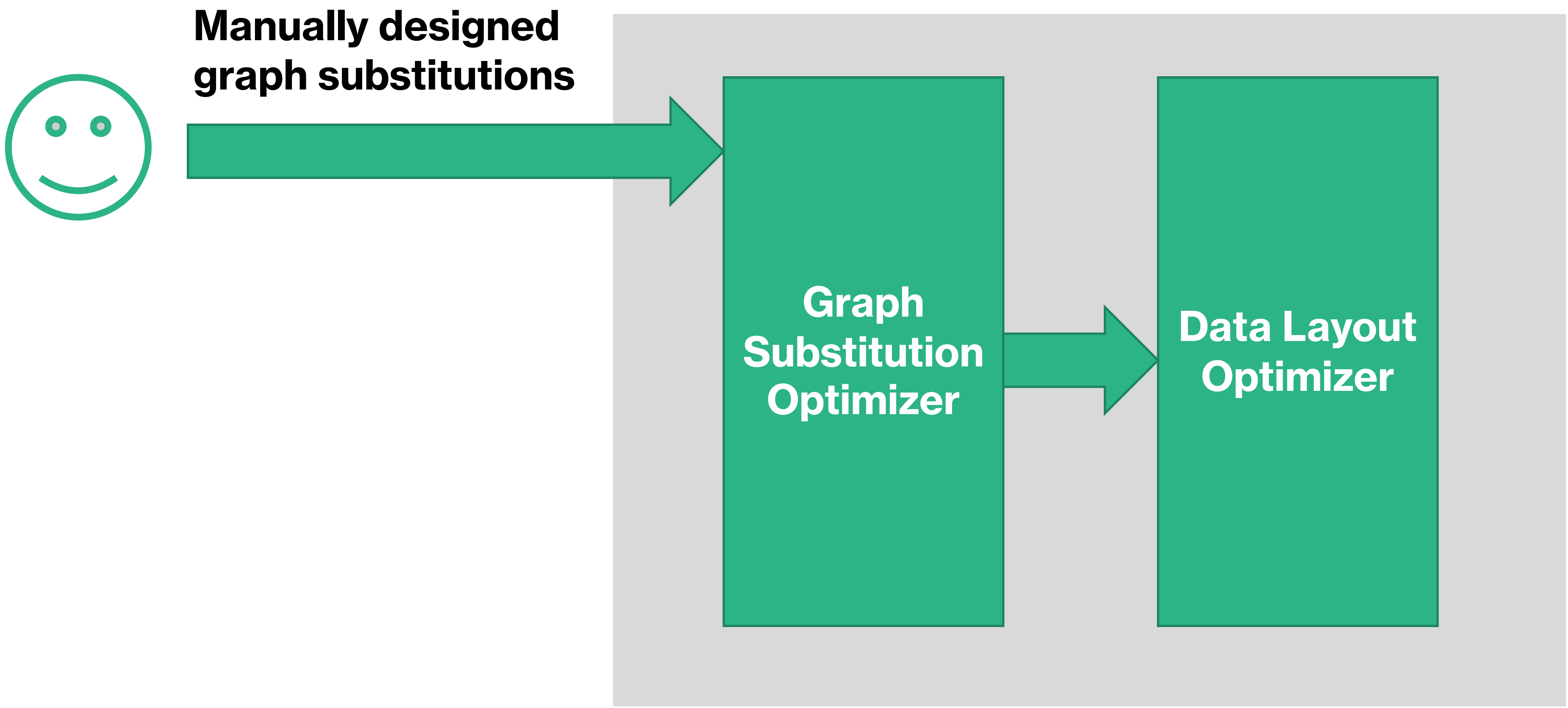


*Diagram adapted from Z. Jia et al., 2019

Existing Approach

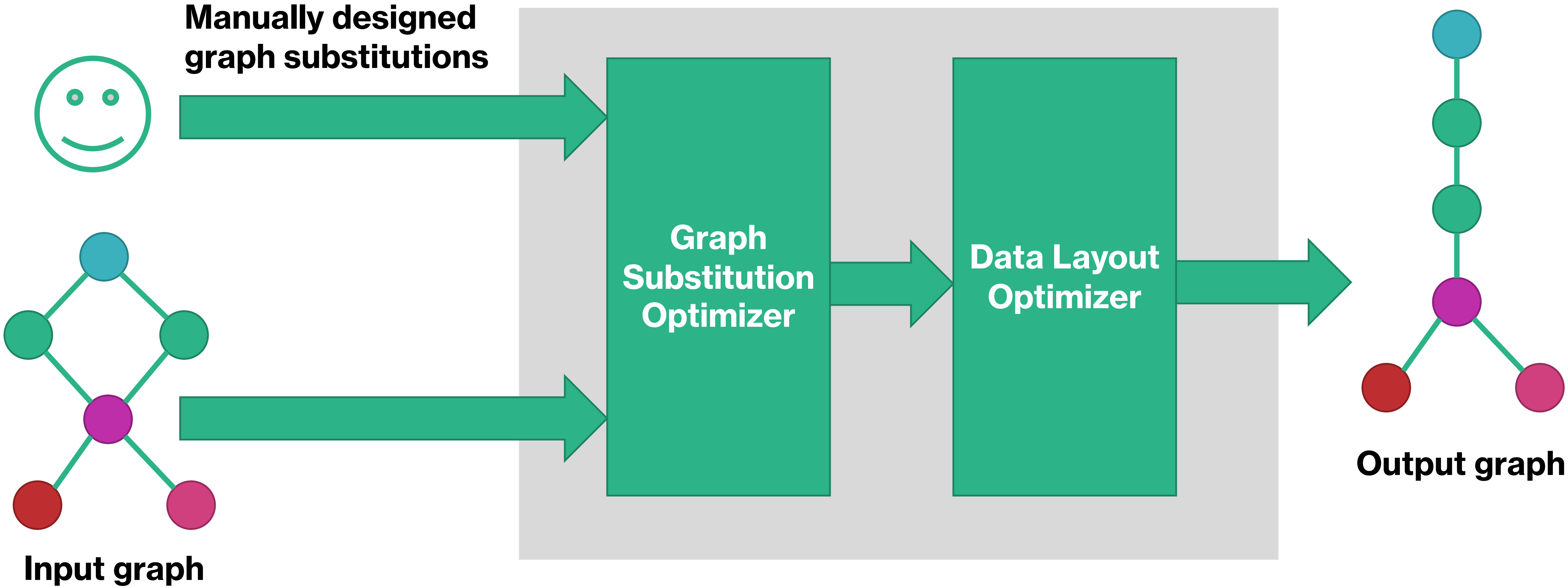


*Diagram adapted from Z. Jia et al., 2019

Problems With Current Solutions

- **Maintainability**
  - Manually writing substitutions is time-consuming.
  - TensorFlow's 155 substitutions implemented in ~53K lines of code.
  - Hard to add new operators.
- **Data layout**
  - Graph substitutions and data layout are interconnected.
  - However, current approaches treat them as isolated optimization problems.
- **Correctness**
  - Hard-coded substitution rules are error-prone.
  - No verification mechanism to validate substitutions.

TASO (Tensor Algebra SuperOptimizer)

- Automatic **generation of graph substitutions**.
- **Formal verification** of generated substitutions.
- **Joint optimization** over graph substitution and data layout.

