Zuse Institute Berlin

Takustr. 7
14195 Berlin
Germany

MARK TURNER[1], THORSTEN KOCH[2], FELIPE SERRANO[3], MICHAEL WINKLER

Adaptive Cut Selection in Mixed-Integer Linear Programming

[1] 0000-0001-7270-1496
[2] 0000-0002-1967-0077
[3] 0000-0002-7892-3951

ZIB Report 22-04 (February 2022)

Zuse Institute Berlin
Takustr. 7
14195 Berlin
Germany

Telephone: +49 30 84185-0
Telefax: +49 30 84185-125

E-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

ADAPTIVE CUT SELECTION IN MIXED-INTEGER LINEAR PROGRAMMING

A PREPRINT

Mark Turner[*†] turner@zib.de
Thorsten Koch[*†] koch@zib.de
Felipe Serrano[‡†] serrano@zib.de
Michael Winkler[§†] winkler@gurobi.com

February 22, 2022

ABSTRACT

Cut selection is a subroutine used in all modern mixed-integer linear programming solvers with the goal of selecting a subset of generated cuts that induce optimal solver performance. These solvers have millions of parameter combinations, and so are excellent candidates for parameter tuning. Cut selection scoring rules are usually weighted sums of different measurements, where the weights are parameters. We present a parametric family of mixed-integer linear programs together with infinitely many family-wide valid cuts. Some of these cuts can induce integer optimal solutions directly after being applied, while others fail to do so even if an infinite amount are applied. We show for a specific cut selection rule, that any finite grid search of the parameter space will always miss all parameter values, which select integer optimal inducing cuts in an infinite amount of our problems. We propose a variation on the design of existing graph convolutional neural networks, adapting them to learn cut selection rule parameters. We present a reinforcement learning framework for selecting cuts, and train our design using said framework over MIPLIB 2017. Our framework and design show that adaptive cut selection does substantially improve performance over a diverse set of instances, but that finding a single function describing such a rule is difficult. Code for reproducing all experiments is available at https://github.com/Opt-Mucca/Adaptive-Cutsel-MILP.

1 Introduction

A Mixed-Integer Linear Program (MILP) is an optimisation problem that is classically defined as:

$$\underset{\mathbf{x}}{\operatorname{argmin}}\{\mathbf{c}^\intercal \mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b},\ \mathbf{l} \leq \mathbf{x} \leq \mathbf{u},\ \mathbf{x} \in \mathbb{Z}^{|\mathcal{J}|} \times \mathbb{R}^{n-|\mathcal{J}|}\} \tag{1}$$

Here, $\mathbf{c} \in \mathbb{R}^n$ is the objective coefficient vector, $\mathbf{A} \in \mathbb{R}^{m \times n}$ is the constraint matrix, $\mathbf{b} \in \mathbb{R}^m$ is the right hand side constraint vector, $\mathbf{l}, \mathbf{u} \in \{\mathbb{R}, -\infty, \infty\}^n$ are the lower and upper variable bound vectors, and $\mathcal{J} \subseteq \{1, \dots, n\}$ is the set of indices of integer variables.

One of the main techniques for solving MILPs is the branch-and-cut algorithm, see [1] for an introduction. Generating cutting planes, abbreviated as *cuts*, is a major part of this algorithm, and is one of the most powerful techniques for quickly solving MILPs to optimality, see [2]. A cut is a constraint that does not remove any feasible solutions of (1) when added to the formulation. We restrict ourselves to linear cuts in this paper, and denote a cut as $\boldsymbol{\alpha} = (\alpha_0, \cdots, \alpha_n) \in \mathbb{R}^{n+1}$, and denote the set of feasible solutions as $\mathcal{I}_\mathcal{X}$, to formally define a cut in (2).

$$\sum_{i=1}^{n} \alpha_i x_i \leq \alpha_0, \quad \forall x \in \mathcal{I}_\mathcal{X}, \quad \text{where } \mathbf{x} = (x_1, \cdots, x_n) \tag{2}$$

[*]Chair of Software and Algorithms for Discrete Optimization, Institute of Mathematics, Technische Universität Berlin, Straße des 17. Juni 135, 10623 Berlin, Germany
[†]Zuse Institute Berlin, Department of Mathematical Optimization, Takustr. 7, 14195 Berlin
[‡]I[2]DAMO GmbH, Englerallee 19, 14195 Berlin, Germany
[§]Gurobi GmbH, Ulmenstr. 37-39, 60325 Frankfurt am Main, Germany

The purpose of cuts is to tighten the linear programming (LP) relaxation of (1), where the LP relaxation is obtained by removing all integrality requirements. Commonly, cuts are found that separate the current feasible solution to the LP relaxation, referred to as $\mathbf{x}^{LP}$, from the tightened relaxation, and for this reason are often called *separators*. This property is defined as follows:

$$\sum_{i=1}^{n} \alpha_i x_i^{LP} > \alpha_0, \quad \text{where} \quad \mathbf{x}^{LP} = (x_1^{LP}, \dots, x_n^{LP}) \tag{3}$$

Within modern MILP solvers, the cut aspect of the branch-and-cut algorithm is divided into cut generation and cut selection subproblems. The goal of cut generation is finding cuts that both tighten the LP relaxation at the current node and improve overall solver performance. The cut selection subproblem is then concerned with deciding which of the generated cuts to add to the formulation (1). That is, given the set of generated cuts $\mathcal{S}' = \{\boldsymbol{\alpha}_1, \cdots, \boldsymbol{\alpha}_{|\mathcal{S}'|}\}$, find a subset $\mathcal{S} \subseteq \mathcal{S}'$ to add to the formulation (1).

We focus on the cut selection subproblem in this paper, where we motivate the need for instance-dependent cut selection rules as opposed to fixed rules, and introduce a reinforcement learning (RL) framework for learning parameters of such a rule. The cut selection subproblem is important, as adding either all or none of the generated cuts to the LP usually results in poor solver performance. This is due to the large computational burden of solving larger LPs at each node when all cuts are added, and the large increase in nodes needed to solve MILPs when no cuts are added. For a summary on MILPs we refer readers to [1], for cutting planes [3], for cut selection [4], and for reinforcement learning [5].

The rest of the paper is organised as follows. In Section 2, we summarise existing literature on learning cut selection. In Section 3 we motivate the need for adaptive cut selection by showing worst case performance of fixed cut selection rules. This section was inspired by [6], which proved complexity results for fixed branching rules. In Section 4 we summarise how cut selection is performed in the MILP solver SCIP [7]. In Section 5 we show how to formulate cut selection as a Markov decision process, and phrase cut selection as a reinforcement learning problem. This section was motivated by [8], which presented variable selection as Markov decision process as well as experimental results of an imitation learning approach. Finally, we present a thorough computational experiment on learning cut selector parameters that improve root node performance over MIPLIB 2017 [9] in Section 6 using the MILP solver SCIP version 7.0.2 [7].

2 Related Work

Several authors have proposed cut selection rules and performed several computational studies. The thesis [1] presents a linear weighted sum cut selection rule, which drastically reduces solution time to optimality by selecting a reduced number of good cuts. This cut selection rule and algorithm, see [7], can still be considered the basis of what we use in this paper. A more in-depth guide to cutting plane management is given in [4]. Here, a large variety of cut measures are summarised and additional computational results given that show how a reduced subset of good cuts can drastically improve solution time. A further computational study, focusing on cut selection strategies for zero-half cuts, is presented in [10]. They hypothesise that generating a large amount of cuts followed by heuristic selection strategy is more effective than generating a few deep cuts. Note that the solver and cut selection algorithms used in [1], [4], and [10] are different. More recently, [11] summarises the current state of separators and cut selection in the literature, and poses questions aimed to better develop the science of cut selection. The final remark of the paper ponders whether machine learning can be used to answer some of the posed questions.

Recently, the intersection of mixed-integer programming and machine learning has received a lot of attention, specifically when it comes to branching, see [6, 8, 12] for examples. To the best of our knowledge, however, there are currently only three publications on the intersection of cut selection and machine learning. Firstly, [13] presents a reinforcement learning approach using evolutionary strategies for ranking Gomory cuts via neural networks. They show that their method outperforms standard measures, e.g. max violation, and generalises to larger problem sizes within the same class. Secondly, [14] train a neural network to rank linear cuts by expected objective value improvement when applied to a semi-definite relaxation. Their experiments show that substantial computational time can be saved when using their approximation, and that the gap after each cut selection round is very similar to that found when using the true objective value improvement. Most recently, [15] proposes a multiple instance learning approach for cut selection. They learn a scoring function parameterised as a neural network, which takes as input an aggregated feature vector over a bag of cuts. Their features are mostly composed of measures normally used to score cuts, e.g. norm violation. Cross entropy loss is used to train their network by labelling the bags of cuts before training starts.

Our contribution to the literature is three-fold. First, we provide motivation for instance dependent cut selection by proving the existence of a family of parametric MILPs together with an infinite amount of family-wide valid cuts.

Some of these cuts can induce integer optimal solutions directly after being applied, while others fail to do so even if an infinite amount are applied. Using a basic cut selection strategy and a pure cutting plane approach, we show that any finite grid search of the cut selector's parameter space, will miss all parameter values, which select integer optimal inducing cuts in an infinite amount of our instances. An interactive version of this constructive proof is provided in Mathematica® [16], and instance creation algorithms are provided using SCIP's Python API [7, 17]. Second, we introduce a RL framework for learning instance dependent cut selection rules, and present results on learning parameters to SCIP's default cut selection rule [1] over the data set MIPLIB 2017 [9]. Third and finally, we implemented a new cut selector plugin, which is available in SCIP 8.0 [18], and enables users to include their own cut selection algorithms in the larger MILP solving process.

3 Motivating Adaptive Cut Selection

We begin this section by introducing a simplified cut scoring rule and discussing how the parameters for such a score are traditionally set in solvers. Following this, a series of lemmas and a theorem will be proven using a simulated pure cutting plane approach.

Consider the following simplified version of SCIP's default cut scoring rule (see Section 4 for the default scoring rule):

$$\texttt{simple\_cut\_score}(\lambda, \boldsymbol{\alpha}, \mathbf{c}) := \lambda * \texttt{isp}(\boldsymbol{\alpha}) + (1-\lambda) * \texttt{obp}(\boldsymbol{\alpha}, \mathbf{c}), \quad \lambda \in [0,1], \boldsymbol{\alpha} \in \mathbb{R}^{n+1}, \mathbf{c} \in \mathbb{R}^n \tag{4}$$

Using the general MILP definition given in (1), we define the cut measures integer support (`isp`) and objective parallelism (`obp`) as follows:

$$\texttt{isp}(\boldsymbol{\alpha}) := \frac{\sum_{i \in \mathcal{J}} \texttt{nonzero}(\alpha_i)}{\sum_{i=1}^{n} \texttt{nonzero}(\alpha_i)}, \text{ where } \quad \texttt{nonzero}(\alpha_i) = \begin{cases} 0 & \text{if } \alpha_i = 0 \\ 1 & \text{otherwise} \end{cases} \tag{5}$$

$$\texttt{obp}(\boldsymbol{\alpha}, \mathbf{c}) := \left| \frac{\sum_{i=1}^{n} \alpha_i c_i}{\sqrt{\sum_{i=1}^{n} \alpha_i^2} \sqrt{\sum_{i=1}^{n} c_i^2}} \right| \tag{6}$$

We now introduce Theorem 3.1, which refers to the λ parameter in (4).

Theorem 3.1. *Given a finite discretisation of λ, an infinite family of MILP instances together with an infinite amount of family-wide valid cuts can be constructed. Using a pure cutting plane approach and applying a single cut per selection round, the infinite family of instances do not solve to optimality for any value in the discretisation, but do solve to optimality for an infinite amount of alternative λ values.*

The general purpose of Theorem 3.1 is to motivate the need for instance dependent parameters in the cut selection subroutine. The typical approach for finding the best choice of cut selector parameters, see previous SCIP computational studies [1, 7, 18], is to perform a parameter sweep, most often a grid search. A grid search, however, leaves regions unexplored in the parameter space. In our simplified cut scoring rule (4), we have a single parameter, namely λ, and these unexplored regions are simply intervals. We define Λ, the set of values in the finite grid search of λ, as follows:

$$\Lambda := \{\lambda_1, \ldots, \lambda_{|\Lambda|}\}, \quad \text{where } \; 0 \leq \lambda_i < \lambda_{i+1} \leq 1 \quad \forall i \in \{1, \cdots, n-1\}, \quad |\Lambda| \in \mathbb{N}$$

The unexplored intervals in the parameter space, denoted $\tilde{\Lambda}$, is then defined as:

$$\tilde{\Lambda} := \{[0, \lambda_1) \cup (\lambda_1, \lambda_2) \cup \cdots \cup (\lambda_{n-1}, \lambda_n) \cup (\lambda_{|\Lambda|}, 1]\}$$

Our goal is to show that for any Λ, we can construct an infinite family of MILP instances from Theorem 3.1. Together with our infinite amount of family-wide valid cuts and specific cut selection rule, we will show that the solving process does not finitely terminate for any choice of λ outside of an interval $(\lambda_{lb}, \lambda_{ub}) \subset \tilde{\Lambda}$. In effect, this shows that using the same fixed λ value over all problems in a MILP solver, could result in incredibly poor performance for many problems. This is somewhat expected, as a fixed parameter cannot be expected to perform well on all possible instances, and moreover, cut selection is only a small subroutine in the much larger MILP solving process. Additionally, the instance space of MILPs is non-uniform, and good performance over certain problems may be highly desirable as they occur more frequently in practice.

3.1 Proof of Theorem

For the following theorem, we will simulate a pure cutting plane approach to solving MILPs using scoring rule (4). We will use custom MILPs, cutting planes, and select exactly one cut per round. Each call to the selection subroutine is called an *iteration* or *round*. The theorem is intended to show how a fixed cut selection rule can consistently choose "bad" cuts.

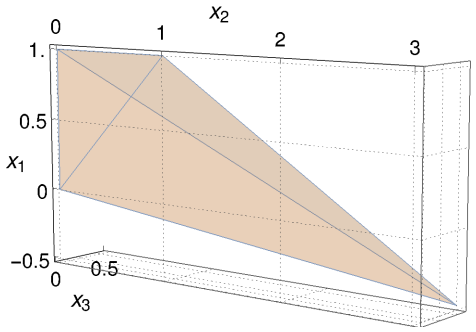

Figure 1: The feasible region of the LP relaxation of $\mathsf{P}(a, d)$.

Theorem 3.1. *Given a finite discretisation of λ, an infinite family of MILP instances together with an infinite amount of family-wide valid cuts can be constructed. Using a pure cutting plane approach and applying a single cut per selection round, the infinite family of instances do not solve to optimality for any value in the discretisation, but do solve to optimality for an infinite amount of alternative λ values.*

The parametric MILP we use to represent our infinite family of instances is defined as follows, where $a \in \mathbb{R}_{\geq 0}$ and $d \in [0, 1]$:

$$\mathsf{P}(a,d) := \begin{cases} \min & x_1 - (10+d)x_2 - ax_3 \\ & -\frac{1}{2}x_2 + 3x_3 \leq 0 \\ & -x_3 \leq 0 \\ \text{s.t.} & -\frac{1}{2}x_1 + \frac{1}{2}x_2 - \frac{7}{2}x_3 \leq 0 \\ & \frac{1}{2}x_1 + \frac{3}{2}x_3 \leq \frac{1}{2} \\ & x_1 \in \mathbb{Z}, \quad x_2 \in \mathbb{R}, \quad x_3 \in \{0,1\} \end{cases}$$

The polytope of our MILPs LP relaxation is the convex hull of the following points:

$$\mathcal{X} := \{(0,0,0), (1,0,0), (1,1,0), (\frac{-1}{2}, 3, \frac{1}{2})\} \tag{7}$$

The convex hull of $\mathcal{X}$ is a 3-simplex, or alternatively a tetrahedron, see Figure 1 for a visualisation. For such a feasible region, we can exhaustively write out all integer feasible solutions:

Lemma 3.2. *The integer feasible set of $\mathsf{P}(a, d)$, $(a, d) \in \mathbb{R}_{\geq 0} \times [0, 1]$ is:*

$$\{(0,0,0)\} \cup \{(1, x_2, 0) : \forall x_2 \in [0,1]\}$$

As we are dealing with linear constraints and objectives, we know that $(1, x_2, 0)$, where $0 < x_2 < 1$ cannot be optimal without both $(1, 1, 0)$ and $(1, 0, 0)$ being also optimal. We therefore simplify the integer feasible set, $\mathcal{I}_{\mathcal{X}}$, to:

$$\mathcal{I}_{\mathcal{X}} := \{(0,0,0), (1,0,0), (1,1,0)\} \tag{8}$$

At each iteration of adding cuts we will always present exactly three candidate cuts. We name these cuts as follows:

- The 'good cut', denoted $\mathcal{GC}$: Applying this cut immediately results in the next LP solution being integer optimal.
- The 'integer support cut', denoted $\mathcal{ISC}^n$: Applying this cut will result in a new LP solution barely better than the previous iteration. The cut has very high integer support as the name suggests, and would be selected if λ from (4) is set to a high value. The superscript n refers to the iteration number.
- The 'objective parallelism cut', denoted $\mathcal{OPC}^n$: Applying this cut will also result in a new LP solution barely better than the previous iteration. The cut has very high objective parallelism, and would be selected if λ from (4) is set to a low value. The superscript n refers to the iteration number.

