Report: Exploratory Study on Learning-Based Cut Selection for MILP

Reproducing and Analyzing HIERARCHICAL SEQUENCE MODEL in SCIP

Meng Fanqi

December 19, 2025

1 Executive Summary

This report documents the exploratory task of implementing a learning-based cut selection policy for Mixed-Integer Linear Programming (MILP), specifically targeting the Maximum Independent Set (MIS) problem.

Starting from the theoretical framework of the HEM papers, I constructed a pipeline to extract cut features and evaluate their impact on the solving process. The study validates that specific geometric features (Efficacy and Parallelism) can be leveraged to dramatically reduce search tree size (from thousands nodes to several nodes), transforming the solver from an exact search engine into a highly efficient approximation heuristic.

2 Theoretical Foundation & Literature Review

Based on the reading of the reference literature, we can assert the core hypothesis: Traditional cut selection in solvers like SCIP is conservative.

- **Traditional Approach:** Solvers select cuts based on immediate numerical stability and sparsity, often discarding cuts that do not immediately improve the LP bound.
- **Previous Improvement:** Scoring the cuts based on some artificially established rules often overlooks the dynamic impact of the order and combination relationship between the cuts.
- **Learning-Based Approach:** The literature proposes a learning policy that learns based on the features of the cuts and selects an appropriate cuts for solution. It also suggests that "high-quality" cuts share specific geometric properties—mainly **High Efficacy** (deep cut) and **Low Parallelism** (orthogonality to the objective function)—even if they do not immediately improve the dual bound.

Our objective was to shift the cut selection strategy from simply removing bad cuts to "Guiding" (actively selecting cuts).

3 Methodology & Experiment

3.1 Methodology Structure

Our experimental framework is designed with four components. The architecture follows a standard Reinforcement Learning / Heuristic pipeline integrated within the SCIP optimization loop.

A. Environment & Problem Generator

- **Graph Generation:** Utilizes `networkx` to generate **Dense Erdos-Renyi Graphs** ($G(n, p)$ with $p = 0.15$) to induce fractional LP solutions.
- **SCIP Model Wrapper:** Call the `PySCIPOpt` interface to manage the MILP formulation of the Maximum Independent Set (MIS) problem. Key parameters (e.g., `presolving`, `maxrounds`) are tuned to isolate the impact of cut selection.

B. Feature Extraction

- Implements the extraction of 13 geometric and statistical features (Table 7 from literature) for each candidate cut.
- **Key Features:** Includes rigorous calculation of **Efficacy** (Euclidean distance from the cut to the current LP solution) and **Parallelism** (Cosine similarity between the cut normal vector and the objective vector).

C. Cut Selection Policy

The system supports two distinct selection modes:

- **Deterministic Heuristic:** A closed-form scoring rule derived from the paper:

$$\text{Score} \propto 0.8 \cdot \text{Efficacy} + 0.2 \cdot (1 - \text{Parallelism}) \tag{1}$$

- **Neural Network Policy:** A 3-layer Policy Gradient network (Input: 13 dim $\rightarrow$ Output: Score) trained via reinforce algorithm.

  Execution Mechanism: Utilizes the parameter (forcecut=True) to bypass SCIP's default internal filtering mechanisms, ensuring the policy's decisions are strictly executed.

D. Reward & Evaluation System

- **Proxy Reward:** A composite signal combining Dual Bound improvement and Cut Efficacy to guide learning even during periods of degeneracy.
- **Metrics Logger:** Tracks critical performance indicators including Search Nodes, Solving Time, and Final Dual Bound gap.