TASO's Approach

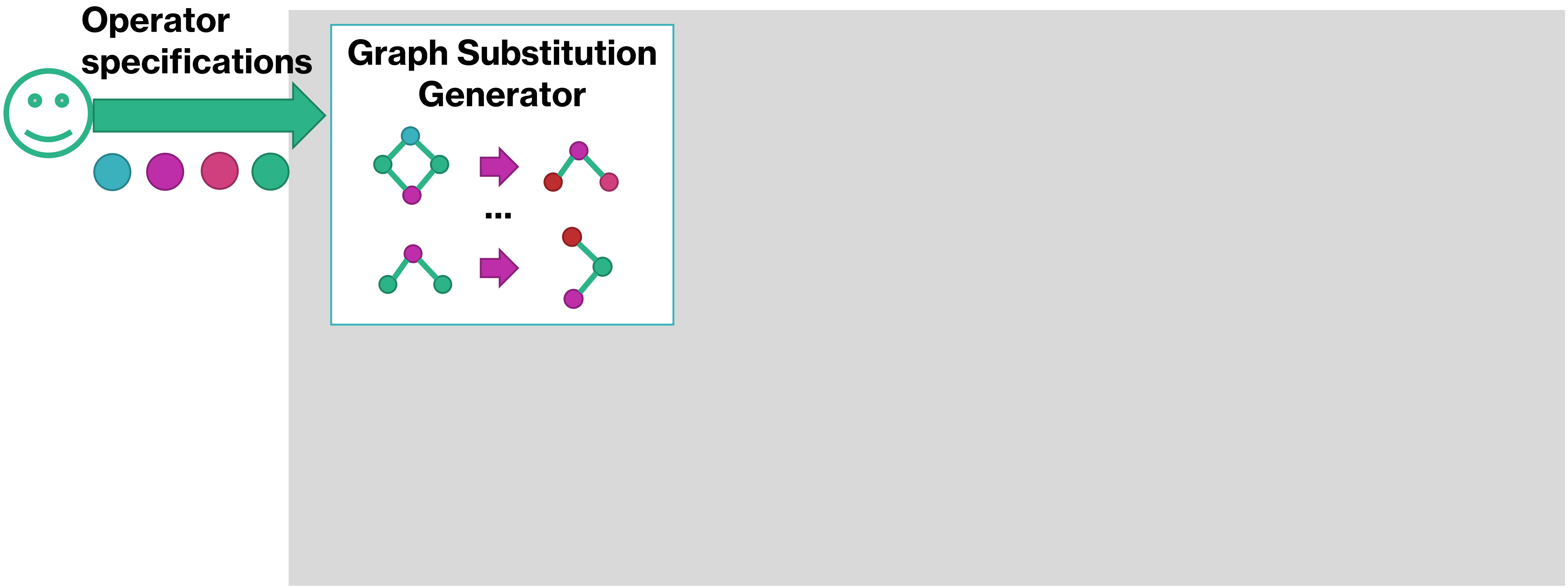


*Diagram adapted from Z. Jia et al., 2019

TASO's Approach

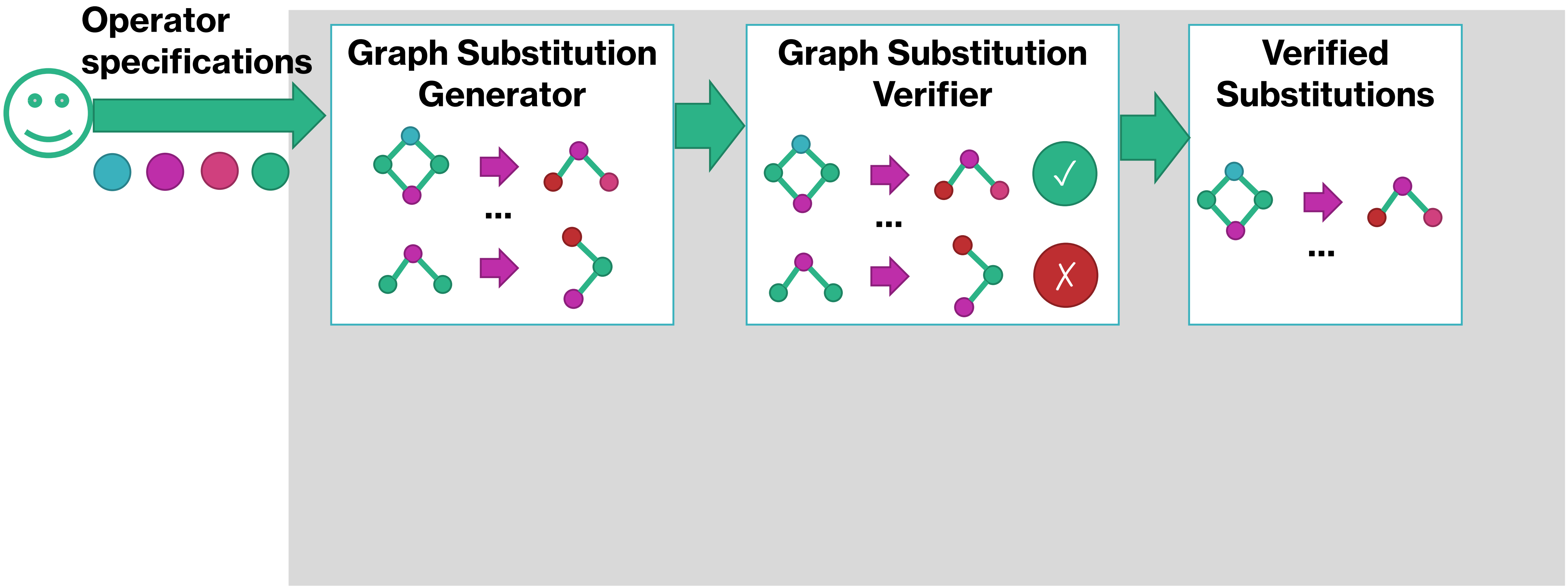