The cuts are defined as follows, where $\mathcal{GC}$ has an additional property of it being selected in the case of a scoring tie:

$$\mathcal{GC} : -10x_1 + 10x_2 + x_3 \leq 0 \tag{9}$$

$$\mathcal{ISC}^n : -x_1 + x_3 \leq 1 - \epsilon_n \tag{10}$$

$$\mathcal{OPC}^n : -x_1 + 10x_2 \leq \frac{61}{2} - \epsilon_n \tag{11}$$

We use ϵ_n here to denote a small shift of the cut, with a greater ϵ_n resulting in a deeper cut. We define ϵ_n as follows:

$$\begin{gathered} 0 < \epsilon_i < \epsilon_{i+1} \quad \forall i \in \mathbb{N} \\ \lim_{n \to \infty} \epsilon_n = 0.1 \end{gathered} \tag{12}$$

A better overview of the proof of Theorem 3.1 can now be imagined. At each cut selection round we present three cuts, where for high values of λ, $\mathcal{ISC}^n$ is selected, and for low values $\mathcal{OPC}^n$ is selected. As the scoring rule (4) is linear w.r.t. λ, our aim is to controllably sandwich the intermediate values of λ that will select $\mathcal{GC}$. Specifically, for any given Λ, we aim to construct an infinite amount of parameter values for a and d s.t. the intermediate values of λ all belong to $\tilde{\Lambda}$.

Lemma 3.3. *The vertex set of the LP relaxation of* $\mathsf{P}(a,d), (a,d) \in \mathbb{R}_{\geq 0} \times [0,1]$*, after having individually applied cuts* (9)*,* (10)*, or* (11) *are, respectively:*

$$\mathcal{GC}_{\mathcal{X}} := \mathcal{I}_{\mathcal{X}} \cup \{(\frac{61}{91}, \frac{60}{91}, \frac{10}{91})\} \tag{13}$$

$$\mathcal{ISC}^n_{\mathcal{X}} := \mathcal{I}_{\mathcal{X}} \cup \{(\frac{-1}{2} + \frac{3\epsilon_n}{4}, 3 - \frac{3\epsilon_n}{2}, \frac{1}{2} - \frac{\epsilon_n}{4}), (\frac{-1}{2} + \frac{3\epsilon_n}{4}, 3 - \epsilon_n, \frac{1}{2} - \frac{\epsilon_n}{4}), (\frac{-1}{2} + \frac{\epsilon_n}{2}, 3 - 3\epsilon_n, \frac{1}{2} - \frac{\epsilon_n}{2})\} \tag{14}$$

$$\mathcal{OPC}^n_{\mathcal{X}} := \mathcal{I}_{\mathcal{X}} \cup \{(\frac{-1}{2} + \frac{\epsilon_n}{21}, 3 - \frac{2\epsilon_n}{21}, \frac{1}{2} - \frac{\epsilon_n}{63}), (\frac{-1}{2} + \frac{3\epsilon_n}{43}, 3 - \frac{4\epsilon_n}{43}, \frac{1}{2} - \frac{\epsilon_n}{43}), (\frac{-1}{2} + \frac{\epsilon_n}{61}, 3 - \frac{6\epsilon_n}{61}, \frac{1}{2} - \frac{\epsilon_n}{61})\} \tag{15}$$

Proof. Apply $\mathcal{GC}$, $\mathcal{ISC}^n$, and $\mathcal{OPC}^n$ to $\mathsf{P}(a,d)$ individually and then compute the vertices of the convex hull of the LP relaxation. □

We note that because both integer support and objective parallelism do not depend on the current LP solution, iteratively applying deeper cuts of the same kind would leave the cut's scores unchanged. Thus, provided they do not separate any integer points and continue to cut off the LP solution, deeper cuts of the same kind can be recursively applied. This is why both $\mathcal{ISC}^n$ and $\mathcal{OPC}^n$ have a superscript.

After applying a cut of one kind, e.g. $\mathcal{ISC}^n$, we cannot always simply increment n in the other cut, e.g. $\mathcal{OPC}^{n+1}$. This is because $\mathcal{OPC}^{n+1}$ does not guarantee separation of the now new LP solution in problem $\mathsf{P}(a,d) \cap \mathcal{ISC}^n$ for all sequences of $\{\epsilon_n, \epsilon_{n+1}\}$. Instead, we create a variant of $\mathcal{OPC}^{n+1}$, namely $\widehat{\mathcal{OPC}^{n+1}}$, which will always entirely remove the facet of the LP created by adding $\mathcal{ISC}^n$ independent of how the series of ϵ_n values increase. Once again, we note that as only the RHS values are changing, the score for all cuts within the same type remain unchanged, and thus no two cuts from different types can be applied. We have proven our results using Mathematica [16], and a complete notebook containing step-by-step instructions can be found at x. Below we will outline the necessary cumulative lemmas to prove Theorem 3.1, and summarise the calculations we have taken to achieve each step.

Lemma 3.4. *Having applied the cut* $\mathcal{ISC}^n$ *to* $\mathsf{P}(a,d), (a,d) \in \mathbb{R}_{\geq 0} \times [0,1]$*, a new facet is created. Applying either* $\mathcal{GC}$ *or a deeper variant of* $\mathcal{OPC}^{n+1}$ *cuts off that facet. The deeper variant, denoted* $\widehat{\mathcal{OPC}^{n+1}}$*, which differs from* $\mathcal{OPC}^{n+1}$ *only by the RHS value is defined as:*

$$\widehat{\mathcal{OPC}^{n+1}} : -x_1 + 10x_2 \leq \frac{61}{2} - 31\epsilon_n \tag{16}$$

Proof. One can first verify that the vertex set of the facet is $\mathcal{ISC}^n_{\mathcal{X}} \setminus \mathcal{I}_{\mathcal{X}}$. One can then find the smallest ϵ' s.t the following cut is valid for all $\mathbf{x} \in \mathcal{ISC}^n_{\mathcal{X}} \setminus \mathcal{I}_{\mathcal{X}}$:

$$-x_1 + 10x_2 \leq \frac{61}{2} - \epsilon'$$

The statement is valid for all $\epsilon' > \frac{61\epsilon_n}{2}$, and we arbitrarily select $\epsilon' = 31\epsilon_n$. One can also check that $\mathcal{GC}$ dominates $\mathcal{ISC}^n$ by seeing that it separates all vertices of $\mathcal{ISC}^n_{\mathcal{X}} \setminus \mathcal{I}_{\mathcal{X}}$ for all $n \in \mathbb{N}$. Finally, we need to ensure that no integer solution is cut off. We can verify this by checking that every $\mathbf{x} \in \mathcal{I}_{\mathcal{X}}$ satisfies (16). This statement holds whenever $\epsilon_n < 0.1$. Therefore $\epsilon' = 31\epsilon_n$ is valid, and we arrive at the cut $\widehat{\mathcal{OPC}^{n+1}}$. □

Lemma 3.5. *Having applied the cut $\mathcal{OPC}^n$ to $\mathsf{P}(a,d)$, $(a,d) \in \mathbb{R}_{\geq 0} \times [0,1]$, a new facet is created. Applying $\mathcal{ISC}^{n+1}$ or $\mathcal{GC}$ cuts off that facet.*

Proof. This follows the same structure as the proof of Lemma 3.4. We get that $\epsilon' > \frac{4\epsilon_n}{43}$, and that $\epsilon' = \epsilon_{n+1}$ is valid w.r.t. the integer constraints. □

Using our definition of integer support and objective parallelism in (5)-(6), we derive the scores for each cut from the simple cut selection scoring rule (4). We let $\mathbf{c}_{\mathsf{P}(a,d)}$ denote the vector of coefficients from the objective of $\mathsf{P}(a,d)$, $(a,d) \in \mathbb{R}_{\geq 0} \times [0,1]$. The integer support and objective parallelism values of each cut are as follows:

$$\texttt{isp}(\mathcal{GC}) = \frac{2}{3} \tag{17}$$

$$\texttt{isp}(\mathcal{ISC}^n) = 1 \quad \forall n \in \mathbb{N} \tag{18}$$

$$\texttt{isp}(\mathcal{OPC}^n) = \frac{1}{2} \quad \forall n \in \mathbb{N} \tag{19}$$

$$\texttt{obp}(\mathcal{GC}, \mathbf{c}_{\mathsf{P}(a,d)}) = \frac{110 + a + 10d}{\sqrt{201}\sqrt{1 + a^2 + (10+d)^2}} \tag{20}$$

$$\texttt{obp}(\mathcal{ISC}^n, \mathbf{c}_{\mathsf{P}(a,d)}) = \frac{1 + a}{\sqrt{2}\sqrt{1 + a^2 + (10+d)^2}} \quad \forall n \in \mathbb{N} \tag{21}$$

$$\texttt{obp}(\mathcal{OPC}^n, \mathbf{c}_{\mathsf{P}(a,d)}) = \frac{101 + 10d}{\sqrt{101}\sqrt{1 + a^2 + (10+d)^2}} \quad \forall n \in \mathbb{N} \tag{22}$$

Using our simplified cut scoring rule as defined in (4), we derive the necessary conditions defining the λ values, which assign $\mathcal{GC}$ a score at least as large as the other cuts.

Lemma 3.6. *$\mathcal{GC}$ is selected and added to $\mathsf{P}(a,d)$, $(a,d) \in \mathbb{R}_{\geq 0} \times [0,1]$, using scoring rule (4) if and only if a λ is used that satisfies the following conditions:*

$$\lambda * \texttt{isp}(\mathcal{GC}) + (1-\lambda) * \texttt{obp}(\mathcal{GC}, \mathbf{c}_{\mathsf{P}(a,d)}) \geq \lambda * \texttt{isp}(\mathcal{ISC}^n) + (1-\lambda) * \texttt{obp}(\mathcal{ISC}^n, \mathbf{c}_{\mathsf{P}(a,d)}) \tag{23}$$

$$\lambda * \texttt{isp}(\mathcal{GC}) + (1-\lambda) * \texttt{obp}(\mathcal{GC}, \mathbf{c}_{\mathsf{P}(a,d)}) \geq \lambda * \texttt{isp}(\mathcal{OPC}^n) + (1-\lambda) * \texttt{obp}(\mathcal{OPC}^n, \mathbf{c}_{\mathsf{P}(a,d)}) \tag{24}$$

Proof. We know that the integer support and objective parallelism do not depend on ϵ_n as seen in equations (17)-(22). Our cut selector rule also selects exactly one cut per iteration, namely the largest scoring cut. Therefore, whenever λ satisfies constraints (23) and (24), $\mathcal{GC}$ will be selected over both $\mathcal{OPC}^n$ and $\mathcal{ISC}^n$, and applied to $\mathsf{P}(a,d)$. If λ does not satisfy constraints (23) and (24), then $\mathcal{GC}$ is not the largest scoring cut and will not be applied to $\mathsf{P}(a,d)$. □

The inequalities (23)-(24) define the region, $\mathcal{R}_{\mathcal{GC}}$, which exactly contains all tuples $(a,d,\lambda) \in \mathbb{R}_{\geq 0} \times [0,1]^2$ that result in $\mathcal{GC}$ being the best scoring cut. The region, $\mathcal{R}_{\mathcal{GC}}$ is visualised in Figure 2. We define the function $\mathtt{r}_{\mathcal{GC}}(a,d)$ for all $(a,d) \in \mathbb{R}_{\geq 0} \times [0,1]$, which maps any pairing of (a,d) to the set of λ values contained in $\mathcal{R}_{\mathcal{GC}}$ for the corresponding fixed (a,d) values.

$$\mathtt{r}_{\mathcal{GC}} : \mathbb{R}_{\geq 0} \times [0,1] \to \mathcal{P}([0,1]) \tag{25}$$

Here $\mathcal{P}$ refers to the power set. We are interested in $\mathcal{R}_{\mathcal{GC}}$ as we believe that we can find a continuous function that contains all $(a,d,\lambda) \in \mathbb{R}_{\geq 0} \times [0,1]^2$ pairings, which score all cuts equally. Using this function, we can find for a fixed $d \in [0,1]$, the values of $a \in \mathbb{R}_{\geq 0}$ that would result in a $\lambda \in [0,1]$ value that scores all cuts equally. By perturbing our value of $a \in \mathbb{R}_{\geq 0}$, we aim to generate an infinite amount of $\lambda \in [0,1]$ values that score $\mathcal{GC}$ the largest. Then, by choosing different values of $d \in [0,1]$ originally, we aim to find such an infinite set of λ values that can adaptively lie between any given finite discretisation of $[0,1]$.

Lemma 3.7. *There exists a closed form symbolic solution for the maximum value of a in terms of d over $\mathcal{R}_{\mathcal{GC}}$. We denote this maximum value of a as a function over d, namely $\mathtt{a}_{\max}(d)$, $d \in [0,1]$.*

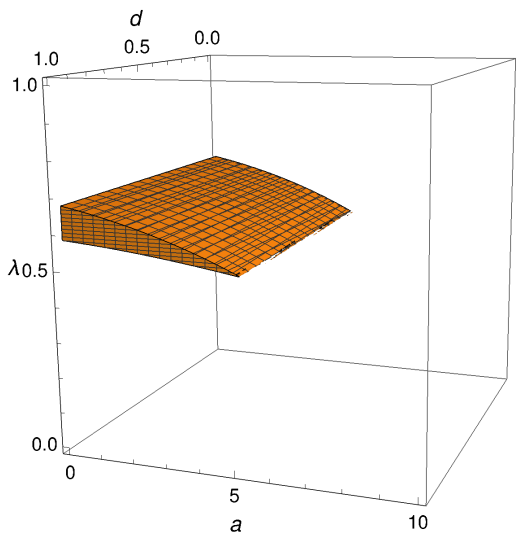

Figure 2: The region, $\mathcal{R}_{\mathcal{GC}}$, where $\mathcal{GC}$ is scored at least as large as both $\mathcal{OPC}^n$ and $\mathcal{ISC}^n$ under cut selection rule (4).