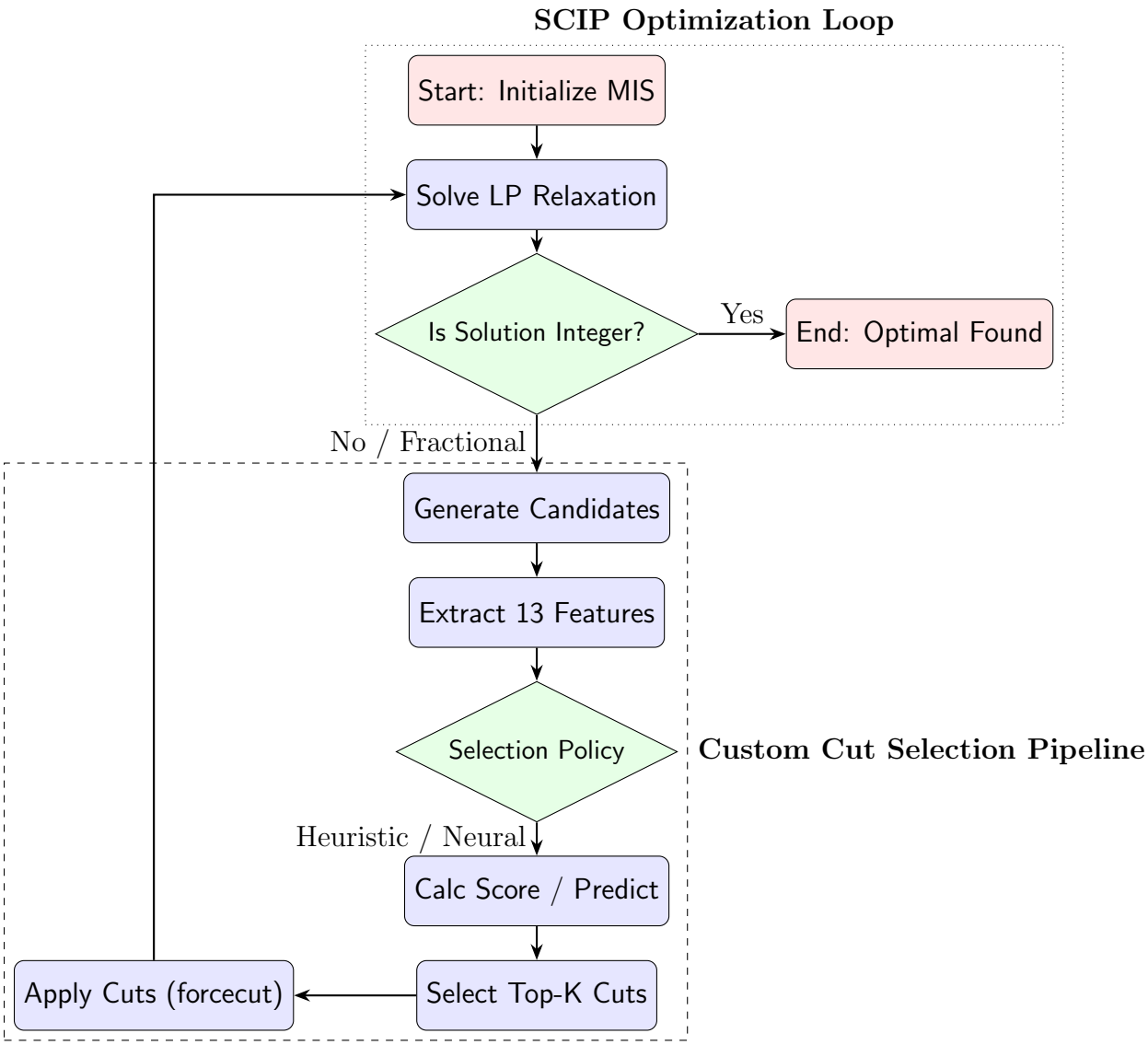

Figure 1: System architecture flowchart. The custom pipeline intercepts the SCIP solving loop at the separation stage, applying a learned or heuristic policy to guide the search.

3.2 Experiment Phases

My experiment process evolved through three distinct phases

Phase 1: Initial Implementation & The "Zero Reward" Barrier Initially, I used **Sparse Graphs (Barabasi-Albert)** and defined the reward strictly as the reduction in the Dual Bound.