*Diagram adapted from Z. Jia et al., 2019

TASO's Approach

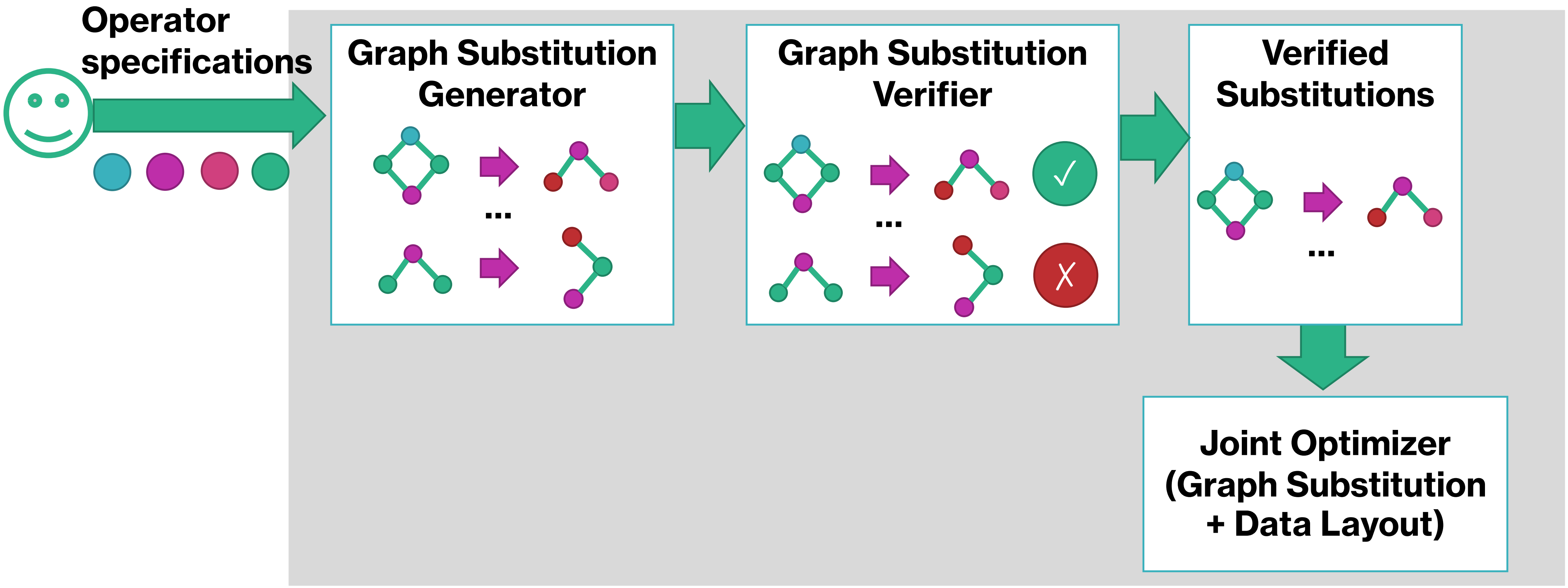


*Diagram adapted from Z. Jia et al., 2019

TASO's Approach

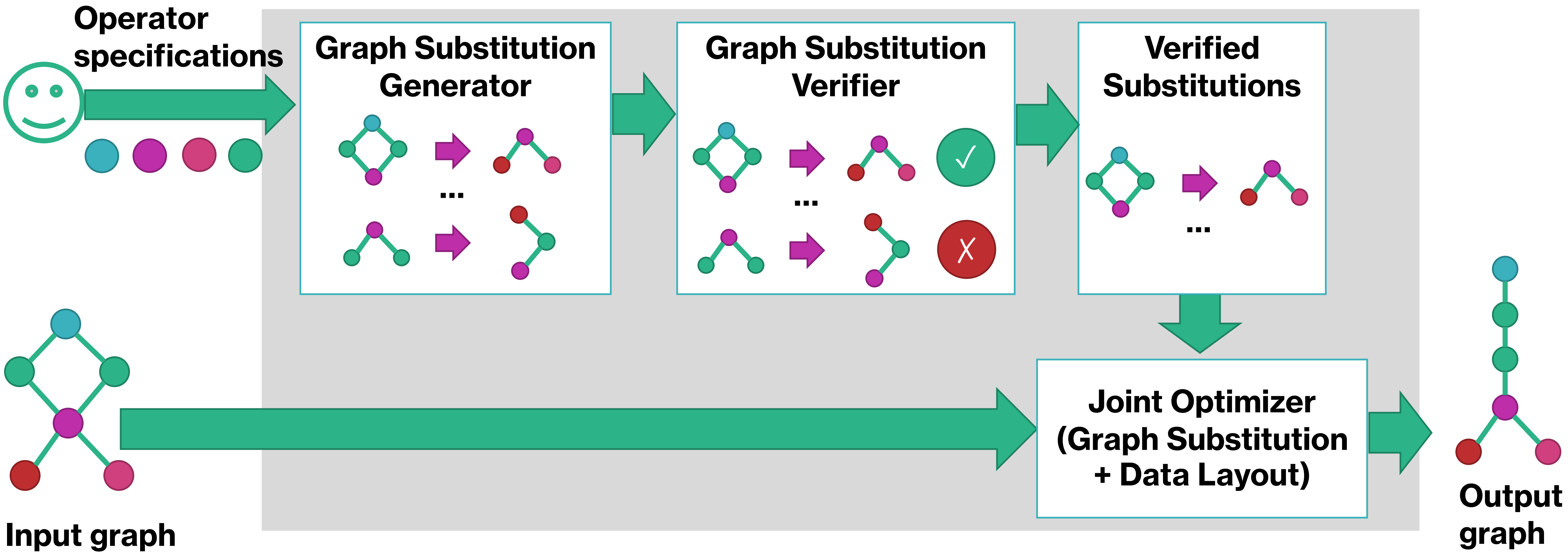