Proof. We verify this by solving the following optimisation problem, where $\mathtt{a}_{\max}(d)$ is printed in Appendix A:

$$\underset{a,\lambda}{\operatorname{argmax}}\{a \mid (a,d,\lambda) \in \mathcal{R}_{\mathcal{GC}},\ \ a \geq 0,\ \ 0 \leq d \leq 1,\ \ 0 \leq \lambda \leq 1\}$$

□

Lemma 3.8. *Closed form symbolic solutions for the upper and lower bounds of λ can be found. These bounds are continuous functions defined over $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$, and we refer to them as $\lambda_{ub}(a,d)$ and $\lambda_{lb}(a,d)$ respectively.*

Proof. We know that the region respects inequalities (23) and (24) and that $\mathtt{a}_{\max}(d)$ is an upper bound on a for all $d \in [0,1]$. Using this information, we can rearrange the inequalities to get $\lambda_{ub}(a,d)$ and $\lambda_{lb}(a,d)$. The result for both $\lambda_{ub}(a,d)$ and $\lambda_{lb}(a,d)$ is a ratio of polynomials in terms of the parameters a and d. As the zeros of the denominators lie outside of the domains $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$, we can conclude that both $\lambda_{ub}(a,d)$ and $\lambda_{lb}(a,d)$ are continuous and defined over our entire domain. These bounds define the interval, $[\lambda_{lb}(a,d), \lambda_{ub}(a,d)]$, of λ values for any fixed a and d, which result in $\mathcal{GC}$ being the largest scoring cut. Taken together with the bounds $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$ they make up $\mathcal{R}_{\mathcal{GC}}$. □

Lemma 3.9. *The lower and upper bounds for λ meet at $a = \mathtt{a}_{\max}(d)$, $d \in [0,1]$. That is, $\lambda_{ub}(\mathtt{a}_{\max}(d),d) = \lambda_{lb}(\mathtt{a}_{\max}(d),d)$ for all $0 \leq d \leq 1$. This means that for all $d \in [0,1]$, $\lambda = \lambda_{ub}(\mathtt{a}_{\max}(d),d)$ (identically $\lambda = \lambda_{lb}(\mathtt{a}_{\max}(d),d)$) would score all cuts equally.*

Proof. This can be checked by substituting $\mathtt{a}_{\max}(d)$ into the equations of $\lambda_{ub}(\mathtt{a}_{\max}(d),d)$ and $\lambda_{lb}(\mathtt{a}_{\max}(d),d)$, equating both sides and rearranging. The result is that $\lambda_{ub}(\mathtt{a}_{\max}(d),d) = \lambda_{lb}(\mathtt{a}_{\max}(d),d)$. □

Lemma 3.10. *$\lambda_{ub}(\mathtt{a}_{\max}(d),d)$ (identically: $\lambda_{lb}(\mathtt{a}_{\max}(d),d)$), where $0 \leq d \leq 1$, is a continuous function and has different valued end points.*

Proof. We know from Lemma 3.8 that $\lambda_{ub}(a,d)$ is continuous, and can conclude that $\lambda_{ub}(\mathtt{a}_{\max}(d),d)$ is continuous. The different valued endpoints can be derived by evaluating $\lambda_{ub}(\mathtt{a}_{\max}(0),0)$ and $\lambda_{ub}(\mathtt{a}_{\max}(1),1)$, which have the relation $\lambda_{ub}(\mathtt{a}_{\max}(1),1) > \lambda_{ub}(\mathtt{a}_{\max}(0),0)$. □

Figure 3 visualises the function $\lambda_{ub}(\mathtt{a}_{\max}(d),d)$ (identically $\lambda_{lb}(\mathtt{a}_{\max}(d),d)$) for $0 \leq d \leq 1$. For any $d \in [0,1]$, these functions alongside slight changes to $\mathtt{a}_{\max}(d)$, will be used to generate intervals of λ values, which score $\mathcal{GC}$ the largest and lie between a finite discretisation of $[0,1]$.

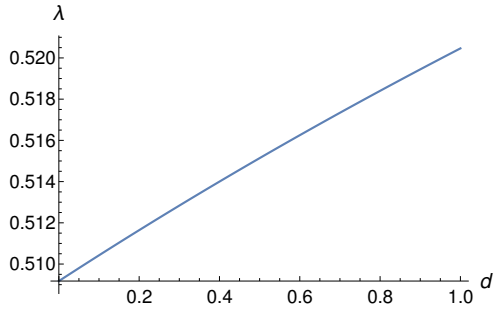

Figure 3: Plot of λ values that scores all cuts equally for all $d \in [0,1]$. That is, $\lambda_{ub}(\mathtt{a}_{\max}(d), d)$, $d \in [0,1]$ (identically: $\lambda_{lb}(\mathtt{a}_{\max}(d), d)$)

Lemma 3.11. *$\lambda_{ub}(a,d)$ - $\lambda_{lb}(a,d) > 0$ for all $0 \leq d \leq 1$ and $0 \leq a < \mathtt{a}_{\max}(d)$. That is, $a = \mathtt{a}_{\max}(d)$ is the only time at which $\lambda_{ub}(a,d) = \lambda_{lb}(a,d)$ for $0 \leq a \leq \mathtt{a}_{\max}(d)$.*

Proof. We can verify this by setting the constraints (23)-(24) to hard equalities, and then solving over the domain $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$. Solving such a system gives the unique solution $a = \mathtt{a}_{\max}(d)$ for $0 \leq d \leq 1$. As $\lambda_{ub}(0,d) > \lambda_{lb}(0,d)$, for all $d \in [0,1]$, and both $\lambda_{ub}(a,d)$ and $\lambda_{lb}(a,d)$ are continuous functions from Lemma 3.8, we can conclude that $\lambda_{ub}(a,d)$ - $\lambda_{lb}(a,d) > 0$ for all $0 \leq d \leq 1$ and $0 \leq a < \mathtt{a}_{\max}(d)$. □

Lemma 3.12. *An interval $[\lambda_{lb}(a',d), \lambda_{ub}(a',d)] \subseteq r_{\mathcal{GC}}(a',d)$ can be constructed, where $\lambda_{ub}(a',d) - \lambda_{lb}(a',d) > 0$ for all $0 \leq d \leq 1$ and $0 \leq a' < \mathtt{a}_{\max}(d)$.*

Proof. We define following function, $\widehat{\mathtt{a}_{\max}}(d,\hat{\epsilon})$, representing $\mathtt{a}_{\max}(d)$ with a shift of $\hat{\epsilon}$:

$$\widehat{\mathtt{a}_{\max}}(d,\hat{\epsilon}) := \mathtt{a}_{\max}(d) - \hat{\epsilon}, \text{ where } \quad 0 \leq d \leq 1, \quad 0 < \hat{\epsilon} \leq \mathtt{a}_{\max}(d) \tag{26}$$

We know from Lemma 3.11 that $a = \mathtt{a}_{\max}(d)$ is the only time at which $\lambda_{lb}(a,d) = \lambda_{ub}(a,d)$ for any $d \in [0,1]$. We also know that $\lambda_{ub}(a,d)$ and $\lambda_{lb}(a,d)$ are defined over all $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$. Therefore the following holds for any $d \in [0,1]$ and $\hat{\epsilon} \in (0, \mathtt{a}_{\max}(d)]$:

$$\lambda_{ub}(\widehat{\mathtt{a}_{\max}}(d,\hat{\epsilon}), d) - \lambda_{lb}(\widehat{\mathtt{a}_{\max}}(d,\hat{\epsilon}), d) > 0$$

Additionally, by the definition of $\mathcal{R}_{\mathcal{GC}}$ from the inequalities (23) - (24), we know that the following interval is connected:

$$\mathbf{I}(\widehat{\mathtt{a}_{\max}}(d,\hat{\epsilon}), d) := [\lambda_{lb}(\widehat{\mathtt{a}_{\max}}(d,\hat{\epsilon}), d), \lambda_{ub}(\widehat{\mathtt{a}_{\max}}(d,\hat{\epsilon}), d)], \text{ where } \quad 0 \leq d \leq 1, \quad 0 < \hat{\epsilon} \leq \mathtt{a}_{\max}(d)$$

We therefore can construct a connected non-empty interval $\mathbf{I}(a',d) \subseteq \mathtt{r}_{\mathcal{GC}}(a',d)$ for all $d \in [0,1]$, where $a' = \widehat{\mathtt{a}_{\max}}(d,\hat{\epsilon})$ and $0 \leq \hat{\epsilon} < \mathtt{a}_{\max}(d)$. □

While we have shown the necessary methods to construct an interval of λ values, $\mathbf{I}(a,d)$, that result in $\mathcal{GC}$ being selected, we have yet to guarantee that at all stages of the solving process, the desired LP optimal solution is taken for all $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$. Specifically, we need to show that the originally optimal point is always $(\frac{-1}{2}, 3, \frac{1}{2})$, that after applying $\mathcal{GC}$ the integer solution $(1,1,0)$ is optimal, and that after applying $\mathcal{ISC}^n$ (or $\mathcal{OPC}^n$) a fractional solution from $\mathcal{ISC}^n_{\mathcal{X}}$ (or $\mathcal{OPC}^n_{\mathcal{X}}$) for all $n \in \mathbb{N}$, is optimal.

Lemma 3.13. *The fractional solution $(\frac{-1}{2}, 3, \frac{1}{2})$ is LP optimal for $\mathsf{P}(a,d)$ for all $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$.*

Proof. This can be done done by substituting all points from $\mathcal{X} \setminus (\frac{-1}{2}, 3, \frac{1}{2})$ into the objective, and then showing that the objective is strictly less when evaluated at $(\frac{-1}{2}, 3, \frac{1}{2})$. This shows that for all $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$:

$$\mathsf{P}(a,d)\big|_{\mathbf{x}=(\frac{-1}{2},3,\frac{1}{2})} < \mathsf{P}(a,d)\big|_{\mathbf{x}=\mathbf{x}'} \quad \forall \mathbf{x}' \in \mathcal{X} \setminus \{(\frac{-1}{2}, 3, \frac{1}{2})\}$$

□

Lemma 3.14. *The integer solution $(1,1,0)$ is LP optimal after applying $\mathcal{GC}$ to $\mathsf{P}(a,d)$ for all $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$.*

Proof. This can be done in an identical fashion to Lemma 3.13. That is, we show that for all $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$:

$$\mathsf{P}(a,d)\big|_{\mathbf{x}=(1,1,0)} < \mathsf{P}(a,d)\big|_{\mathbf{x}=\mathbf{x}'} \quad \forall \mathbf{x}' \in \mathcal{GC}_{\mathcal{X}} \setminus \{(1,1,0)\}$$

□

Lemma 3.15. *Having applied the cut $\mathcal{ISC}^n$ to $\mathsf{P}(a,d)$, a point from $\mathcal{ISC}^n_{\mathcal{X}} \setminus \mathcal{I}_{\mathcal{X}}$ is LP optimal for all $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$.*

Proof. This can be done by showing that for any choice of $a \in [0, \mathtt{a}_{\max}(d)]$ and $d \in [0,1]$, there is at least one point from $\mathcal{ISC}^n_{\mathcal{X}} \setminus \mathcal{I}_{\mathcal{X}}$ at which the objective is strictly less than at all integer points $\mathcal{I}_{\mathcal{X}}$. Specifically, for all $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$:

$$\exists \mathbf{x}' \in \mathcal{ISC}^n_{\mathcal{X}} \setminus \mathcal{I}_{\mathcal{X}} \quad s.t \quad \mathsf{P}(a,d)\big|_{\mathbf{x}=\mathbf{x}'} < \mathsf{P}(a,d)\big|_{\mathbf{x}=\mathbf{x}''} \quad \forall \mathbf{x}'' \in \mathcal{I}_{\mathcal{X}}$$

□

Lemma 3.16. *Having applied the cut $\mathcal{OPC}^n$ to $\mathsf{P}(a,d)$, a point from $\mathcal{OPC}^n_{\mathcal{X}} \setminus \mathcal{I}_{\mathcal{X}}$ is LP optimal for all $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$.*

Proof. This proof follows the same logic as that of Lemma 3.15. □

We can now prove Theorem 3.1 using the Lemmas 3.2 - 3.16 that we have built up throughout this paper.

Theorem 3.1. *Given a finite discretisation of λ, an infinite family of MILP instances together with an infinite amount of family-wide valid cuts can be constructed. Using a pure cutting plane approach and applying a single cut per selection round, the infinite family of instances do not solve to optimality for any value in the discretisation, but do solve to optimality for an infinite amount of alternative λ values.*

Proof. From Lemmas 3.2 - 3.5, we know the exact vertex set of our feasible region at each stage of the solving process, as well as the exact set of cuts at each round. Furthermore, as at each round only the RHS value for each proposed cut changes, the scoring of the cuts at each new round remains constant, and we can therefore completely describe the three scenarios of how cuts would be added. Let $\mathcal{S}$ be the set containing all cuts added during the solution process to an instance $\mathsf{P}(a,d)$, where $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$:

$$\mathcal{S} := \begin{cases} \{\mathcal{ISC}^n : \forall n \in \mathbb{N}\}, & \text{if } \lambda > \lambda_{ub}(a,d) \\ \{\mathcal{GC}\}, & \text{if } \lambda_{lb}(a,d) \leq \lambda \leq \lambda_{ub}(a,d) \\ \{\mathcal{OPC}^n : \forall n \in \mathbb{N}\}, & \text{if } \lambda < \lambda_{lb}(a,d) \end{cases} \tag{27}$$

From Lemma 3.6 we know the sufficient conditions for a λ value that results in $\mathcal{GC}$ being scored at least as well as the other cuts. Lemmas 3.7 - 3.11 show how these sufficient conditions can be used to construct the region $\mathcal{R}_{\mathcal{GC}}$. Moreover, they show that $\mathcal{R}_{\mathcal{GC}}$ is bounded, and that $a = \mathtt{a}_{\max}(d)$, for all $d \in [0,1]$, is the only time at which the following occurs:

$$\lambda_{lb}(a,d) = \lambda_{ub}(a,d) \quad \forall d \in [0,1]$$

We therefore conclude that $\mathcal{R}_{\mathcal{GC}}$ is connected. We know from Lemma 3.10 that both $\lambda_{ub}(\mathtt{a}_{\max}(d),d)$ and $\lambda_{lb}(\mathtt{a}_{\max}(d),d)$ are continuous, where $d \in [0,1]$, and that $\lambda_{ub}(\mathtt{a}_{\max}(1),1) > \lambda_{ub}(\mathtt{a}_{\max}(0),0)$. From the intermediate value theorem, we then know the following:

$$\forall \lambda \in [\lambda_{ub}(\mathtt{a}_{\max}(0),0),\ \lambda_{ub}(\mathtt{a}_{\max}(1),1)], \quad \exists d' \ s.t \ \lambda = \lambda_{ub}(\mathtt{a}_{\max}(d'),d') \tag{28}$$

From Lemma 3.12 we have shown an explicit way to construct an interval $\mathbf{I}(a,d) \subseteq \mathtt{r}_{\mathcal{GC}}(a,d)$ for all $(a,d) \in [0, \mathtt{a}_{\max}(d)) \times [0,1]$. We can therefore construct the following intervals:

$$\mathbf{I}(\widehat{\mathtt{a}_{\max}}(d',\hat{\epsilon}),d') = [\lambda_{lb}(\widehat{\mathtt{a}_{\max}}(d',\hat{\epsilon}),d'), \lambda_{ub}(\widehat{\mathtt{a}_{\max}}(d',\hat{\epsilon}),d')], \text{ where } \quad 0 \leq d' \leq 1, \quad 0 < \hat{\epsilon} \leq \mathtt{a}_{\max}(d) \tag{29}$$

These intervals can be arbitrarily small as $\hat{\epsilon}$ can be arbitrarily small. Moreover, as d' values that satisfy (28) can be used, and $\lambda_{lb}(a,d)$, $\lambda_{ub}(a,d)$, and $\widehat{\mathtt{a}_{\max}}(d,\hat{\epsilon})$ are polynomials, we can generate infinitely many disjoint intervals. We

can therefore conclude that for any finite discretisation of λ, Λ, an interval can be created that contains no values from $\{\lambda_1, \dots, \lambda_{|\Lambda|}\}$, but contains all values of λ for which $\mathsf{P}(a, d)$ solves to optimality.

$$\mathbf{I}(\widehat{\mathtt{a}_{\max}}(d', \hat{\epsilon}), d') \subset \{[0, \lambda_1) \cup (\lambda_1, \lambda_2) \cup \dots \cup (\lambda_{n-1}, \lambda_n) \cup (\lambda_{|\Lambda|}, 1]\}, \text{ where } \quad 0 < \hat{\epsilon} \leq \mathtt{a}_{\max}(d') \tag{30}$$

Finally, Lemmas 3.13 - 3.16 ensure that each stage of the solving process, all cuts are valid for any fractional feasible LP optimal solution for all $\mathsf{P}(a, d)$, where $0 \leq d \leq 1$ and $0 \leq a \leq \mathtt{a}_{\max}(d)$. Moreover, the Lemmas guarantee that only after applying $\mathcal{GC}$ is an integer optimal solution found.

