Approximating the Gomory Mixed-Integer Cut Closure Using Historical Data

Berkay Becu[1], Santanu S. Dey[1], Feng Qiu[2], and Álinson S. Xavier[2]

[1] Georgia Institute of Technology, Atlanta, GA, USA
bbecu3@gatech.edu,santanu.dey@isye.gatech.edu
[2] Argonne National Laboratory, Lemont, IL, USA {fqiu,axavier}@anl.gov

Abstract. Many operations related optimization problems involve repeatedly solving similar mixed integer linear programming (MILP) instances with the same constraint matrix but differing objective coefficients and right-hand-side values. The goal of this paper is to generate good cutting-planes for such instances using historical data. Gomory mixed integer cuts (GMIC) for a general MILP can be parameterized by a vector of weights to aggregate the constraints into a single equality constraint, where each such equality constraint in turn yields a unique GMIC. In this paper, we prove that for a family of MILP instances, where the right-hand-side of the instances belongs to a lattice, the GMIC closure for every instance in this infinite family can be obtained using the same finite list of aggregation weights. This result motivates us to build a simple heuristic to efficiently select aggregations for generating GMICs from historical data of similar instances with varying right-hand-sides and objective function coefficients. For testing our method, we generated families of instances by perturbing the right-hand-side and objective functions of MIPLIB 2017 instances. The proposed heuristic can significantly accelerate the performance of Gurobi for many benchmark instances, even when taking into account the time required to predict aggregation multipliers and compute the cut coefficients. To the best of our knowledge, this is the first work in the literature of data-driven cutting plane generation that is able to significantly accelerate the performance of a commercial state-of-the-art MILP solver, using default solver settings, on large-scale benchmark instances.

Keywords: Mixed-Integer Linear Programming · Gomory Mixed-Integer Cuts · Machine Learning

1 Introduction

Mixed-Integer Linear Programming (MILP) models are essential for solving optimization problems across various industrial sectors, including logistics, manufacturing and power systems. In many practical settings, decision-makers must solve challenging instances under strict time limits, as the data needed to formulate the problem often becomes available only shortly before a solution is required.

For example, the unit commitment problem, which Independent System Operators solve daily to clear day-ahead electricity markets, requires solutions within minutes, since operators only have a few hours to clear the markets after receiving bids from participants [52]. Further applications are discussed in [37,53,38].

Operational MILPs often exhibit two key characteristics that suggest a solution approach leveraging historical data. First, these problems are usually solved on a recurring basis — often hourly or daily — creating an opportunity for data collection, analysis and offline processing, which traditional MILP research, focused on one-shot solution methods, has not taken advantage of. Second, instances of these problems tend to be structurally similar: decisions are made over the same set of variables, and the constraint matrix typically remains the same, as it encodes decision logic or invariant physical resources. What usually varies between instances are constraint right-hand-sides and objective function coefficients, reflecting fluctuations for example in costs, revenues, demands and availability of different resources. Thus, while each instance may be challenging to solve independently, from scratch, if instances are viewed as random samples drawn from a particular probability distribution, it may be feasible to leverage offline processing and historical information to accelerate the solution of new instances drawn from this distribution.

Many papers have proposed data-driven methods to construct high-quality feasible solutions [52,11,37] and improve the performance of branching rules in branch-and-bound methods [2,41,35,53], as well as conduct sensitivity analysis with respect to changing right-hand-sides [18]. Recent work has also explored data-driven cutting plane generation, which we consider in this manuscript. In the following, we provide only a brief review of this literature, and we refer to [27] for a more complete survey. One of the pioneering works in this area is by Tang et al. [47], which models the cut selection process as a Markov decision process and proposes the use of reinforcement learning. Paulus et al. [43] apply imitation learning instead, where a greedy algorithm acts as the expert to select an individual cut that maximizes gap closure. Several studies have trained neural networks to select multiple cuts simultaneously [36,50], instead of adding one cut at a time. Other research [48,10] focus on learning cut-related solver parameters tailored to specific instances. On the theoretical side, Balcan et al. [9] derive upper bounds on the sample size required to estimate the expected branch-and-bound tree size when one *Gomory mixed integer cut* (GMIC) is added, Cheng et al. [16] further study sample complexity for learning multiple cutting planes using neural networks, and Cheng et al. [15] provide such bounds for learning general group cuts. Several methods consider generating cutting-planes directly using machine learning. Chetelat et al. [17] show how the separation problem of GMI cuts can be reformulated as a training problem using ReLU activation functions. Dragotto et al. [28] introduce a "cutting-plane layer", a differentiable generator that maps problem data and previous iterates to cutting planes. Finally, and most recently, Guage et al. [34] present a hybrid framework where an ML model simplifies the cut-generating optimization problem.