*Diagram adapted from Z. Jia et al., 2019

Graph Substitution Generator

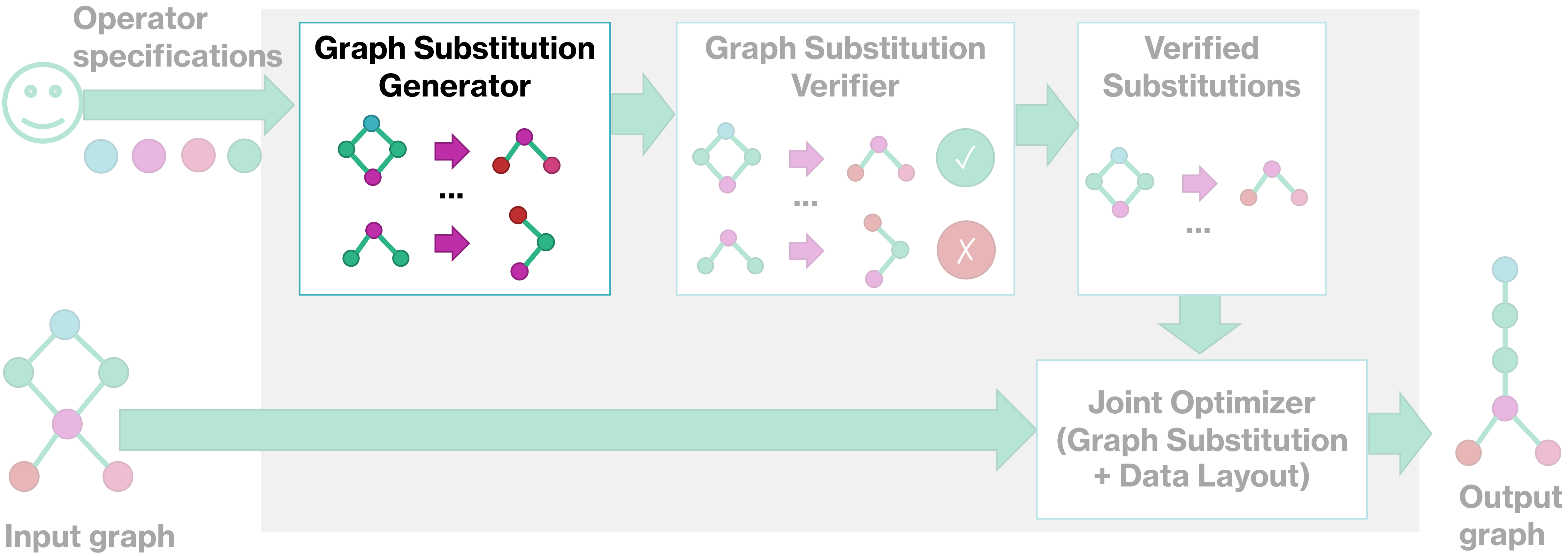
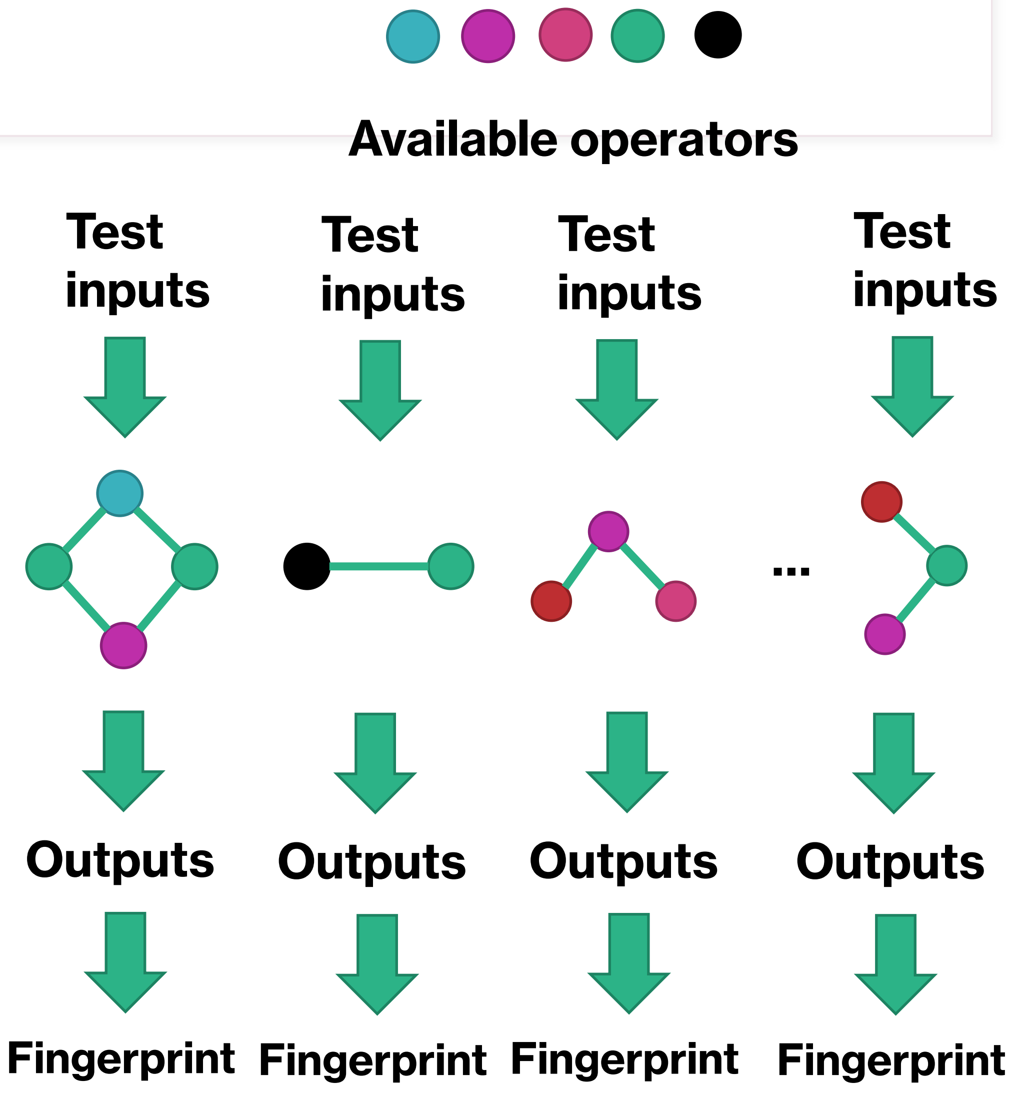