We therefore have shown how fixing a global value of λ to a constant for use in the MILP solving process while disregarding all instance information can result in infinitely worse performance for infinitely many instances. □

Corollary 3.1.1. *There exists an infinite family of MILP instances together with an infinite amount of family-wide valid cuts, which do not solve to integer optimality for any λ when using a pure cutting plane approach and applying a single cut per selection round.*

Proof. To show this we take the following function, where $0 \leq d \leq 1$:

$$\tilde{a_{max}}(d, \tilde{\epsilon}) := \mathtt{a}_{\max}(d) + \tilde{\epsilon}, \quad 0 < \tilde{\epsilon} \leq 0.1$$

Any such value of a retrieved from this function will lie outside of $\mathcal{R}_{\mathcal{GC}}$ for all $0 \leq d \leq 1$. There thus would exist no λ value that results in finite termination, as $\mathcal{GC}$ is never scored at least as high as the other cuts.

Similar to the proof of Theorem 3.1, we need to ensure that the LP optimal point at all times during the solving process is appropriate, and that the same integer optimal point stays integer optimal for all $0 \leq d \leq 1$ and $\mathtt{a}_{\max}(d) < a \leq \tilde{a_{max}}(d, \tilde{\epsilon})$. We therefore redo the proofs of Lemmas 3.13 - 3.16 but change the range of values of a. □

4 Cut Selection in SCIP

Until now we have motivated adaptive cut selection in a theoretical manner, by simulating poor performance of fixed cut selector rules in a pure cutting approach. Using this motivation, we now present results of how parameters of a cut selection scoring rule can be learnt, and made to adapt with the input instance. We begin with an introduction to cut selection in SCIP [18].

The official SCIP cut scoring rule (31) that has been used since SCIP 6.0 and is used for our experiments in Section 6, is defined as:

$$\begin{aligned} \mathtt{cut\_score}(\boldsymbol{\lambda}, \boldsymbol{\alpha}, \mathbf{c}, \mathbf{x}^{LP}, \hat{\mathbf{x}}) := \lambda_1 * \mathtt{eff}(\boldsymbol{\alpha}, \mathbf{x}^{LP}) + \lambda_2 * \mathtt{dcd}(\boldsymbol{\alpha}, \mathbf{x}^{LP}, \hat{\mathbf{x}}) + \lambda_3 * \mathtt{isp}(\boldsymbol{\alpha}) + \lambda_4 * \mathtt{obp}(\boldsymbol{\alpha}, \mathbf{c}) \\ \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 = 1, \quad \lambda_i \geq 0 \quad \forall i \in \{1, 2, 3, 4\}, \quad \boldsymbol{\lambda} = [\lambda_1, \lambda_2, \lambda_3, \lambda_4] \end{aligned} \tag{31}$$

The measures integer support (`isp`) and objective parallelism (`obp`) are defined in (5) and (6). Using the general MILP definition (1), letting $\mathbf{x}^{LP}$ be the LP optimal solution of the current relaxation, and $\hat{\mathbf{x}}$ be the current best incumbent solution, we define the cut measures efficacy (`eff`) and directed cutoff distance (`dcd`) as follows:

$$\mathtt{eff}(\boldsymbol{\alpha}, \mathbf{x}^{LP}) := \frac{\sum_{i=1}^{n} \alpha_i x_i^{LP} - \alpha_0}{\sqrt{\alpha_1^2 + \dots + \alpha_n^2}} \tag{32}$$

$$\mathtt{dcd}(\boldsymbol{\alpha}, \mathbf{x}^{LP}, \hat{\mathbf{x}}) := \frac{\sum_{i=1}^{n} \alpha_i x_i^{LP} - \alpha_0}{|\sum_{i=1}^{n} \alpha_i y_i|}, \text{ where } \quad y = \frac{\hat{\mathbf{x}} - \mathbf{x}^{LP}}{||\hat{\mathbf{x}} - \mathbf{x}^{LP}||} \tag{33}$$

We note that in SCIP the cut selector does not control how many times it itself is called, which candidate cuts are provided, nor the maximum amount of cuts we can apply each round. We reiterate that each call to the selection subroutine is called an *iteration* or *round*. Algorithm 1 gives an outline of the SCIP cut selection rule.

The SCIP cut selector rule in Algorithm 1 still follows the major principles presented in [1]. Cuts are greedily added by the largest score according to the scoring rule (31). After a cut is added, all other candidate cuts that are deemed too parallel to the added cut are filtered out and can no longer be added to the formulation this round. Forced cuts, which are always added to the formulation, prefilter all candidate cuts for parallelism, and are most commonly one-dimensional cuts or user defined cuts. We note that Algorithm 1 is a summarised version of the true algorithm, and has abstracted some procedures.

Motivated by work from this paper, users can now define their own cut selection algorithms and include them in SCIP since SCIP 8.0 [18]. We hope that this leads to additional research about machine learning cut selection algorithms in modern MILP solvers.

Algorithm 1: SCIP Default Cut Selector (Summarised)

Input : *cuts* $\in \mathbb{R}^{s_1 \times n}$, *forced_cuts* $\in \mathbb{R}^{s_2 \times n}$, *max_cuts* $\in \mathbb{Z}_{\geq 0}$, $(s_1, s_2) \in \mathbb{Z}^2_{\geq 0}$
Return: Sorted array of selected cuts, the amount of cuts selected

n_cuts $\leftarrow s_1$ `// Size of cuts array`
for *forced_cut in forced_cuts* **do**
  cuts, n_cuts $\leftarrow$ remove cuts from *cuts* too parallel to *forced_cut*
end
n_selected_cuts $\leftarrow 0$
selected_cuts $\leftarrow \emptyset$
while *n_cuts* > 0 *and max_cuts* $>$ *n_selected_cuts* **do**
  `// Scoring done with (31). If no primal, efficacy replaces cutoff distance`
  best_cut $\leftarrow$ select highest scoring cut remaining in *cuts*
  selected_cuts $\leftarrow$ *selected_cuts* $\cup$ *best_cut*
  n_selected_cuts $\leftarrow$ *n_selected_cuts* + 1
  cuts, n_cuts $\leftarrow$ remove cuts from *cuts* too parallel to *best_cut*
end
return *forced_cuts* $\cup$ *selected_cuts*, s_2 + *n_selected_cuts*

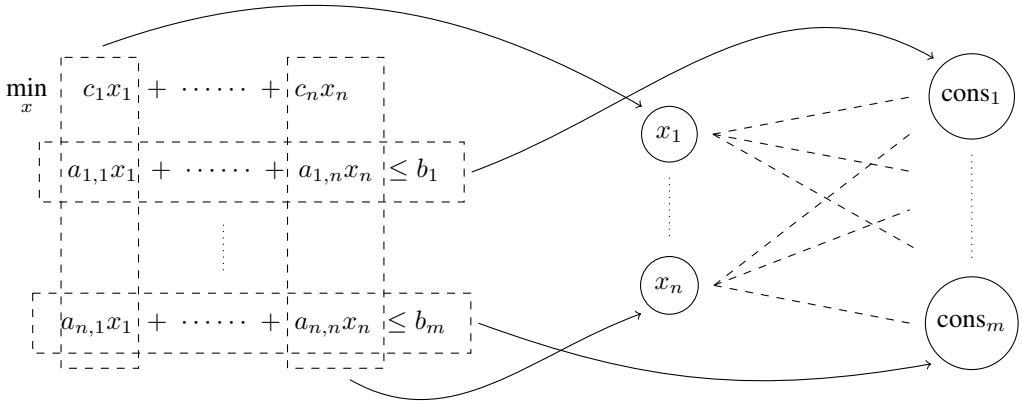

Figure 4: A visualisation of the variable-constraint bipartite graph construction from a MILP.

5 Problem Representation and Solution Architecture

We now present our approach for learning cut selector parameters for MILPs. In Subsection 5.1 we describe our encoding of a general MILP instance into a bipartire graph. Subsection 5.2 introduces a framework for posing cut selection parameter choices as a RL problem, with Subsection 5.3 describing the GCNN architecture used as our policy network. Subsection 5.4 outlines the training method to update our policy network.

5.1 Problem representation as a graph

The current standard for deep learning representation of a general MILP instance is the constraint-variable bipartite graph as described in [8]. Some extensions to this design have been proposed, see [19], as well as alternative non graph embeddings, see [9] and [20]. We use the embedding as introduced in [8] and the accompanying graph convolutional neural network (GCNN) design, albeit with the removal of all LP solution specific features and a different interpretation of the output. The construction process for the bipartite graph can be seen in Figure 4.

The bipartite graph representation can be written as $G = \{\mathbf{V}, \mathbf{C}, \mathbf{E}\} \in \mathcal{G}$, where $\mathcal{G}$ is the set of all bipartite graph representations of MILP instances. $\mathbf{V} \in \mathbb{R}^{n \times 7}$ is the feature matrix of nodes on one side of the graph, which correspond one-to-one with the variables (columns) in the MILP. $\mathbf{C} \in \mathbb{R}^{m \times 2}$ is the feature matrix of nodes on the other side, and correspond one-to-one with the constraints (rows) in the MILP. An edge $(i, j) \in \mathbf{E}$ exists when the variable represented by x_i has non-zero coefficient in constraint cons_j, where $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. We abuse notation slightly and say that $\mathbf{E} \in \mathbb{R}^{m \times n \times 1}$, where $\mathbf{E}$ is the edge feature tensor. Note that this representation

Tensor	Features	Value Range
$\mathbf{V}$	Normalised objective coefficient	[-1, 1]
	Normalised lower bound | upper bound	$\{-2, [-1, 1], 2\}^2$
	Type: binary | integer | continuous | implicit integer	one-hot encoding
$\mathbf{C}$	Absolute objective parallelism (cosine similarity)	[0,1]
	Normalised RHS per constraint	[-1, 1]
$\mathbf{E}$	Normalised coefficients per constraint	[-1, 1]

Table 1: Feature descriptions of variable (column) feature matrix $\mathbf{V}$, constraint (row) feature matrix $\mathbf{C}$, and edge feature tensor $\mathbf{E}$.

would be extended once cuts are added as they then become constraints. The exact set of features can be seen in Table 1

5.2 Reinforcement Learning Framework

We formulate our problem as a single step Markov decision process. The initial state of our environment is $s_0 = G^0 = G$. An agent takes an action $a_0 \in \mathbb{R}^4$, resulting in an instant reward $\mathtt{r}(s_0, a_0) \in \mathbb{R}$, and deterministically transitions to a terminal state $s_1 = G^{N_r}$, $N_r \in \mathbb{Z}$. The action taken, a_0, is dictated by a policy $\pi_\theta(a_0|s_0)$ that maps any initial state to a distribution over our action space, i.e. $a_0 \sim \pi_\theta(\cdot|s_0)$.

The MILP solver in this framework is our environment, and the cut selector our agent. Let N_r be the number of paired separation and cut selection rounds we wish to apply, and $G^i \in \mathcal{G}$ be the bipartite graph representation of $G \in \mathcal{G}$ after i rounds have been applied. The action $a_0 \in \mathbb{R}^4$ is the choice of cut selector parameters $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ followed by N_r paired separation rounds. Applying action a_0 to state s_0 results in a deterministic transition to $s_1 = G^{N_r}$, defined by the function $\mathtt{f} : \mathcal{G} \times \mathbb{R}^4 \to \mathcal{G}$.

The baseline function, $\mathtt{b}(s_0) : \mathcal{G} \to \mathbb{R}$, maps an initial state s_0 to the gap of the LP solution of $\mathtt{f}(s_0, a') \in \mathcal{G}$, where the solver is run with standard cut selector parameters, $a' \in \mathbb{R}^4$, and some pre-loaded primal solution. The gap in this experiment can be thought of as a normalised dual bound, as the pre-loaded primal cannot be improved upon without a provable optimal solution itself. The pre-loaded primal also serves to make directed cutoff distance active from the beginning of the solving process. We do note that this is different to the normal solve process and introduces some bias. Let $\mathtt{g}_{a_0}(s_0)$ be the gap of the LP solution of $\mathtt{f}(s_0, a_0)$ if a_0 are the cut selector parameter values used. The reward $\mathtt{r}(s_0, a_0)$ can then be defined as:

$$\mathtt{r}(s_0, a_0) := \frac{\mathtt{b}(s_0) - \mathtt{g}_{a_0}(s_0)}{|\mathtt{b}(s_0)| + 10^{-8}}$$

Let $(s_0, a_0, s_1) \in \mathcal{G} \times \mathbb{R}^4 \times \mathcal{G}$ be a trajectory, also called a roll out in the literature. The goal of reinforcement learning is to maximise the expected reward over all trajectories. That is, we want to find θ that parameterises:

$$\underset{\theta}{\operatorname{argmax}} \underset{(s_0,a_0,s_1)\sim\pi_\theta}{\mathbb{E}} [\mathtt{r}(s_0, a_0, s_1)] = \underset{\theta}{\operatorname{argmax}} \int_{s_1 \in \mathcal{G}} \int_{(s_0,a_0)\in \mathtt{f}^{-1}(s_1)} p(s_0)\pi_\theta(a_0|s_0)\mathtt{r}(s_0, a_0)\, ds_0 da_0 ds_1 \tag{34}$$

Here, $p(s_0)$ is the density function on instances $s \in \mathcal{G}$ evaluated at $s = s_0$. The pre-image $\mathtt{f}^{-1}(s_1) : \mathcal{G} \to \mathbb{R}^4 \times \mathcal{G}$ is defined as:

$$\mathtt{f}^{-1}(\mathcal{G}') := \{(s_0, a_0) \in \mathcal{G} \times \mathbb{R}^4 \mid \mathtt{f}(s_0, a_0) \in \mathcal{G}'\}, \quad \mathcal{G}' \subseteq \mathcal{G}$$

We note that equation (34) varies from the standard definition as seen in [5], and those presented in similar research [8, 13], as our action space is continuous. Additionally, as the set $\mathcal{G}$ is infinite and we do not know the density function $p(s)$, we use sample average approximation, creating a uniform distribution around MIPLIB 2017 [9].

5.3 Policy Architecture

Our policy network, $\pi_\theta(\cdot|s_0 \in \mathcal{G})$, is parameterised as a graph convolutional neural network, and follows the general design as in [8], where θ fully describes the complete set of weights and biases in the GCNN. The changes in design are that we use 32 dimensional convolutions instead of 64 due to our lower dimensional input, and output a 4 dimensional vector as we are interested in cut selector parameters. This technique of using the constraint-variable graph as an embedding for graph neural networks has gained recent popularity, see [21] for an overview of applications in combinatorial optimisation.

Our policy network takes as input the constraint-variable bipartite graph representation $s_0 = \{\mathbf{V}, \mathbf{C}, \mathbf{E}\}$. Two staggered half-convolutions are then applied, with messages being passed from the embedding $\mathbf{V}$ to $\mathbf{C}$ and then

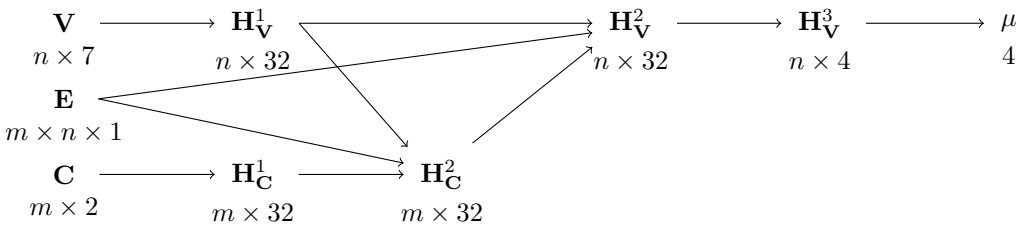

Figure 5: The architecture of policy network $\pi_\theta(a_0|s_0)$. $\mathbf{H}$ represent hidden layers of the network.

back. The result is a bipartite graph with the same topology but new feature matrices. Our policy is then obtained by normalising feature values over all nodes and averaging the result into a vector $\mu \in \mathbb{R}^4$. This vector $\mu \in \mathbb{R}^4$ represents the mean of a multivariate normal distribution, $\mathcal{N}_4(\mu, \gamma I)$, where $\gamma \in \mathbb{R}$. We note that having our policy network only output the mean was a design choice to simplify the learning process, and that our design can be extended to also output γ or additional distribution information. Any sample from the distribution $\mathcal{N}_4(\mu, \gamma I)$ can be considered an action $a_0 \in \mathbb{R}^4$, which represent $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ with the non-negativity constraints relaxed. Figure 5 provides an overview of this architecture.