In this paper, our main goal is to show how one may exploit historical data to generate effective cutting planes for families of similar MILP instances. More specifically, we aim to accelerate MILP solver performance by learning to generate cuts that approximate the Gomory Mixed-Integer Cut (GMIC) closure for families of instances with same constraint matrix, but varying right-hand-sides and objective functions. As we further discuss in Section 2, GMIC cuts were among the first cutting planes developed for solving MILPs and their closure has many desirable theoretical and computational properties. Concretely, the GMIC closure is known to be polyhedral [20], rank-1 GMI cuts are substantially more numerically stable than higher-rank cuts, and computational studies on MIPLIB instances have shown that they effectively close a significant portion of the integrality gap [8,23]. However, exact separation over the GMIC closure is NP-hard [14,39], and heuristics such as those in [23,30] remain too computationally expensive to be integrated into state-of-the-art commercial MILP solvers.

We have two main contributions in this manuscript. First, in Section 3, we show a new finiteness result regarding GMI closures for infinite families of MILPs: given an infinite family of MILPs where each instance of the family has the same constraint matrix and the right-hand-sides of these instances belong to a lattice, we show that there exists a finite list of constraint aggregation multipliers that produces the GMIC closure for each instance in this family. Second, inspired by this theoretical result, in Section 4 we propose a simple new heuristic for learning to generate constraint aggregation multipliers in order to approximate the GMIC closure. In Section 5, we conduct extensive computational experiments on randomly perturbed MIPLIB 2017 [31] instances and show that the proposed heuristic can significantly accelerate the performance of Gurobi for many benchmark instances, even when taking into account the time required to predict aggregation multipliers and compute the cut coefficients. To the best of our knowledge, this is the first work in the literature of data-driven cutting plane generation that is able to significantly accelerate the performance of a commercial state-of-the-art MILP solver, using default solver settings, on large-scale benchmark instances. The proposed methods have been made publicly available as part of the open-source MIPLearn software package [45].

2 The Gomory Mixed-Integer Cut Closure

Cutting-planes are an integral tool for improving the dual bounds in modern state-of-the-art MILP solvers [12,26]. One of the first classes of cutting-planes invented for MILPs was that of the Gomory Mixed-Integer Cuts (GMIC) [32,7,44,21]. Given a single constraint,

$$S := \left\{ x \in \mathbb{R}^n_+ \;\middle|\; \sum_{i\in[n]} \alpha_j x_j = \beta,\ x_j \in \mathbb{Z} \text{ for } j \in J \right\},$$

where $J \subseteq [n]$, the GMIC is a valid inequality for S, which we denote as GMIC(S), and is obtained as:

$$\left.\begin{array}{l}\sum_{j\in J, f(\alpha_j)\le f(\beta)} \frac{f(\alpha_j)}{f(\beta)} x_j + \sum_{i\in J, f(\alpha_j)>f(\beta)} \frac{1-f(\alpha_j)}{1-f(\beta)} x_j \\ + \sum_{j\in[n]\setminus J, \alpha_j\ge 0} \frac{\alpha_j}{f(\beta)} x_j + \sum_{j\in[n]\setminus J, \alpha_j<0} \frac{-\alpha_j}{1-f(\beta)} x_j \ge 1,\end{array}\right\} \text{GMIC}(S) \tag{1}$$

where $f(u) = u - \lfloor u \rfloor$. Henceforth, for simplicity, we will also refer to the half-space described by the inequality (1) as GMIC(S). Moreover, if $\beta \in \mathbb{Z}$, then the inequality (1) is not well-defined, in which case GMIC(S) will simply refer to $\mathbb{R}^n_+$. When S is a row of the simplex tableau corresponding to basic integral variable, and the right-hand-side β is fractional, then it is easy to see that the GMIC separates the current basis feasible solution.

A *cutting-plane closure* is informally the set obtained by simultaneously adding all cuts that can be derived using a given type of cutting-plane procedure. The practical importance of closures is closely related to the history of GMIC cuts. In his seminal finite cutting-plane algorithm [33], Gomory prescribes iteratively adding a *single* cut that separates the current fractional optimal solution to the linear programming (LP) relaxation, and then re-resolving the updated linear program. By the 1990s, several experimental studies using this *single-cut* approach with GMICs, lead researchers, including Gomory himself, to mistakenly believe that GMIC were “useless in practice" [21]. One of the most exciting advances in the field of MILPs is a result by Balas et al. [7], where they debunked this belief by showing that GMICs perform very well in practice when inequalities are added *simultaneously* from all fractional rows corresponding to integral basic variables of the optimal tableaux. Although cut closures were already theoretically well-studied by the time of [7], this work solidified their practical importance.