*Diagram adapted from Z. Jia et al., 2019

Graph Substitution Generator

1. Enumerate all possible graphs with **up to N operators** using depth-first-search.
2. Run the generated graphs on test inputs and obtain their **fingerprints** by hashing their outputs.
3. Maintain a dictionary D mapping $FingerPrint(G) \mapsto G$.
4. Iterate over fingerprints in D and return all pairs (G_1, G_2) with identical fingerprints.



Graph Substitution Verifier

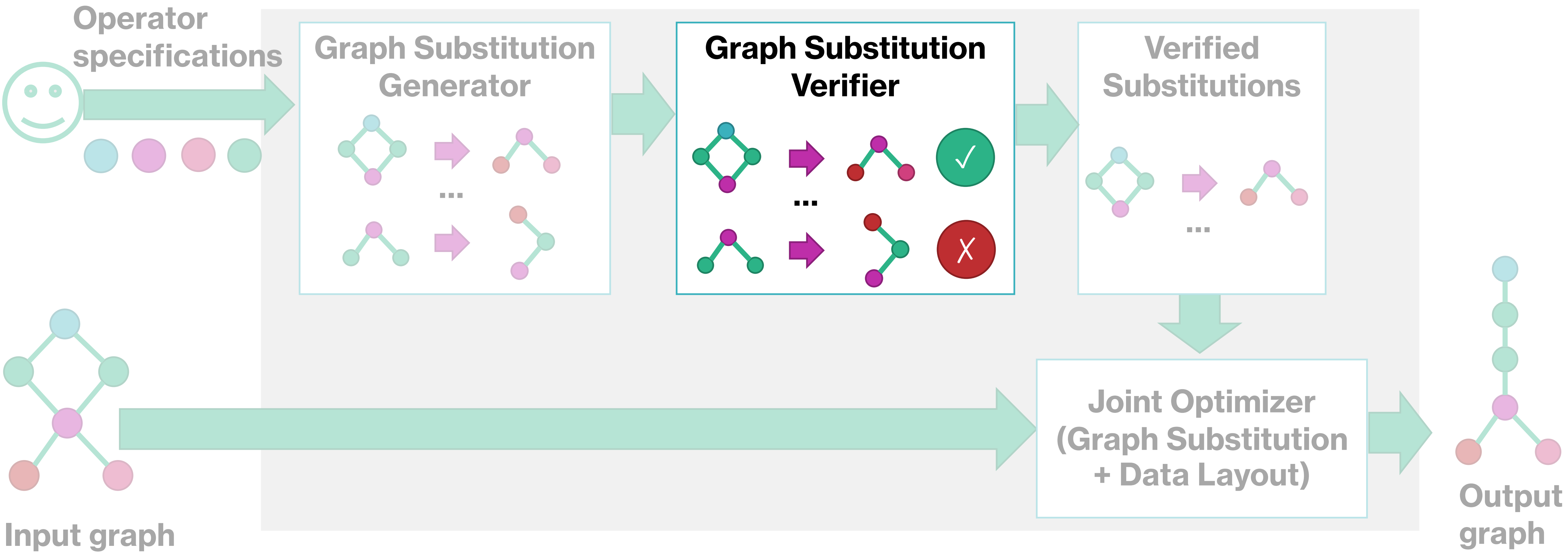


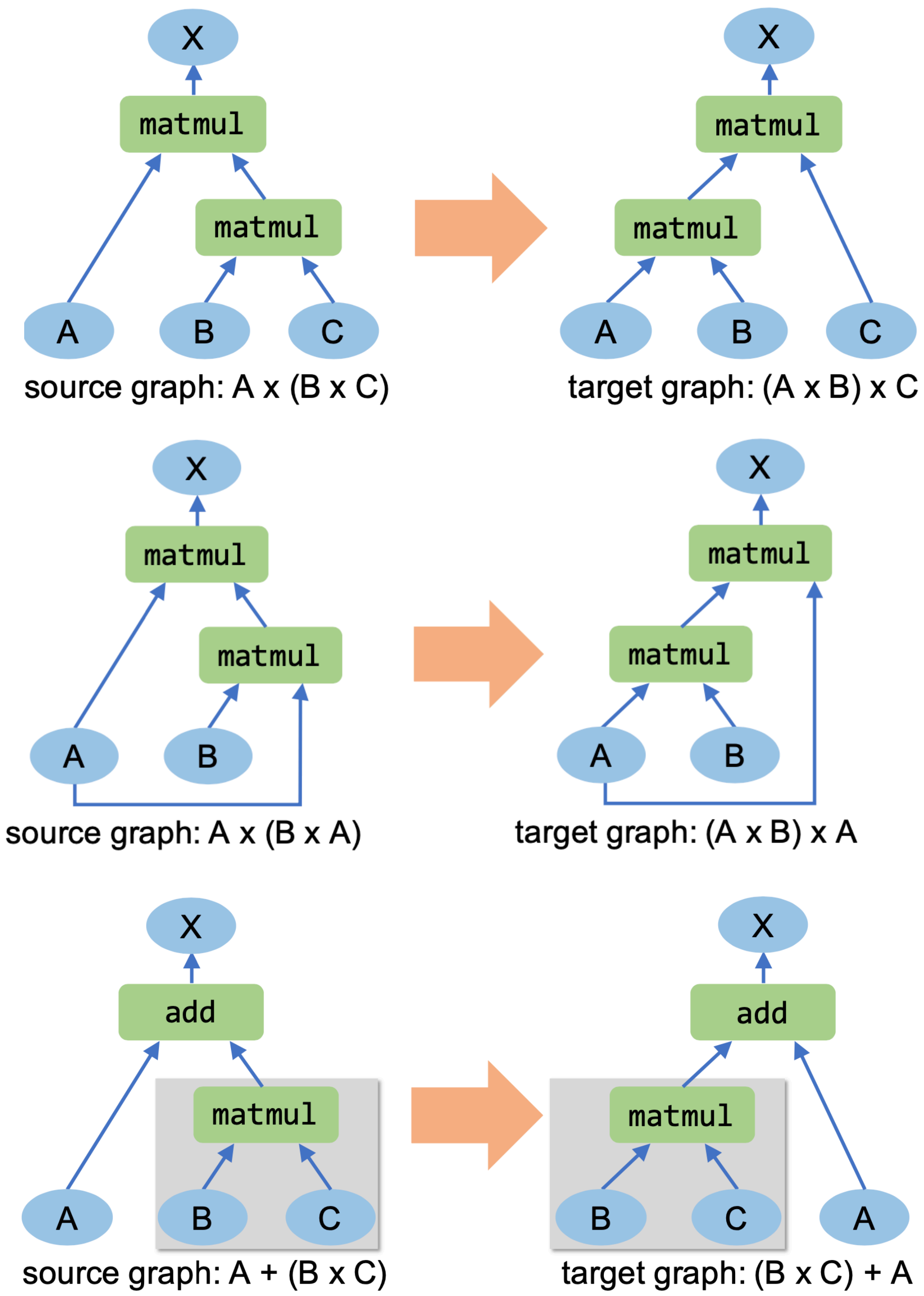
*Diagram adapted from Z. Jia et al., 2019

Graph Substitution Verifier

- Use a **theorem prover** to verify the generated substitutions satisfy **operator properties** defined in **first-order logic**.