5.4 Training Method

To train our GCNN we use policy gradient methods, specifically the REINFORCE algorithm with baseline and gaussian exploration, see [5] for an overview. An outline of the algorithm is given in Algorithm 2.

Algorithm 2: Batch REINFORCE

Input: Policy network π_θ, MILP instances $batch$, $n_{\text{samples}} \in \mathbb{N}$, $N_r \in \mathbb{N}$

$\mathcal{L} \leftarrow 0$
for s_0 *in batch* **do**
  $\mu \leftarrow \pi_\theta(\cdot|s_0)$ `// Note that` $\pi_\theta(\cdot|s_0)$ `is technically` $\mathcal{N}_4(\mu, \gamma I)$
  for *i in* $\{1, \ldots, n_{samples}\}$ **do**
    $a_0 \leftarrow$ sample $\mathcal{N}_4(\mu, \gamma I)$
    $s_1 \leftarrow$ Apply N_r rounds of separation and cut selection to s_0
    $r \leftarrow$ Relative dual bound improvement of s_1 to some baseline
    $\mathcal{L} \leftarrow \mathcal{L} + (-r \times log(\pi_\theta(a_0|s_0)))$ `// Use log probability for numeric stability`
  end
end
$\theta \leftarrow \theta + \nabla_\theta \mathcal{L}$ `// We use the Adam update rule in practice [22]`

Algorithm 2 is used to update the weights and biases, θ, of our GCNN, $\pi_\theta(\cdot|s_0 \in \mathcal{G})$. It does this for a batch of instances by minimising $\mathcal{L}$, referred to as the loss function, see [23]. We used default parameter settings in the Adam update rule, aside from a learning rate with value $1e^{-5}$.

6 Experiments

We use MIPLIB 2017[5] [9] as our data set, and from now will simply refer to it as MIPLIB. For subsections 6.1 - 6.4, we run experiments on presolved instances. Each individual run on a presolved instance consists of a single round of presolve (to remove fixed variables), then solving the root node, using 50 separation rounds with a limit of 10 cuts per round, and a pre-loaded primal that is the best solution found within 600s when solved under standard conditions. Propagation and heuristics for these runs are disabled, and after applying SCIP's default cut selector in Algorithm 1, the highest scoring cuts that were filtered for parallelism are added until the 10 cuts per round limit is reached, or no more cuts exist. We believe these conditions best represent a sandbox environment, which allows cut selection to be the largest influence on solver performance. Additionally, all results are obtained by averaging results over the SCIP random seeds $\{1, 2, 3\}$. All code for reproducing experiments can be found at https://github.com/Opt-Mucca/Adaptive-Cutsel-MILP.

For all experiments SCIP 7.0.2 [7] is used, with PySCIPOpt [17] as the API, and Gurobi 9.1.2 [24] as the LP solver. PyTorch 1.7.0 [25] and PyTorch-Geometric 2.0.1 [26] are used to model the GCNN. All experiments are run on a cluster equipped with Intel Xeon E5-2670 v2 CPUs with 2.50GHz and 64GB main memory.

[5] MIPLIB 2017 – The Mixed Integer Programming Library https://miplib.zib.de/.

Criteria	% of instances removed
Tags: *feasibility*, *numerics*, *infeasible*, *no solution*	4.5%, 17.4%, 2.8%, 0.9%
Unbounded objective, MIPLIB solution unavailable	0.9%, 2.6%
Presolve longer than 300s under default conditions	4.8%
No feasible solution found in 600s under default conditions	15.3%
Solved to optimality at root	9.9%
Too few cuts applied (< 250)	9.3%
Root solve longer than 20s	14.7%

Table 2: Percentage of instances removed from MIPLIB

For instance selection we discard instances from MIPLIB that satisfy any of the criteria in Table 2. We believe that these conditions focus on instances where a good selection strategy of cuts can improve the dual bound in a reasonable amount of time. We note that improving the dual bound is a proxy for overall solver performance, and does not necessarily result in improved solution time.

6.1 Lower bounding potential improvement

To begin our experiments, we first perform a grid search to give a lower bound on the potential improvement that adaptive cut selection can provide. Given our training subset of MIPLIB, we generate all parameter scenarios satisfying the following condition:

$$\sum_{i=1}^{4} \lambda_i = 1, \quad \text{where} \quad \lambda_i = \frac{\beta_i}{10}, \quad \beta_i \in \mathbb{N}, \quad \forall i \in \{1, 2, 3, 4\}$$

Recall that λ_i for all $i \in \{1, 2, 3, 4\}$ are respectively multipliers of the cut scoring measures directed cutoff distance (`dcd`), efficacy (`eff`), integer support (`isp`), and objective parallelism (`obp`).

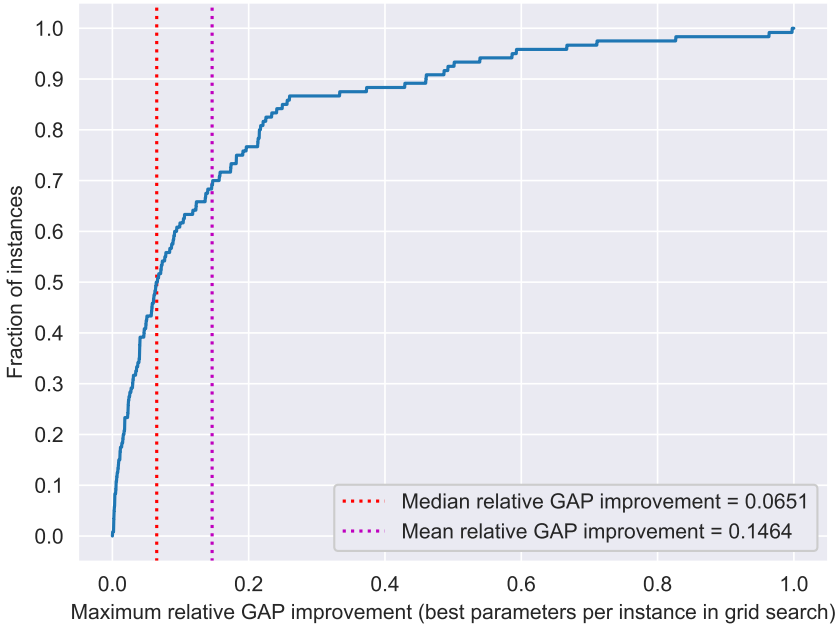

Figure 6: Relative gap improvement from best choice parameters in Experiment 6.1.

We solve the root node for all instances and parameter choices, and store the cut selector parameters that result in the smallest gap, as well as their relative gap improvement compared to the default cut selector parameter values. We remove all instances where the worst case parameter choice compared to the best case parameter choice differ by a relative gap performance of less than 0.1%. Additionally, we remove instances where a quarter or more of the

parameter choices result in the identical best gap performance. These removals are made due to the sparse learning opportunities provided by the instances, as the best case performance is minimally different from the worst, or the best case performance is too common. This results in an additional 5.3% and 0.1% of instances being removed, leaving 120 (11.3%) instances remaining.

We conclude from the results presented in Figure 6 that there exists notable amounts of improvement potential per instance from better cut selection rules. Specifically, we observe that the median instance can have a relative gap improvement of 6.5% when only considering at most 500 cuts (50 rounds of 10 cuts), with this potential improvement being only a lower bound as the results come from a grid search of the parameter space. Instance specific results are available in Appendix B.

We draw attention to the interesting results on λ_1 (multiplier of `dcd`) in Table 3. These results summarise that cutoff distance is on average less important than the other measures in the context of cut scoring rule (31), which is also reflected in the default settings of SCIP 8.0. These are aggregated results, however, and we note that there exist instances where cutoff distance improves performance provided primal solutions of a reasonable quality exist.

Parameter	Mean	Median	Std Deviation
λ_1 (`dcd`)	0.154	0.075	0.209
λ_2 (`eff`)	0.304	0.200	0.273
λ_3 (`isp`)	0.252	0.200	0.262
λ_4 (`obp`)	0.289	0.200	0.270

Table 3: Statistics of best choice parameters per instance in Experiment 6.1.

6.2 Random Seed Initialisation

Let θ_i be the initialised weights and biases using random seed i, where $i \in \mathbb{N}$. To minimise the bias of our initialised policy with respect to $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \lambda_3, \lambda_4)$, the random seed that satisfies (35) is used throughout our experiments.

$$\underset{i \in \{0,\dots,999\}}{\operatorname{argmin}} \sum_{s_0} \|\mathbb{E}[\pi_{\theta_i}(\cdot|s_0)] - [\frac{1}{4}, \frac{1}{4}, \frac{1}{4}, \frac{1}{4}]\|_1 \tag{35}$$

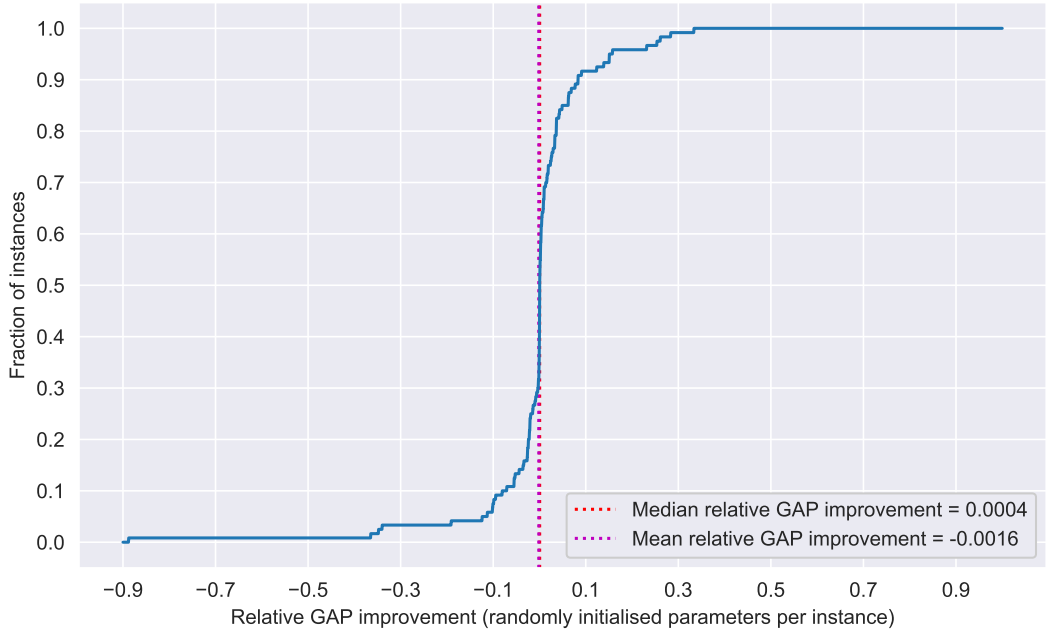

Figure 7: Relative gap improvement from random initialised parameters in Experiment 6.2.

The performance of the randomly initialised GCNN can be seen in Figure 7 and Table 4, with instance specific results available in Appendix B.

Parameter	Mean	Median	Std Deviation
λ_1 (`dcd`)	0.265	0.258	0.023
λ_2 (`eff`)	0.279	0.280	0.019
λ_3 (`isp`)	0.199	0.213	0.035
λ_4 (`obp`)	0.257	0.254	0.034

Table 4: Statistics of random initialised parameters per instance in Experiment 6.2.

6.3 Learning Capabilities

Before we attempt to generalise our result, we first verify that our RL framework, policy architecture, and training method, are capable of learning good cut selector parameters. To do so, we run 500 iterations of Algorithm 2 on each instance individually, with n_{samples} set to 20. γ of the multivariate normal distribution, $\mathcal{N}_4(\mu, \gamma I)$, is defined by the following, where n_{epochs} is the total amount of iterations of Algorithm 2 and i_{epoch} is the current iteration:

$$\gamma(i_{\text{epoch}}, n_{\text{epochs}}) := 0.01 - \frac{0.009 * i_{\text{epoch}}}{n_{\text{epochs}}}, \quad i_{\text{epoch}}, n_{\text{epochs}} \in \mathbb{N}, \quad i_{\text{epoch}} \leq n_{\text{epochs}} \tag{36}$$

We note that γ represents one of many opportunities, such as the GCNN structural design and training algorithm, where a substantial amount of additional effort could be invested to (over)tune the learning experiment. After training GCNNs individually for each instance, we observe a median relative gap improvement of 2.18% in Figure 8 compared to the 0.04% improvement of the random initialisation in Figure 7. Despite this performance by overfitting, the grid search in Subsection 6.1 produced considerably better results. Nevertheless, we conclude from this experiment that our RL framework, policy architecture, and training method are capable of learning good cut selector parameters, albeit not globally optimal parameters. Instance specific results are available in Appendix B.

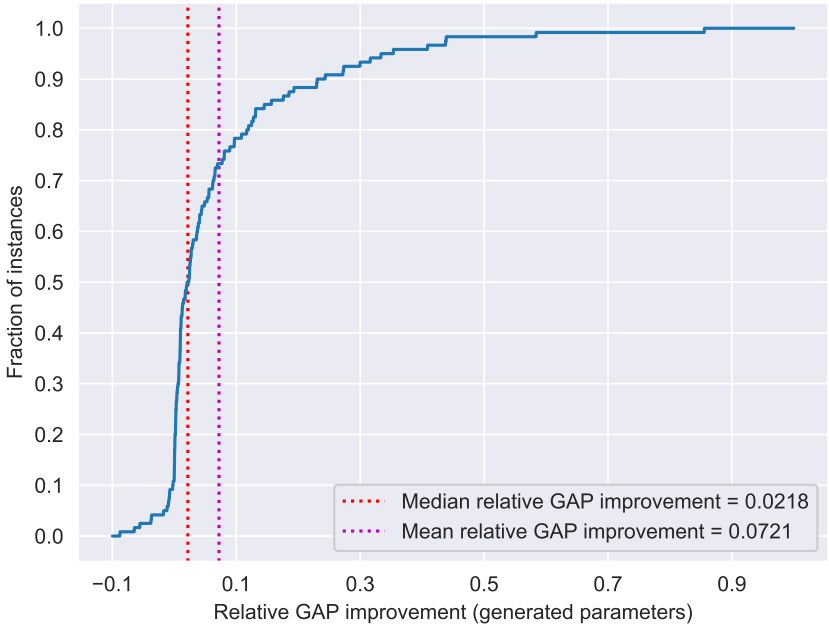

Figure 8: Relative gap improvement from generated parameters in Experiment 6.3.

Parameter	Mean	Median	Std Deviation
λ_1 (`dcd`)	0.252	0.244	0.079
λ_2 (`eff`)	0.308	0.298	0.148
λ_3 (`isp`)	0.192	0.202	0.139
λ_4 (`obp`)	0.249	0.234	0.144

Table 5: Statistics of generated parameters per instance in Experiment 6.3.

6.4 Generalisation Capabilities

We now show the performance of our RL framework over MIPLIB itself instead of over individual instances. To do so, we run 10000 iterations of Algorithm 2 (500 epochs), with n_{samples} set to 20, γ defined as in (36), and allocate 5% of remaining MIPLIB instances per batch.