Given a general MILP in the following form:

$$\text{IP} := \{x \in \mathbb{R}^n_+ \mid Ax = b,\ x_j \in \mathbb{Z} \text{ for } j \in J\}, \tag{2}$$

where $A \in \mathbb{Q}^{m\times n}$ and $b \in \mathbb{Q}^m$, the GMIC closure is obtained by applying the GMIC procedure as described in (1) to all single-constraint relaxations of IP. That is, the GMIC closure, denoted by $\mathcal{G}(\text{IP})$, is given by:

$$\mathcal{G}(\text{IP}) := \bigcap_{\lambda\in\mathbb{R}^m} \text{GMIC}\left(\text{IP}_\lambda\right), \tag{3}$$

where

$$\text{IP}_\lambda := \left\{x \in \mathbb{R}^n_+ \mid \lambda^\top Ax = \lambda^\top b,\ x_j \in \mathbb{Z} \text{ for } j \in J\right\}. \tag{4}$$

We will refer to $\lambda \in \mathbb{R}^m$ in (4) as an aggregation multiplier [13]. All the cuts above are considered *rank-1* as they can be directly generated from the original problem constraints.

The GMIC closure has various desirable theoretical and computational properties. For example, the GMIC closure has interestingly been shown to be equivalent to various other cutting plane closures, including the mixed integer rounding inequality closure [42,40,24] and the split disjunctive cut closure [6,20], which highlights its theoretical importance. Moreover, these closures have been shown to be polyhedral, with a finite number of the cuts dominating all the other cuts [20,3,49,24,5]. Computationally, the strength of GMIC closures was empirically demonstrated through multiple studies on standard benchmark libraries [8,24]. In particular, on MIPLIB 2003 instances [1], Balas and Saxena [8] showed that these cuts close more than 70% of the integrality gap on average. Another advantage of using rank-1 GMI cuts, in contrast to higher-rank cuts, is that they do not suffer significantly from the numerical challenges [22].

While the above theoretical and computational results make a compelling case for using the GMIC closure, in general this closure is very hard to build. Indeed, separating an arbitrary point in the linear programming (LP) relaxation using a split cut (equivalent to GMI cut) is NP-hard [14,39]. This challenge led to a number of research projects on approximating these closures through heuristic algorithms [23,30]. However, to the best of our knowledge, these heuristics remain too computationally expensive for most problems, and have not been integrated into commercial state-of-the-art MILP solvers.

3 The GMIC Closure for an Infinite Family of MILPs

In this paper, we choose to learn aggregation multipliers $\lambda \in \mathbb{R}^m$ from historical data, which are then used to produce one-constraint relaxations (4) of the current instance under consideration. We then apply (1) to these one-constraint relaxations to obtain GMICs for the current instance.

A basic question is the following: What "information" from historical data, if any, could be exploited to learn good aggregation multipliers? For example, consider two integer programs:

$$\max\{c^\top x \mid Ax = b, x \in \mathbb{R}^n_+, x_j \in \mathbb{Z}, j \in J\},$$

and its perturbation:

$$\max\{(c+\delta)^\top x \mid Ax = b + \varepsilon, x \in \mathbb{R}^n_+, x_j \in \mathbb{Z}, j \in J\},$$

where $\delta \in \mathbb{R}^n$ and $\varepsilon \in \mathbb{R}^m$ are small perturbations. If the LP relaxation of the first MILP is integral, then there are no interesting GMICs to be generated, that is there are no aggregation multipliers that lead to useful GMICs. On the other hand, for arbitrarily small values of ϵ, the LP relaxation of the second MILP may not be integral, and thus there may be several GMICs that are useful. Therefore, simple perturbation can change the GMIC closure dramatically.

Somewhat counter-intuitively, the next theorem establishes that if we examine an *infinite family of MILPs* with same constraint matrix and with right-hand-sides belonging to a lattice (so not arbitrarily close-by right-hand-sides) generated by rational vectors, then there is a *finite list* of multipliers λ that generate the GMIC closure of every instance.

Consider the lattice $\Gamma \subseteq \mathbb{R}^m$ generated by rational vectors $b^1, \ldots, b^k$, that is

$$\Gamma = \left\{ \sum_{i=1}^{k} z_i b^i \mid z_i \in \mathbb{Z} \ \forall i \in [k] \right\}.$$

For the rest of this section, we assume without loss of generality that all the data in (2) is integral, that is $A \in \mathbb{Z}^{m\times n}$ and $b \in \mathbb{Z}^m$. We study the following parametric family of MILPs with varying right-hand-sides:

$$\mathrm{IP}(\gamma) := \left\{ x \in \mathbb{R}^n_+ \mid Ax = b + \gamma,\ x_j \in \mathbb{Z} \text{ for } j \in J \right\}\ \forall \gamma \in \Gamma. \tag{5}$$

With this notation, IP denoted in (2) is equal to IP(0).