- **43** operator properties defined in TASO including:
  - $\forall x. transpose\big(transpose(x)\big) = x$
  - $\forall x. matmul(x, I_{matmul}) = x$
  - $\forall x, y, z. matmul\big(x, matmul(y, x)\big) = matmul(matmul(x, y), z)$
- **Additional validation steps**
  - Testing operator properties on small tensors.
  - Checking if operator properties are consistent.

Pruning Redundant Substitutions

- A substitution is **redundant** if it can be inferred from another valid substitution.
- Pruning strategies in TASO:
  - Input tensor renaming
  - Common subgraph
- Pruning reduces the number of verified substitutions in TASO by **39 times**.



Joint Optimizer

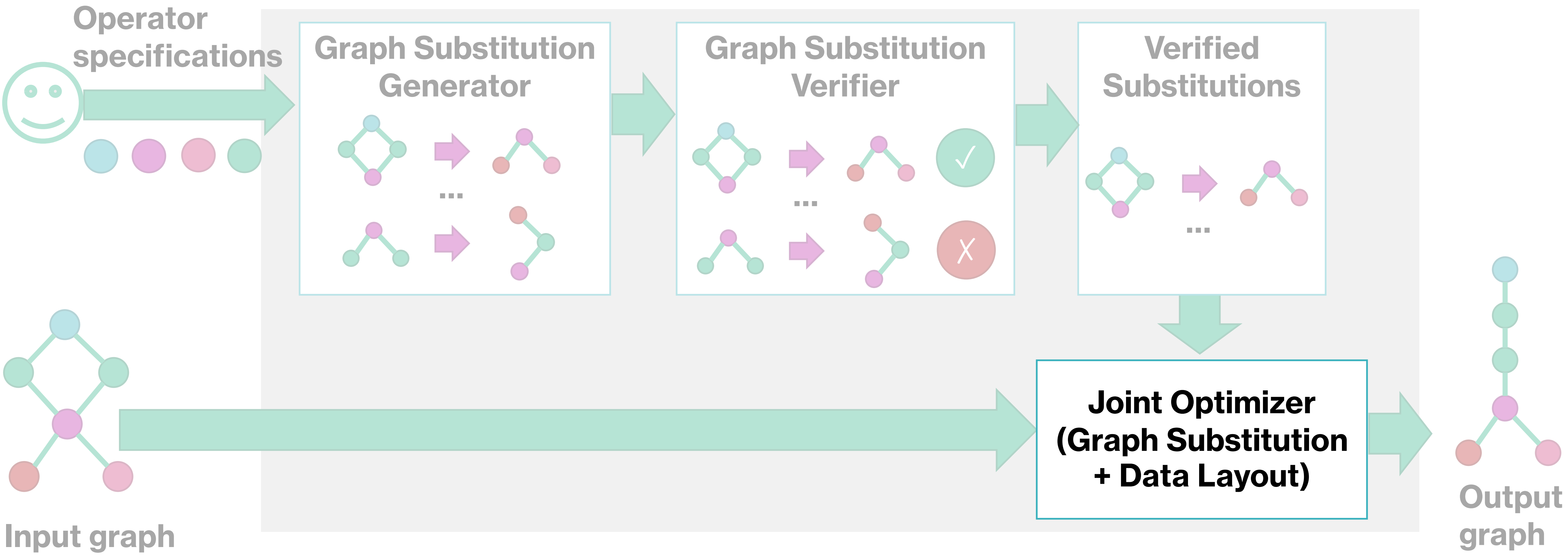
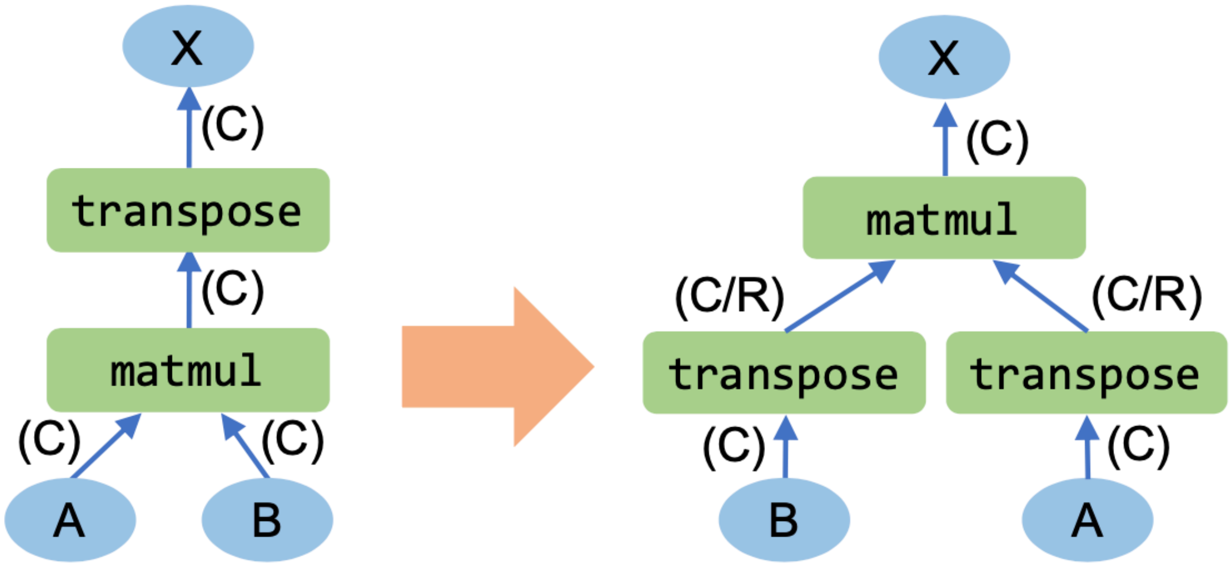