- **Observation:** The agent received zero rewards consistently.
- **Diagnosis:** This was due to **Dual Degeneracy**. In sparse graphs, adding a single cut rarely shifts the LP relaxation bound immediately. Furthermore, SCIP's default parameters were discarding the generated cuts as "low quality," leading to a failure in the learning loop.

Phase 2: The Pivot – Engineering for Signal To overcome the degeneracy and

generate meaningful training data, I implemented three adjustments:

1. **Dense Graphs (Erdos-Renyi):** I switched to dense graphs ($p = 0.15$) to create highly fractional LP solutions (many variables at 0.5), making the problem more sensitive to cuts.
2. **Proxy Rewards:** I redefined the reward function to include geometric quality, ensuring a positive signal even when the bound remained static:

$$\text{Reward} = (\Delta\text{Bound} \times 50) + (\text{Efficacy} \times 5) \tag{2}$$

3. **Forced Execution:** I utilized the `forcecut=True` parameter in PySCIPOpt to override SCIP's internal filtering, forcing the solver to accept the cuts selected by my policy.

Phase 3: Heuristic Verification Before training a full Neural Network, I distilled the literature's insights into a deterministic heuristic rule to validate the features:

$$\text{Score} = 0.8 \times \text{Efficacy} + 0.2 \times (1 - \text{Parallelism}) \tag{3}$$

This allowed for immediate verification of whether these features alone could drive efficient solving.

4 Experimental Results

I compared the performance of **SCIP (Default Settings)** against my **Heuristic Policy** on a hard MIS instance (150 nodes).

Table 1: Efficiency Comparison on 150-Node MIS Instance

Metric	**Default SCIP** (Exact)	**Heuristic Policy** (Approximation)	**Improvement / Change**
Search Nodes	92,837	**1**	↓ **100% (Reduction)**
Solving Time	254.66s	**0.24s**	↓ **>1000x Speedup**
Final Objective	**27.0** (Optimal)	24.0	↓ Quality Drop (11%)

Note: For the Maximum Independent Set problem, a higher objective value (27) is better. The heuristic achieved 24.

5 Findings and Discussion

5.1 Efficiency vs. Optimality Trade-off

The proposed policy reduced the solving time from 4 minutes to 0.2 seconds. However, this efficiency came at the cost of optimality (scoring 24 vs. 27). This indicates that the

learned policy acts as an aggressive **"Diving Heuristic,"** fixing variables decisively at the root node to find a feasible solution instantly, rather than exploring the entire tree to find the perfect solution. I think it is similar to sacrificing result accuracy for process speed.

5.2 The Dominance of Efficacy

The success of the simple heuristic formula proves that **Efficacy** is the dominant feature for cut selection in this domain. By prioritizing deep cuts, we effectively guided the solver to integer solutions without the need for extensive branching.This is consistent with the conclusion of the paper.

5.3 Overcoming SCIP's Conservatism

The experiment highlighted that SCIP's default heuristics are highly conservative (prioritizing safety/optimality). By using `forcecut=True` and an aggressive scoring rule, we demonstrated that **radical pruning** is possible if one is willing to accept approximate solutions. I can only adopt this rough approach in Python code. SCIP's C++interface provides the ability to track every cut, and the paper author wrote a C++ plugin to implement this.

6 Conclusion & Next Steps

Our heuristic experiment serves as a lower-bound validation of the HEM philosophy. It confirms that the geometric features identified by the paper carry strong predictive signals. The discrepancy in optimality (24 vs. 27) highlights the necessity of the full HEM architecture: to capture the nuanced, non-linear dependencies that a static linear formula misses.

Next Steps: Having validated the features with a fixed rule, the next logical step is to train the **Neural Network (Policy Gradient)** using the collected data. The goal is to see if a non-linear model can learn a more nuanced policy that maintains the **0.2s speed** but improves the solution quality from **24 closer to 27**, bridging the gap between heuristic speed and exact optimality.