The randomly initialised GCNN had a median relative gap improvement of 0.004% as seen in Figure 7 compared to the 0% of our MIPLIB trained GCNN as seen in Figure 9. From this we conclude that our trained GCNN failed to generalise individual instance performance onto MIPLIB as a whole.

As in Experiments 6.1 and 6.3, the multiplier λ_2 (multiplier of `eff`) was on average found to be the largest valued. We believe that this supports the common understanding in the MILP community that efficacy is an important measure of cut effectiveness. The standard deviations of all λ's were lower than those in Experiments 6.1 and 6.3, which we believe indicates that our technique was converging to a fixed cut selection scoring rule. Such a convergence would further highlight the inability of our design to generalise its success from individual instances in Experiment 6.3 to an approximation of the MILP instance space. For specific instance results, see Appendix B.

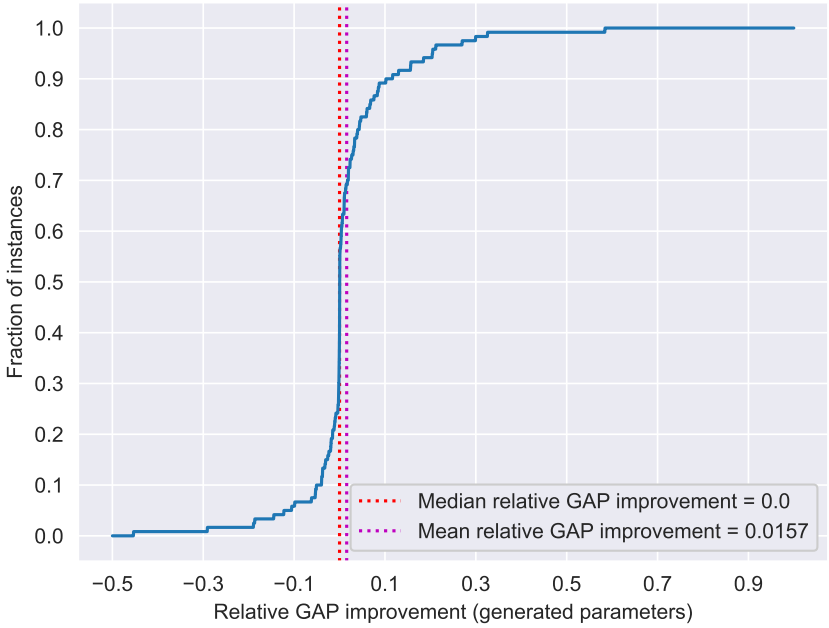

Figure 9: Relative gap improvement from generated parameters in Experiment 6.4

Parameter	Mean	Median	Std Deviation
λ_1 (`dcd`)	0.197	0.229	0.074
λ_2 (`eff`)	0.398	0.414	0.081
λ_3 (`isp`)	0.200	0.202	0.104
λ_4 (`obp`)	0.205	0.157	0.128

Table 6: Statistics of generated parameters per instance in Experiment 6.4

7 Conclusion

We presented a parametric family of MILPs together with infinitely many family-wide valid cuts. We showed for a specific cut selection rule, that any finite grid search of the parameter space will always miss all parameter values, which select integer optimal inducing cuts in an infinite amount of our instances. We then presented a reinforcement learning framework for learning cut selection parameters, and phrased cut selection in MILP as a Markov decision process. By representing MILP instances as a bipartite graph, we used policy gradient methods to train a graph convolutional neural network. The framework generates good performing, albeit sub-optimal, parameter values for

SCIP's default cut scoring rule when trained on individual instances, however, fails to generalise its performance to MIPLIB 2017 as a whole.

Results from our grid search experiments showed that there is a large amount of potential improvements to be made in adaptive cut selection, with a median relative gap improvement of 6.5% with only 50 rounds of 10 cuts. This work also implemented a new cut selector plugin for SCIP, which enables future research via easy inclusion of custom cut selection algorithms in a modern MILP solver.

Acknowledgements

The work for this article has been conducted in the Research Campus MODAL funded by the German Federal Ministry of Education and Research (BMBF) (fund numbers 05M14ZAM, 05M20ZBM). The described research activities are funded by the Federal Ministry for Economic Affairs and Energy within the project UNSEEN (ID: 03EI1004-C).

References

[1] Tobias Achterberg. *Constraint integer programming*. PhD thesis, TU Berlin, 2007.

[2] Tobias Achterberg and Roland Wunderling. Mixed integer programming: Analyzing 12 years of progress. In *Facets of combinatorial optimization*, pages 449–481. Springer, 2013.

[3] Hugues Marchand, Alexander Martin, Robert Weismantel, and Laurence Wolsey. Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics*, 123(1-3):397–446, 2002.

[4] Franz Wesselmann and U Stuhl. Implementing cutting plane management and selection techniques. Technical report, Technical report, University of Paderborn, 2012.

[5] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[6] Maria-Florina Balcan, Travis Dick, Tuomas Sandholm, and Ellen Vitercik. Learning to branch. In *International conference on machine learning*, pages 344–353. PMLR, 2018.

[7] Gerald Gamrath, Daniel Anderson, Ksenia Bestuzheva, Wei-Kun Chen, Leon Eifler, Maxime Gasse, Patrick Gemander, Ambros Gleixner, Leona Gottwald, Katrin Halbig, Gregor Hendel, Christopher Hojny, Thorsten Koch, Pierre Le Bodic, Stephen J. Maher, Frederic Matter, Matthias Miltenberger, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Christine Tawfik, Stefan Vigerske, Fabian Wegscheider, Dieter Weninger, and Jakob Witzig. The SCIP Optimization Suite 7.0. ZIB-Report 20-10, Zuse Institute Berlin, March 2020.

[8] Maxime Gasse, Didier Chételat, Nicola Ferroni, Laurent Charlin, and Andrea Lodi. Exact combinatorial optimization with graph convolutional neural networks. *arXiv preprint arXiv:1906.01629*, 2019.

[9] Ambros Gleixner, Gregor Hendel, Gerald Gamrath, Tobias Achterberg, Michael Bastubbe, Timo Berthold, Philipp Christophel, Kati Jarck, Thorsten Koch, Jeff Linderoth, et al. Miplib 2017: data-driven compilation of the 6th mixed-integer programming library. *Mathematical Programming Computation*, pages 1–48, 2021.

[10] Giuseppe Andreello, Alberto Caprara, and Matteo Fischetti. Embedding $\{0, 1/2\}$-cuts in a branch-and-cut framework: A computational study. *INFORMS Journal on Computing*, 19(2):229–238, 2007.

[11] Santanu S Dey and Marco Molinaro. Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170(1):237–266, 2018.

[12] Vinod Nair, Sergey Bartunov, Felix Gimeno, Ingrid von Glehn, Pawel Lichocki, Ivan Lobov, Brendan O'Donoghue, Nicolas Sonnerat, Christian Tjandraatmadja, Pengming Wang, et al. Solving mixed integer programs using neural networks. *arXiv preprint arXiv:2012.13349*, 2020.

[13] Yunhao Tang, Shipra Agrawal, and Yuri Faenza. Reinforcement learning for integer programming: Learning to cut. In *International Conference on Machine Learning*, pages 9367–9376. PMLR, 2020.

[14] Radu Baltean-Lugojan, Pierre Bonami, Ruth Misener, and Andrea Tramontani. Scoring positive semidefinite cutting planes for quadratic optimization via trained neural networks. *optimization-online preprint 2018/11/6943*, 2019.

[15] Zeren Huang, Kerong Wang, Furui Liu, Hui-ling Zhen, Weinan Zhang, Mingxuan Yuan, Jianye Hao, Yong Yu, and Jun Wang. Learning to select cuts for efficient mixed-integer programming. *arXiv preprint arXiv:2105.13645*, 2021.

[16] Wolfram Research, Inc. Mathematica, Version 12.2. Champaign, IL, 2020.

[17] Stephen Maher, Matthias Miltenberger, Joao Pedro Pedroso, Daniel Rehfeldt, Robert Schwarz, and Felipe Serrano. Pyscipopt: Mathematical programming in python with the scip optimization suite. In *International Congress on Mathematical Software*, pages 301–307. Springer, 2016.

[18] Ksenia Bestuzheva, Mathieu Besançon, Wei-Kun Chen, Antonia Chmiela, Tim Donkiewicz, Jasper van Doornmalen, Leon Eifler, Oliver Gaul, Gerald Gamrath, Ambros Gleixner, Leona Gottwald, Christoph Graczyk, Katrin Halbig, Alexander Hoen, Christopher Hojny, Rolf van der Hulst, Thorsten Koch, Marco Lübbecke, Stephen J. Maher, Frederic Matter, Erik Mühmer, Benjamin Müller, Marc E. Pfetsch, Daniel Rehfeldt, Steffan Schlein, Franziska Schlösser, Felipe Serrano, Yuji Shinano, Boro Sofranac, Mark Turner, Stefan Vigerske, Fabian Wegscheider, Philipp Wellner, Dieter Weninger, and Jakob Witzig. The scip optimization suite 8.0, 2021.

[19] Jian-Ya Ding, Chao Zhang, Lei Shen, Shengyin Li, Bing Wang, Yinghui Xu, and Le Song. Accelerating primal solution findings for mixed integer programs based on solution prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1452–1459, 2020.

[20] Zachary Steever, Chase Murray, Junsong Yuan, Mark Karwan, and Marco Lübbecke. An image-based approach to detecting structural similarity among mixed integer programs. *Available at SSRN 3437981*, 2020.

[21] Quentin Cappart, Didier Chételat, Elias Khalil, Andrea Lodi, Christopher Morris, and Petar Veličković. Combinatorial optimization and reasoning with graph neural networks. *arXiv preprint arXiv:2102.09544*, 2021.

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[23] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021.

[25] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[26] Matthias Fey and Jan E. Lenssen. Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

A Functions of Section 3

$$\mathsf{a}_{\max}(d) := \frac{-2680\sqrt{101}d+2020\sqrt{201}d-6767\sqrt{2}-27068\sqrt{101}+22220\sqrt{201}}{6767\sqrt{2}-202\sqrt{201}}$$

$$\lambda_{lb}(a,d)_1 := \sqrt{20301}\sqrt{(a^2+d(d+20)+101)}$$

$$\lambda_{lb}(a,d)_2 := 101a^2+a\big(-20\big(\sqrt{20301}-101\big)d-202\big(\sqrt{20301}-110\big)\big)$$

$$\lambda_{lb}(a,d)_3 := 20d\big(-10\big(\sqrt{20301}-151\big)d-211\sqrt{20301}+31411\big)-22220\sqrt{20301}+3272501$$

$$\lambda_{lb}(a,d)_4 := -606a^2+12a\big(10\big(\sqrt{20301}-101\big)d+101\big(\sqrt{20301}-110\big)\big)$$

$$\lambda_{lb}(a,d)_5 := 120d\big(10\big(\sqrt{20301}-151\big)d+211\sqrt{20301}-31411\big)+606\big(220\sqrt{20301}-32401\big)$$

$$\lambda_{ub}(a,d)_6 := 5555a^2+24a\big(10\big(\sqrt{20301}-101\big)d+101\big(\sqrt{20301}-110\big)\big)$$

$$\lambda_{lb}(a,d)_7 := d\big(\big(2400\sqrt{20301}-355633\big)d+50640\sqrt{20301}-7403300\big)+505\big(528\sqrt{20301}-76409\big)$$

$$\lambda_{lb}(a,d) := \frac{2\big(\lambda_{lb}(a,d)_1\sqrt{\lambda_{lb}(a,d)_2+\lambda_{lb}(a,d)_3}+\lambda_{lb}(a,d)_4+\lambda_{lb}(a,d)_5\big)}{\lambda_{lb}(a,d)_6+\lambda_{lb}(a,d)_7}$$

$$\lambda_{ub}(a,d)_1 := -\big(a^2+d(d+20)+101\big)$$

$$\lambda_{ub}(a,d)_2 := \big(2\sqrt{402}-203\big)a^2+a\big(20\big(\sqrt{402}-2\big)d+222\sqrt{402}-842\big)+20d\big(-10d+\sqrt{402}-220\big)+220\sqrt{402}-24401$$

$$\lambda_{ub}(a,d)_3 := \big(6\sqrt{402}-609\big)a^2+6a\big(10\big(\sqrt{402}-2\big)d+111\sqrt{402}-421\big)+60d\big(-10d+\sqrt{402}-220\big)+660\sqrt{402}-73203$$

$$\lambda_{ub}(a,d)_4 := \big(6\sqrt{402}-475\big)a^2+6a\big(10\big(\sqrt{402}-2\big)d+111\sqrt{402}-421\big)+2d\big(-233d+30\sqrt{402}-5260\big)+660\sqrt{402}-59669$$

$$\lambda_{ub}(a,d) := \frac{\sqrt{402}\sqrt{\lambda_{ub}(a,d)_1\lambda_{ub}(a,d)_2}+\lambda_{ub}(a,d)_3}{\lambda_{ub}(a,d)_4}$$

B Per Instance Results and Statistics of Section 6

Variable Information	Range	Mean	Median
Num. vars	[89, 24500]	4539.20	2597.50
Num. bin. vars	[0, 16360],	2152.99	859.50
Num. int. vars	[0, 5096]	117.03	0
Num. impl. int. vars	[0, 0]	0	0
Num. cont. vars	[0, 24402]	2268.51	417.5
Constraint Information	**Range**	**Mean**	**Median**
Num. cons	[19, 39424]	3634.73	1647
Num. linear cons	[0, 18443]	1850.99	335.5
Num. logicor cons	[0, 17323]	323.68	0
Num. knapsack cons	[0, 288]	10.65	0
Num. setppc cons	[0, 39424]	708.30	0
Num. varbound cons	[0, 14844]	741.12	6.5