*Diagram adapted from Z. Jia et al., 2019

Joint Optimizer

- Layouts of **input** tensors (A, B) and **output** tensor (X) are fixed.
- But we can change the layout of **intermediate** tensors.



source graph: $(A \times B)^T$

target graph: $(B^T \times A^T)$

Joint Optimizer

- Using **cost-based backtracking search** defined by MetaFlow.
- Extended by TASO to account to account for different possible data layouts.
- The α parameter controls the search space. ($\alpha = 1.05$ in TASO)

Algorithm 2 Cost-Based Backtracking Search

Input: an input graph $\mathcal{G}_{in}$, verified substitutions $\mathcal{S}$, a cost model $Cost(\cdot)$, and a hyper parameter α.
Output: an optimized graph.

$\mathcal{P} = \{\mathcal{G}_{in}\}$ *// $\mathcal{P}$ is a priority queue sorted by Cost.*
while $\mathcal{P} \neq \{\}$ **do**
    $\mathcal{G} = \mathcal{P}$.dequeue()
    for substitution $s \in \mathcal{S}$ **do**
        // LAYOUT($\mathcal{G}, s$) returns possible layouts applying s on $\mathcal{G}$.
        for layout $l \in$ LAYOUT($\mathcal{G}, s$) **do**
            // APPLY($\mathcal{G}, s, l$) applies s on $\mathcal{G}$ with layout l.
            $\mathcal{G}' =$ APPLY($\mathcal{G}, s, l$)
            if $\mathcal{G}'$ is valid **then**
                if $Cost(\mathcal{G}') < Cost(\mathcal{G}_{opt})$ **then**
                    $\mathcal{G}_{opt} = \mathcal{G}'$
                if $Cost(\mathcal{G}') < \alpha \times Cost(\mathcal{G}_{opt})$ **then**
                    $\mathcal{P}$.enqueue($\mathcal{G}'$)
return $\mathcal{G}_{opt}$

Implementation

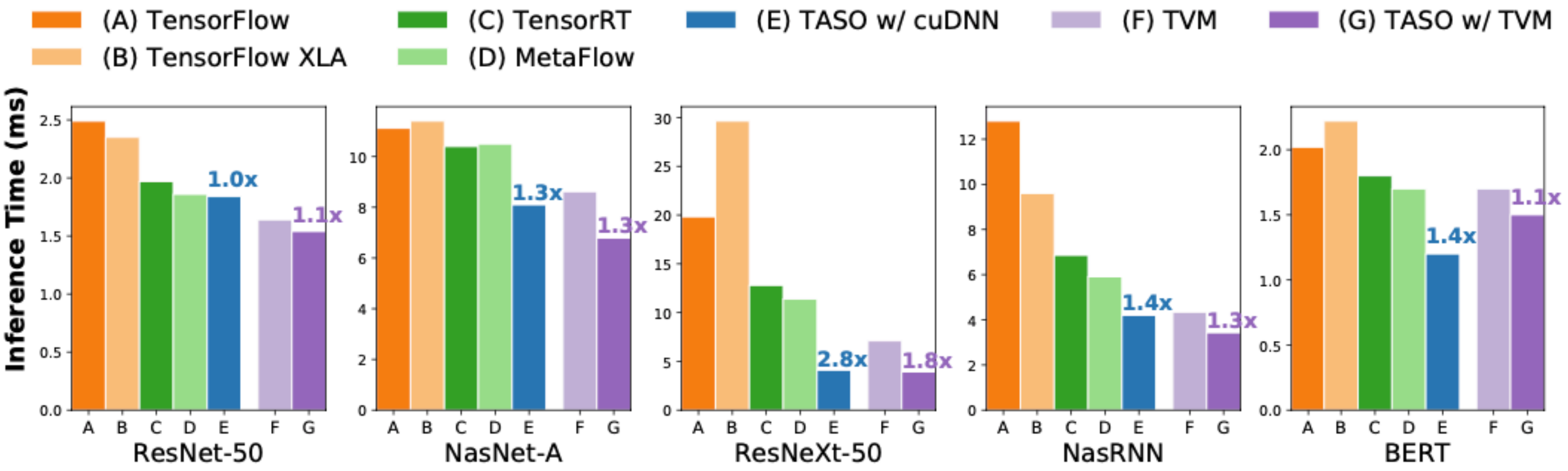
- TASO implementation includes **12 operators** and **43 operator properties**.
- Built on top of MetaFlow
  - **8000** lines of code in total.
  - **1400** lines for operator reference implementations + operator properties.
- Framework-agnostic.