Table 7: Instance statistics for Experiments 6.1, 6.2, 6.3, and 6.4

Instance	GAP	λ_1	λ_2	λ_3	λ_4	#BP
2club200v15p5scn	0.01	0.1	0.0	0.8	0.1	8
50v-10	0.15	0.0	0.1	0.0	0.9	1
a1c1s1	0.02	0.1	0.4	0.0	0.5	1
a2c1s1	0.07	0.0	0.6	0.3	0.1	1
app3	0.07	0.1	0.0	0.4	0.5	1
b-ball	0.59	0.3	0.1	0.3	0.3	1
b1c1s1	0.12	0.0	0.9	0.0	0.1	1
b2c1s1	0.06	0.0	0.1	0.1	0.8	1
bab1	0.04	0.1	0.0	0.4	0.5	1
beasleyC1	0.16	0.1	0.1	0.2	0.6	1
beasleyC2	0.04	0.3	0.2	0.0	0.5	1
berlin	0.01	0.4	0.4	0.2	0.0	2
bg512142	0.01	0.0	0.1	0.9	0.0	1
bienst1	0.14	0.1	0.0	0.1	0.8	10
bienst2	0.22	0.0	0.5	0.5	0.0	1
binkar10_1	0.15	0.1	0.0	0.5	0.4	1
bppc8-02	0.54	0.0	0.3	0.0	0.7	1
brasil	0.06	0.0	0.1	0.0	0.9	1
cost266-UUE	0.03	0.0	0.2	0.2	0.6	1
danoint	0.0	0.0	0.7	0.3	0.0	1
dfn-bwin-DBE	0.23	0.0	0.9	0.0	0.1	1
dg012142	0.0	0.0	0.0	0.4	0.6	1
drayage-100-12	0.17	0.3	0.3	0.0	0.4	1
drayage-100-23	0.46	0.1	0.0	0.9	0.0	9
drayage-25-23	0.49	0.0	0.2	0.3	0.5	1
drayage-25-27	0.2	0.2	0.6	0.0	0.2	2
drayage-25-32	0.14	0.2	0.3	0.0	0.5	1
eil33-2	0.0	0.2	0.0	0.5	0.3	2
exp-1-500-5-5	0.03	0.2	0.3	0.5	0.0	1
f2gap201600	0.03	0.0	1.0	0.0	0.0	1
f2gap401600	0.12	0.0	0.2	0.8	0.0	24
f2gap801600	0.17	0.0	0.8	0.1	0.1	2
fhnw-schedule-paira100	0.01	0.0	1.0	0.0	0.0	1
g200x740	0.59	1.0	0.0	0.0	0.0	1
gmu-35-40	0.0	0.0	1.0	0.0	0.0	1
h50x2450	0.06	0.0	0.3	0.1	0.6	1
h80x6320d	0.03	0.4	0.2	0.1	0.3	1
hypothyroid-k1	0.0	0.1	0.1	0.6	0.2	14
ic97_tension	0.11	0.2	0.7	0.0	0.1	1
istanbul-no-cutoff	0.06	0.0	0.1	0.7	0.2	1
k16x240b	0.05	0.2	0.2	0.6	0.0	1
lectsched-5-obj	0.09	0.1	0.1	0.1	0.7	2
lotsize	0.02	0.0	0.0	0.0	1.0	1
map16715-04	0.0	0.2	0.1	0.5	0.2	1
mik-250-20-75-1	0.14	0.0	0.7	0.1	0.2	1
mik-250-20-75-2	0.1	0.0	0.8	0.0	0.2	1
mik-250-20-75-3	0.08	0.1	0.2	0.5	0.2	1
mik-250-20-75-4	0.09	0.3	0.5	0.2	0.0	1
mik-250-20-75-5	0.08	0.4	0.4	0.1	0.1	1
milo-v12-6-r1-58-1	0.02	0.0	0.4	0.0	0.6	1
mkc1	0.83	0.0	0.0	0.8	0.2	1
mtest4ma	0.5	0.5	0.1	0.0	0.4	1
n3700	0.0	0.0	0.0	0.2	0.8	1
n3705	0.01	0.0	0.2	0.8	0.0	1
n3707	0.01	0.1	0.3	0.5	0.1	1
n3709	0.01	0.2	0.2	0.1	0.5	1
n370b	0.01	0.0	0.6	0.2	0.2	1
n5-3	0.19	0.0	0.3	0.3	0.4	1
n6-3	0.16	0.3	0.3	0.3	0.1	1
n7-3	0.22	0.1	0.4	0.5	0.0	1
n9-3	0.09	0.3	0.3	0.1	0.3	1
neos-1423785	0.43	0.2	0.0	0.1	0.7	1
neos-1445738	0.0	0.0	0.5	0.2	0.3	1
neos-1456979	0.04	0.1	0.6	0.0	0.3	3
neos-1593097	0.49	0.1	0.1	0.3	0.5	1
neos-3046601-motu	0.02	0.0	0.7	0.2	0.1	1
neos-3046615-murg	0.02	0.0	0.1	0.9	0.0	1
neos-3072252-nete	0.03	0.0	0.1	0.9	0.0	1
neos-3581454-haast	0.02	0.0	0.5	0.5	0.0	1
neos-3627168-kasai	0.01	0.0	0.7	0.2	0.1	1
neos-4333596-skien	0.26	0.0	0.1	0.0	0.9	1
neos-4343293-stony	0.05	0.4	0.2	0.1	0.3	1
neos-4650160-yukon	0.04	0.4	0.1	0.0	0.5	1
neos-4736745-arroux	0.02	0.1	0.2	0.0	0.7	1
neos-4738912-atrato	0.22	0.0	0.0	1.0	0.0	1
neos-4954672-berkel	0.08	0.0	0.1	0.9	0.0	1
neos-5076235-embley	0.04	0.1	0.0	0.0	0.9	9
neos-5079731-flyers	0.07	0.0	0.0	0.8	0.2	1
neos-5093327-huahum	0.04	0.2	0.4	0.0	0.4	1
neos-5102383-irwell	0.1	0.0	0.0	0.8	0.2	1
neos-5107597-kakapo	0.33	0.0	0.6	0.2	0.2	1
neos-5140963-mincio	0.0	0.6	0.3	0.0	0.1	1
neos-5260764-orauea	0.01	0.3	0.1	0.2	0.4	1
neos-5261882-treska	0.12	0.1	0.9	0.0	0.0	1
neos-595904	0.02	0.3	0.5	0.1	0.1	1
neos-631517	0.04	0.0	0.1	0.9	0.0	1
neos-691058	0.23	0.0	0.1	0.8	0.1	1
neos-860300	0.46	0.1	0.3	0.1	0.5	1
neos-911970	0.06	0.2	0.6	0.1	0.1	1
net12	0.71	0.0	0.9	0.1	0.0	1
newdano	0.26	0.0	0.4	0.6	0.0	1
nexp-150-20-1-5	0.06	0.0	0.2	0.3	0.5	1
nexp-150-20-8-5	0.25	0.5	0.1	0.4	0.0	1
ns2071214	0.67	0.0	0.0	0.1	0.9	4
p200x1188c	0.02	0.0	0.8	0.1	0.1	1
p500x2988	0.21	0.8	0.2	0.0	0.0	1
pg	0.37	0.0	0.1	0.0	0.9	1
pg5_34	0.18	0.4	0.4	0.2	0.0	1
physiciansched5-3	0.06	0.0	0.1	0.2	0.7	1
pw-myciel4	0.09	0.1	0.8	0.0	0.1	2
r50x360	0.07	0.3	0.3	0.3	0.1	1
ran12x21	0.07	0.8	0.0	0.2	0.0	2
ran13x13	0.03	0.0	0.8	0.2	0.0	1
rococoB10-011000	0.05	0.0	0.8	0.1	0.1	1
rococoC11-010100	1.0	0.0	0.4	0.1	0.5	1
rococoC11-011100	0.96	0.0	0.0	0.4	0.6	1
roll3000	0.06	0.0	0.3	0.4	0.3	1
set3-10	0.09	0.0	0.1	0.5	0.4	1
shipsched	0.22	0.0	0.7	0.3	0.0	1
sorrell7	0.05	0.0	0.2	0.5	0.3	2
sp150x300d	0.18	0.0	0.1	0.1	0.8	2
supportcase20	0.02	0.5	0.2	0.2	0.1	1
supportcase26	0.0	0.4	0.2	0.0	0.4	1
swath	0.0	0.0	0.4	0.0	0.6	1
tanglegram6	0.05	0.0	0.2	0.3	0.5	1
timtab1	0.21	0.6	0.4	0.0	0.0	4
timtab1CUTS	0.02	0.1	0.2	0.1	0.6	1
tr12-30	0.24	0.2	0.2	0.5	0.1	1
uct-subprob	0.04	0.1	0.2	0.2	0.5	3
usAbbrv-8-25_70	0.01	0.1	0.1	0.4	0.4	26

Table 8: Per instance results of Experiment 6.1 (Grid search). GAP refers to the relative gap improvement. $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ are the multipliers for `dcd`, `eff`, `isp`, and `obp`. #BP refers to the number of best parameter combinations. In the case of #BP > 1, a single best choice $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ is provided.

Instance	GAP	λ_1	λ_2	λ_3	λ_4
2club200v15p5scn	0.0	0.33	0.27	0.14	0.26
50v-10	−0.01	0.26	0.31	0.21	0.22
a1c1s1	−0.03	0.26	0.27	0.18	0.3
a2c1s1	−0.02	0.26	0.27	0.18	0.3
app3	−0.02	0.31	0.28	0.14	0.27
b-ball	−0.35	0.27	0.25	0.22	0.26
b1c1s1	0.04	0.26	0.27	0.19	0.29
b2c1s1	0.02	0.26	0.27	0.19	0.29
bab1	0.0	0.29	0.26	0.2	0.25
beasleyC1	0.06	0.26	0.29	0.22	0.23
beasleyC2	0.04	0.26	0.29	0.22	0.23
berlin	0.01	0.26	0.29	0.22	0.23
bg512142	0.0	0.26	0.28	0.2	0.26
bienst1	0.03	0.27	0.25	0.22	0.27
bienst2	−0.11	0.27	0.25	0.22	0.27
binkar10_1	−0.02	0.27	0.25	0.22	0.26
bppc8-02	0.08	0.25	0.26	0.17	0.32
brasil	0.02	0.26	0.29	0.22	0.23
cost266-UUE	0.0	0.26	0.25	0.23	0.26
danoint	0.0	0.26	0.25	0.21	0.28
dfn-bwin-DBE	−0.1	0.26	0.32	0.21	0.21
dg012142	0.0	0.26	0.29	0.19	0.26
drayage-100-12	0.09	0.25	0.3	0.21	0.24
drayage-100-23	0.25	0.25	0.3	0.21	0.24
drayage-25-23	0.15	0.25	0.3	0.21	0.24
drayage-25-27	0.02	0.25	0.3	0.21	0.24
drayage-25-32	−0.01	0.25	0.3	0.21	0.24
eil33-2	0.0	0.32	0.27	0.15	0.26
exp-1-500-5-5	−0.02	0.27	0.27	0.22	0.24
f2gap201600	0.0	0.25	0.3	0.24	0.22
f2gap401600	−0.05	0.25	0.3	0.24	0.22
f2gap801600	0.03	0.24	0.3	0.24	0.22
fhnw-schedule-paira100	0.0	0.26	0.27	0.16	0.31
g200x740	−0.03	0.26	0.3	0.22	0.22
gmu-35-40	0.0	0.37	0.3	0.07	0.26
h50x2450	−0.02	0.26	0.29	0.22	0.23
h80x6320d	0.03	0.26	0.29	0.23	0.23
hypothyroid-k1	0.0	0.31	0.27	0.16	0.25
ic97_tension	−0.89	0.26	0.26	0.13	0.35
istanbul-no-cutoff	0.0	0.25	0.25	0.1	0.39
k16x240b	0.0	0.26	0.3	0.22	0.22
lectsched-5-obj	0.0	0.26	0.26	0.21	0.28
lotsize	−0.02	0.25	0.28	0.19	0.28
map16715-04	0.0	0.27	0.26	0.18	0.3
mik-250-20-75-1	0.06	0.32	0.28	0.15	0.25
mik-250-20-75-2	0.04	0.32	0.28	0.14	0.25
mik-250-20-75-3	−0.04	0.32	0.28	0.14	0.25
mik-250-20-75-4	0.04	0.32	0.28	0.14	0.25
mik-250-20-75-5	−0.04	0.32	0.28	0.15	0.25
milo-v12-6-r1-58-1	0.0	0.25	0.29	0.19	0.26
mkc1	0.28	0.31	0.28	0.13	0.28
mtest4ma	0.15	0.26	0.29	0.22	0.23
n3700	0.0	0.26	0.29	0.22	0.23
n3705	0.0	0.26	0.29	0.22	0.23
n3707	0.0	0.26	0.29	0.22	0.23
n3709	0.0	0.26	0.29	0.22	0.23
n370b	0.0	0.26	0.29	0.22	0.23
n5-3	−0.08	0.26	0.26	0.23	0.25
n6-3	0.06	0.26	0.25	0.23	0.26
n7-3	0.14	0.26	0.25	0.23	0.25

Instance	GAP	λ_1	λ_2	λ_3	λ_4
n9-3	0.04	0.26	0.25	0.22	0.26
neos-1423785	0.02	0.25	0.28	0.17	0.3
neos-1445738	0.0	0.26	0.26	0.19	0.3
neos-1456979	−0.07	0.25	0.32	0.21	0.21
neos-1593097	0.23	0.26	0.27	0.23	0.24
neos-3046601-motu	0.01	0.27	0.25	0.23	0.26
neos-3046615-murg	0.0	0.27	0.25	0.23	0.26
neos-3072252-nete	0.01	0.26	0.25	0.2	0.29
neos-3581454-haast	0.0	0.25	0.31	0.21	0.23
neos-3627168-kasai	0.0	0.26	0.27	0.23	0.24
neos-4333596-skien	0.08	0.29	0.27	0.19	0.26
neos-4343293-stony	0.0	0.26	0.29	0.22	0.24
neos-4650160-yukon	0.0	0.26	0.29	0.18	0.27
neos-4736745-arroux	0.01	0.25	0.28	0.17	0.3
neos-4738912-atrato	−0.05	0.25	0.28	0.18	0.3
neos-4954672-berkel	−0.02	0.26	0.27	0.22	0.24
neos-5076235-embley	0.01	0.25	0.28	0.16	0.31
neos-5079731-flyers	−0.02	0.25	0.28	0.16	0.31
neos-5093327-huahum	−0.01	0.25	0.29	0.18	0.27
neos-5102383-irwell	0.03	0.25	0.28	0.16	0.31
neos-5107597-kakapo	0.0	0.26	0.26	0.21	0.27
neos-5140963-mincio	0.0	0.25	0.26	0.25	0.24
neos-5260764-orauea	0.0	0.25	0.3	0.23	0.23
neos-5261882-treska	−0.06	0.25	0.32	0.2	0.23
neos-595904	−0.1	0.26	0.27	0.23	0.24
neos-631517	0.02	0.25	0.26	0.2	0.29
neos-691058	0.08	0.25	0.25	0.24	0.26
neos-860300	−0.01	0.24	0.29	0.24	0.22
neos-911970	−0.03	0.25	0.26	0.19	0.3
net12	0.26	0.25	0.25	0.23	0.26
newdano	0.05	0.27	0.25	0.22	0.27
nexp-150-20-1-5	−0.12	0.26	0.3	0.22	0.22
nexp-150-20-8-5	0.12	0.25	0.3	0.22	0.23
ns2071214	0.33	0.25	0.26	0.17	0.31
p200x1188c	0.0	0.26	0.3	0.22	0.22
p500x2988	0.01	0.26	0.3	0.22	0.22
pg	0.07	0.32	0.29	0.09	0.3
pg5_34	−0.19	0.3	0.25	0.18	0.26
physiciansched5-3	0.01	0.25	0.27	0.19	0.29
pw-myciel4	0.0	0.27	0.25	0.23	0.26
r50x360	0.03	0.26	0.3	0.23	0.22
ran12x21	0.03	0.26	0.3	0.22	0.22
ran13x13	−0.03	0.26	0.3	0.22	0.22
rococoB10-011000	0.0	0.26	0.25	0.23	0.25
rococoC11-010100	−0.1	0.24	0.28	0.14	0.34
rococoC11-011100	−0.1	0.24	0.28	0.13	0.34
roll3000	−0.37	0.26	0.32	0.2	0.23
set3-10	0.0	0.25	0.27	0.17	0.3
shipsched	0.0	0.25	0.28	0.15	0.32
sorrell7	0.01	0.33	0.27	0.14	0.26
sp150x300d	−0.34	0.26	0.31	0.21	0.22
supportcase20	−0.01	0.25	0.31	0.24	0.21
supportcase26	0.0	0.27	0.25	0.23	0.26
swath	0.0	0.24	0.3	0.24	0.22
tanglegram6	0.0	0.24	0.3	0.25	0.21
timtab1	0.04	0.26	0.29	0.22	0.23
timtab1CUTS	0.0	0.25	0.28	0.22	0.25
tr12-30	0.16	0.26	0.29	0.22	0.24
uct-subprob	0.0	0.25	0.29	0.23	0.23
usAbbrv-8-25_70	0.0	0.26	0.26	0.17	0.3

Table 9: Per instance results of Experiment 6.2 (Random seed). GAP refers to the relative gap improvement. $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ are the multipliers for `dcd`, `eff`, `isp`, and `obp`.