Evaluation Setup

- Evaluated on optimizing **5 deep neural networks**:
  - ResNet-50
  - ResNeXt-50
  - NasNet-A
  - NasRNN
  - BERT
- Substitution generation:
  - Enumerated graphs with **up to 4 operators**.
  - Generated **743 verified substitutions**.

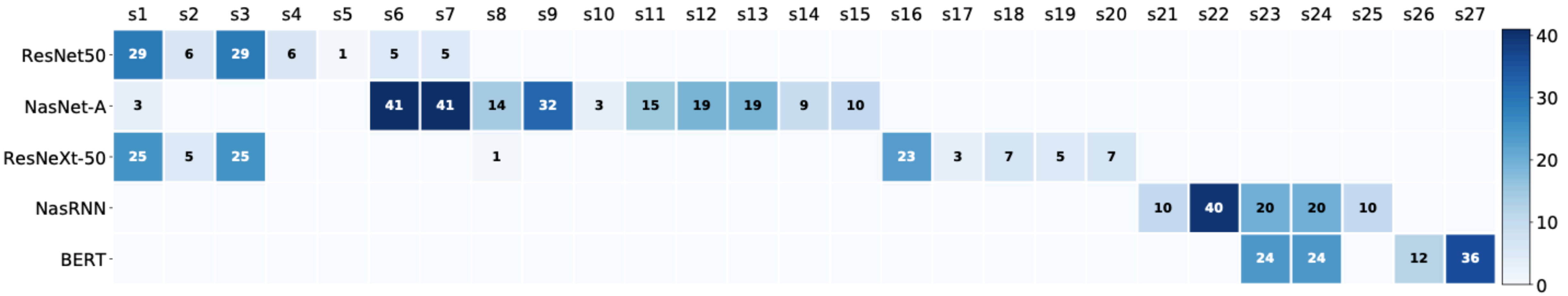
Performance Comparison

- Compared against **TensorFlow**, **TensorRT**, **MetaFlow**, **TVM** both on the **cuDNN** and **TVM** backends.



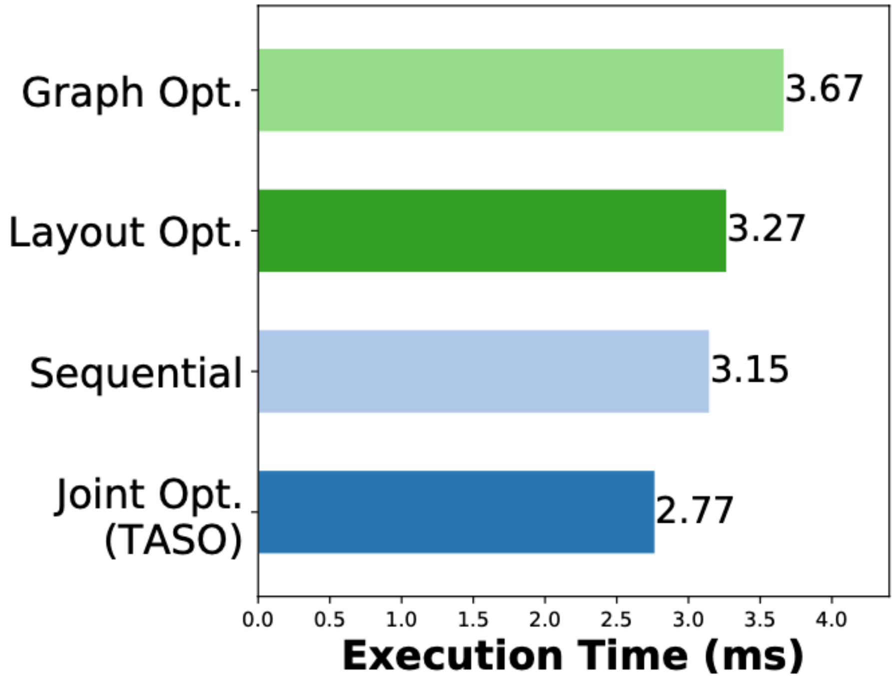
Why Does TASO Perform Better?

- Different architectures require different substitutions.
- Old heuristics do not necessarily apply to new models.



Why Does TASO Perform Better?

- Joint optimization compared with:
  - Only graph substitution optimizations.
  - Only data layout optimizations.
  - Sequential optimization.
- Joint optimization reduces execution time by 12% compared to sequential optimization.



Review

Strengths

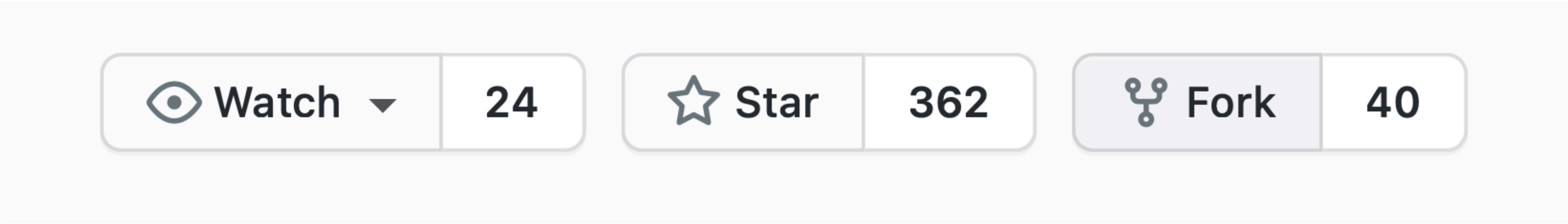
- Novel approach to optimizing DNNs via graph substitutions.
- Evaluation demonstrates clear improvement in performance.
- Good presentation of motivations and findings.

Potential Improvements

- Approach does not scale beyond graphs of 4 operators.
- No evaluation of system performance (e.g. memory consumption).
- Evaluation done only on a single machine.

Impact

- TASO builds on the findings of the MetaFlow paper:
  - https://theory.stanford.edu/~aiken/publications/papers/sysml19b.pdf
- TASO is publicly available on GitHub.
  - https://github.com/jiazhihao/TASO



Thank you for the attention!