Instance	GAP	λ_1	λ_2	λ_3	λ_4	Instance	GAP	λ_1	λ_2	λ_3	λ_4
2club200v15p5scn	0.0	0.48	0.06	0.19	0.27	n9-3	0.04	0.27	0.53	0.2	0.0
50v-10	0.05	0.26	0.27	0.47	0.0	neos-1423785	0.13	0.25	0.27	0.27	0.22
a1c1s1	0.03	0.19	0.36	0.14	0.32	neos-1445738	0.0	0.25	0.23	0.0	0.52
a2c1s1	0.01	0.23	0.09	0.25	0.43	neos-1456979	−0.01	0.28	0.42	0.2	0.11
app3	0.02	0.26	0.22	0.0	0.53	neos-1593097	0.3	0.29	0.41	0.09	0.21
b-ball	0.44	0.25	0.26	0.28	0.22	neos-3046601-motu	0.01	0.31	0.27	0.23	0.19
b1c1s1	0.1	0.26	0.4	0.24	0.11	neos-3046615-murg	0.01	0.29	0.65	0.0	0.05
b2c1s1	0.04	0.22	0.2	0.39	0.19	neos-3072252-nete	0.01	0.25	0.41	0.21	0.14
bab1	0.01	0.29	0.3	0.19	0.22	neos-3581454-haast	0.01	0.21	0.17	0.27	0.35
beasleyC1	0.03	0.21	0.27	0.17	0.34	neos-3627168-kasai	0.0	0.19	0.31	0.26	0.24
beasleyC2	0.01	0.27	0.3	0.24	0.19	neos-4333596-skien	0.06	0.27	0.34	0.2	0.19
berlin	0.0	0.31	0.42	0.1	0.17	neos-4343293-stony	0.04	0.27	0.26	0.14	0.33
bg512142	0.0	0.28	0.44	0.0	0.28	neos-4650160-yukon	0.03	0.21	0.31	0.06	0.42
bienst1	0.02	0.18	0.32	0.31	0.2	neos-4736745-arroux	0.01	0.23	0.37	0.01	0.39
bienst2	−0.04	0.46	0.33	0.0	0.21	neos-4738912-atrato	0.19	0.23	0.32	0.02	0.43
binkar10	0.1	0.17	0.36	0.0	0.46	neos-4954672-berkel	0.04	0.2	0.52	0.03	0.25
bppc8-02	0.58	0.16	0.38	0.0	0.46	neos-5076235-embley	0.02	0.18	0.27	0.37	0.18
brasil	0.01	0.25	0.48	0.27	0.0	neos-5079731-flyers	−0.01	0.21	0.34	0.11	0.34
cost266-UUE	0.01	0.22	0.21	0.37	0.21	neos-5093327-huahum	0.02	0.24	0.29	0.11	0.36
danoint	0.0	0.22	0.44	0.25	0.1	neos-5102383-irwell	0.04	0.26	0.31	0.0	0.43
dfn-bwin-DBE	0.08	0.23	0.49	0.05	0.23	neos-5107597-kakapo	0.0	0.25	0.28	0.2	0.28
dg012142	0.0	0.22	0.29	0.19	0.3	neos-5140963-mincio	0.0	0.14	0.18	0.39	0.28
drayage-100-12	0.12	0.24	0.27	0.41	0.08	neos-5260764-orauea	0.01	0.38	0.09	0.27	0.27
drayage-100-23	0.35	0.23	0.41	0.19	0.17	neos-5261882-treska	0.06	0.45	0.28	0.08	0.19
drayage-25-23	0.06	0.22	0.25	0.33	0.2	neos-595904	−0.06	0.23	0.15	0.42	0.2
drayage-25-27	0.0	0.25	0.17	0.26	0.32	neos-631517	0.02	0.29	0.21	0.33	0.17
drayage-25-32	−0.06	0.21	0.27	0.28	0.24	neos-691058	0.11	0.14	0.16	0.43	0.27
eil33-2	0.0	0.47	0.18	0.01	0.34	neos-860300	0.23	0.24	0.39	0.25	0.12
exp-1-500-5-5	0.0	0.21	0.19	0.16	0.44	neos-911970	−0.04	0.23	0.2	0.25	0.31
f2gap201600	0.03	0.23	0.77	0.0	0.0	net12	0.27	0.26	0.32	0.27	0.15
f2gap401600	0.12	0.21	0.78	0.0	0.01	newdano	0.09	0.18	0.32	0.39	0.1
f2gap801600	0.18	0.21	0.73	0.0	0.06	nexp-150-20-1-5	0.0	0.22	0.14	0.26	0.39
fhnw-schedule-paira100	0.0	0.38	0.34	0.0	0.28	nexp-150-20-8-5	0.24	0.27	0.15	0.24	0.33
g200x740	0.44	0.24	0.13	0.21	0.42	ns2071214	0.33	0.23	0.04	0.39	0.34
gmu-35-40	0.01	0.13	0.12	0.31	0.44	p200x1188c	0.0	0.24	0.55	0.0	0.21
h50x2450	0.04	0.36	0.39	0.04	0.2	p500x2988	−0.01	0.31	0.36	0.0	0.33
h80x6320d	0.02	0.37	0.26	0.26	0.11	pg	0.27	0.33	0.13	0.0	0.54
hypothyroid-k1	0.0	0.32	0.48	0.01	0.18	pg5	0.12	0.43	0.38	0.19	0.0
ic97	0.04	0.12	0.52	0.02	0.34	physiciansched5-3	0.03	0.19	0.4	0.1	0.31
istanbul-no-cutoff	0.04	0.3	0.07	0.33	0.3	pw-myciel4	0.0	0.4	0.31	0.17	0.12
k16x240b	0.02	0.24	0.22	0.12	0.41	r50x360	0.06	0.19	0.35	0.39	0.07
lectsched-5-obj	0.08	0.09	0.14	0.02	0.75	ran12x21	0.07	0.25	0.19	0.15	0.42
lotsize	−0.01	0.24	0.3	0.07	0.39	ran13x13	−0.01	0.33	0.49	0.0	0.18
map16715-04	0.0	0.3	0.07	0.28	0.36	rococoB10-011000	0.01	0.28	0.46	0.04	0.22
mik-250-20-75-1	0.13	0.28	0.13	0.35	0.24	rococoC11-010100	0.86	0.0	0.46	0.1	0.44
mik-250-20-75-2	0.06	0.42	0.2	0.28	0.1	rococoC11-011100	0.32	0.21	0.0	0.44	0.35
mik-250-20-75-3	0.01	0.34	0.4	0.04	0.22	roll3000	0.01	0.24	0.18	0.23	0.35
mik-250-20-75-4	0.07	0.28	0.48	0.22	0.02	set3-10	0.05	0.23	0.42	0.17	0.18
mik-250-20-75-5	0.07	0.33	0.44	0.12	0.11	shipsched	−0.02	0.0	0.47	0.0	0.53
milo-v12-6-r1-58-1	0.0	0.36	0.39	0.0	0.26	sorrell7	0.03	0.29	0.07	0.0	0.64
mkc1	0.41	0.39	0.13	0.27	0.21	sp150x300d	−0.09	0.28	0.45	0.03	0.24
mtest4ma	0.23	0.25	0.35	0.27	0.13	supportcase20	0.01	0.26	0.29	0.39	0.06
n3700	0.0	0.24	0.15	0.25	0.37	supportcase26	0.0	0.27	0.4	0.25	0.08
n3705	0.0	0.18	0.28	0.32	0.22	swath	0.0	0.25	0.21	0.26	0.28
n3707	0.01	0.14	0.23	0.35	0.27	tanglegram6	0.02	0.22	0.3	0.18	0.29
n3709	0.0	0.2	0.27	0.21	0.32	timtab1	0.16	0.12	0.12	0.39	0.37
n370b	0.01	0.23	0.45	0.25	0.07	timtab1CUTS	0.01	0.3	0.38	0.2	0.13
n5-3	0.15	0.27	0.41	0.32	0.0	tr12-30	0.13	0.21	0.28	0.34	0.17
n6-3	0.08	0.21	0.26	0.37	0.17	uct-subprob	0.01	0.28	0.6	0.1	0.01
n7-3	0.18	0.23	0.14	0.32	0.3	usAbbrv-8-25	0.01	0.16	0.1	0.58	0.16

Table 10: Per instance results of Experiment 6.3 (Learning capabilities). GAP refers to the relative gap improvement. $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ are the multipliers for `dcd`, `eff`, `isp`, and `obp`.

Instance	GAP	λ_1	λ_2	λ_3	λ_4	Instance	GAP	λ_1	λ_2	λ_3	λ_4
2club200v15p5scn	0.0	0.24	0.28	0.3	0.18	n9-3	0.03	0.21	0.32	0.33	0.14
50v-10	0.0	0.26	0.49	0.1	0.14	neos-1423785	0.06	0.09	0.47	0.08	0.36
a1c1s1	0.01	0.1	0.45	0.12	0.33	neos-1445738	0.0	0.15	0.4	0.19	0.25
a2c1s1	−0.03	0.1	0.45	0.12	0.33	neos-1456979	−0.12	0.24	0.46	0.11	0.18
app3	0.0	0.11	0.4	0.1	0.38	neos-1593097	0.33	0.23	0.33	0.3	0.13
b-ball	0.07	0.26	0.29	0.36	0.09	neos-3046601-motu	0.0	0.24	0.23	0.4	0.13
b1c1s1	0.03	0.12	0.42	0.15	0.3	neos-3046615-murg	0.0	0.24	0.23	0.4	0.13
b2c1s1	0.0	0.12	0.43	0.15	0.3	neos-3072252-nete	0.01	0.14	0.39	0.25	0.22
bab1	0.01	0.23	0.26	0.37	0.13	neos-3581454-haast	−0.02	0.22	0.46	0.1	0.22
beasleyC1	0.03	0.25	0.43	0.22	0.1	neos-3627168-kasai	0.0	0.25	0.33	0.29	0.12
beasleyC2	−0.02	0.26	0.43	0.21	0.1	neos-4333596-skien	0.07	0.21	0.35	0.23	0.21
berlin	0.0	0.25	0.42	0.21	0.11	neos-4343293-stony	0.0	0.22	0.37	0.24	0.18
bg512142	0.0	0.18	0.42	0.16	0.24	neos-4650160-yukon	0.0	0.15	0.46	0.11	0.28
bienst1	0.0	0.21	0.3	0.34	0.14	neos-4736745-arroux	0.0	0.12	0.46	0.07	0.36
bienst2	−0.15	0.21	0.3	0.34	0.14	neos-4738912-atrato	−0.04	0.13	0.45	0.08	0.35
binkar10_1	−0.01	0.21	0.32	0.32	0.15	neos-4954672-berkel	−0.04	0.24	0.31	0.32	0.13
bppc8-02	0.58	0.14	0.38	0.1	0.38	neos-5076235-embley	0.0	0.05	0.53	0.05	0.38
brasil	0.02	0.24	0.41	0.22	0.12	neos-5079731-flyers	−0.05	0.05	0.53	0.05	0.38
cost266-UUE	0.0	0.23	0.29	0.37	0.12	neos-5093327-huahum	0.0	0.15	0.5	0.08	0.27
danoint	0.0	0.17	0.31	0.31	0.21	neos-5102383-irwell	0.04	0.05	0.52	0.06	0.37
dfn-bwin-DBE	−0.1	0.26	0.51	0.09	0.14	neos-5107597-kakapo	0.0	0.18	0.27	0.28	0.27
dg012142	0.0	0.18	0.44	0.14	0.24	neos-5140963-mincio	0.0	0.23	0.29	0.38	0.1
drayage-100-12	0.12	0.19	0.39	0.15	0.27	neos-5260764-orauea	0.0	0.22	0.41	0.24	0.13
drayage-100-23	0.3	0.19	0.39	0.14	0.27	neos-5261882-treska	−0.05	0.23	0.5	0.08	0.19
drayage-25-23	0.13	0.19	0.39	0.15	0.27	neos-595904	−0.02	0.24	0.32	0.31	0.12
drayage-25-27	0.08	0.19	0.39	0.15	0.27	neos-631517	0.03	0.11	0.36	0.2	0.32
drayage-25-32	−0.03	0.19	0.4	0.14	0.27	neos-691058	0.16	0.21	0.23	0.4	0.17
eil33-2	0.0	0.34	0.43	0.22	0.0	neos-860300	0.21	0.24	0.46	0.26	0.05
exp-1-500-5-5	−0.04	0.25	0.31	0.31	0.13	neos-911970	−0.19	0.11	0.39	0.15	0.35
f2gap201600	0.01	0.25	0.49	0.21	0.06	net12	0.16	0.21	0.26	0.36	0.17
f2gap401600	0.03	0.25	0.49	0.21	0.06	newdano	0.09	0.21	0.3	0.34	0.14
f2gap801600	0.05	0.24	0.5	0.2	0.06	nexp-150-20-1-5	−0.06	0.27	0.45	0.19	0.09
fhnw-schedule-paira100	0.0	0.06	0.44	0.08	0.42	nexp-150-20-8-5	−0.02	0.24	0.41	0.16	0.18
g200x740	−0.02	0.26	0.43	0.21	0.1	ns2071214	0.0	0.07	0.44	0.1	0.39
gmu-35-40	0.0	0.06	0.4	0.0	0.54	p200x1188c	0.01	0.26	0.44	0.2	0.1
h50x2450	−0.03	0.25	0.38	0.23	0.14	p500x2988	0.02	0.25	0.44	0.2	0.11
h80x6320d	0.01	0.24	0.37	0.25	0.14	pg	0.27	0.01	0.15	0.01	0.82
hypothyroid-k1	0.0	0.29	0.34	0.3	0.07	pg5_34	0.01	0.3	0.32	0.33	0.06
ic97_tension	−0.19	0.04	0.45	0.07	0.44	physiciansched5-3	0.02	0.11	0.39	0.15	0.34
istanbul-no-cutoff	0.02	0.02	0.44	0.03	0.5	pw-myciel4	0.0	0.24	0.23	0.4	0.13
k16x240b	0.0	0.27	0.41	0.2	0.11	r50x360	−0.04	0.27	0.43	0.21	0.1
lectsched-5-obj	0.0	0.16	0.33	0.26	0.25	ran12x21	0.02	0.27	0.41	0.21	0.11
lotsize	−0.01	0.14	0.41	0.15	0.31	ran13x13	−0.02	0.27	0.41	0.21	0.11
map16715-04	0.0	0.15	0.42	0.16	0.26	rococoB10-011000	−0.01	0.24	0.32	0.35	0.09
mik-250-20-75-1	0.04	0.28	0.51	0.1	0.1	rococoC11-010100	−0.11	0.0	0.54	0.0	0.46
mik-250-20-75-2	0.04	0.28	0.52	0.1	0.1	rococoC11-011100	−0.05	0.0	0.53	0.01	0.45
mik-250-20-75-3	0.0	0.28	0.52	0.1	0.1	roll3000	−0.45	0.23	0.44	0.09	0.24
mik-250-20-75-4	0.09	0.28	0.52	0.1	0.09	set3-10	0.02	0.08	0.46	0.12	0.34
mik-250-20-75-5	0.06	0.29	0.52	0.1	0.09	shipsched	0.0	0.05	0.49	0.03	0.43
milo-v12-6-r1-58-1	0.0	0.16	0.43	0.13	0.28	sorrell7	0.01	0.23	0.27	0.31	0.19
mkc1	0.2	0.15	0.49	0.04	0.32	sp150x300d	−0.29	0.27	0.44	0.18	0.11
mtest4ma	0.08	0.25	0.37	0.24	0.14	supportcase20	−0.01	0.26	0.44	0.21	0.09
n3700	0.0	0.24	0.42	0.18	0.16	supportcase26	0.0	0.24	0.23	0.4	0.13
n3705	0.0	0.24	0.42	0.18	0.16	swath	0.0	0.23	0.45	0.23	0.09
n3707	0.0	0.24	0.42	0.18	0.15	tanglegram6	0.0	0.24	0.47	0.23	0.06
n3709	0.0	0.24	0.42	0.19	0.15	timtab1	0.04	0.23	0.32	0.28	0.16
n370b	0.01	0.24	0.42	0.18	0.16	timtab1CUTS	0.0	0.21	0.36	0.25	0.18
n5-3	0.1	0.23	0.32	0.36	0.09	tr12-30	0.2	0.23	0.35	0.24	0.19
n6-3	0.01	0.22	0.32	0.35	0.12	uct-subprob	0.0	0.24	0.43	0.25	0.08
n7-3	0.18	0.23	0.31	0.36	0.1	usAbbrv-8-25_70	0.0	0.09	0.4	0.12	0.39

Table 11: Per instance results of Experiment 6.4 (Generalisation capabilities). GAP refers to the relative gap improvement. $\{\lambda_1, \lambda_2, \lambda_3, \lambda_4\}$ are the multipliers for `dcd`, `eff`, `isp`, and `obp`.