Theorem 1. *Let Γ be the lattice generated by rational vectors $b^1, \ldots, b^k \in \mathbb{Q}^m$. Consider the infinite family of instances corresponding to Γ as described in (5). Then there exists a finite set $\Lambda \subseteq \mathbb{R}^m$, such that the GMIC closure of every instance $\mathrm{IP}(\gamma)$ can be obtained using aggregation multipliers in Λ, that is:*

$$\mathcal{G}(\mathrm{IP}(\gamma)) = \bigcap_{\lambda \in \Lambda} \mathrm{GMIC}\left(\mathrm{IP}(\gamma)_\lambda\right) \quad \forall \gamma \in \Gamma.$$

Remark 1. Dash et al. [25] show that the split closure for a union of finite number of polyhedron can be described finitely. These results are not directly applicable in our case due to incompatible definitions. Another way to prove a result similar to Theorem 1 is to consider a family of instances where we apply unimodular transformations to the LP relaxation [25]. However, in this case we would be restricted to the right-hand-side being a lattice generated by the columns of the matrix A and not an arbitrary lattice as presented in Theorem 1.

In the case of pure integer programming, we can obtain a stronger result showing that there exist a finite set of multipliers (4) which gives the GMIC closure *for all right-hand-sides.*

Corollary 1. *Consider $\mathrm{IP}(\gamma)$ as defined in (5), where we allow $\gamma \in \mathbb{R}^m$. Let $J = [n]$, that is all variables are integral. Then there exists a finite set of multipliers Γ such that*

$$\mathcal{G}(\mathrm{IP}(\gamma)) = \bigcap_{\lambda \in \Lambda} \mathrm{GMIC}\left(\mathrm{IP}(\gamma)_\lambda\right) \quad \forall \gamma \in \mathbb{R}^m.$$

Remark 2. Corollary 1 is similar in flavor to a result obtained by Wolsey [51] where it is shown that there is a finite list of subadditive functions that give the convex hull of pure integer programs for all right-hand-sides.

4 Learning Heuristic

Theorem 1 suggests a simple learning heuristic. When solving a distribution of MILP instances that have a fixed constraint matrix, one can sample multiple *training instances* from this distribution, compute the aggregation multipliers that yield the GMIC closure for each training instance, then reuse these multipliers when solving new *test instances* drawn from the distribution.

Cut Collection. The first consideration in using this approach is determining how to generate aggregation multipliers for a given training sample — we call this our *cut collection algorithm.* In this manuscript, we adopt a simplified version of the procedure by Fischetti and Salvagnin [30], which generates GMIC cuts from multiple tableau bases. Below, we provide a high-level summary of this method; for a more formal description and additional remarks, see Appendix B. The procedure begins by solving the LP relaxation of the problem and generating GMIC cuts for the optimal basis. These cuts are then added as constraints into the LP relaxation, which is resolved, and dual values for the newly added constraints are computed. Next, using these dual values as penalty, the cuts are transferred to the objective function of the LP relaxation, which is then resolved. Finally, GMIC cuts are generated for this updated basis, and the process continues iteratively until a specified iteration limit is reached or a previously visited basis recurs. We also have steps to eliminate cuts that became dominated as the algorithm progresses.

Cut Selection. A challenge in the design of cutting plane methods is deciding how many cuts to add. It is well known that adding too many cuts can lead to poor algorithm performance [4,26], not only due to the computational cost of generating cuts but also due to their impact on the size and density of the LP relaxation. To achieve a better balance between gap closure and computational efficiency, we propose using multipliers collected only from a subset of training instances. However, two questions remain: how many training instances should be used, and how should they be selected? To answer these questions, we conduct computational experiments with varying numbers of training instances, and we explore different selection strategies based on similarity: most-similar, most-different, and random instance selection.

5 Computational Experiments

To evaluate the performance of the learning heuristic described in Section 4, we implemented it in Julia/JuMP within the MIPLearn framework and conducted comprehensive experiments on randomly perturbed MIPLIB 2017 instances.

5.1 Software and Hardware

We utilized a single workstation computer (Ryzen 9 7950x, 16 cores, 32 threads, 128 GB DDR5) to generate all necessary training data and conduct all benchmarks. During the training phase, we used Gurobi Optimizer 11.0.2 as our LP solver, whereas MIPLearn was used to convert problems into standard form, compute the tableau and generate GMI cuts. In the testing phase, Gurobi acted as the MILP solver, receiving cuts computed by MIPLearn through a cut callback. For all runs, Gurobi was configured to use a single thread, and 16 problems were solved in parallel at a time. All other solver settings, including cut generation and pre-solve, were left at their default configurations. To minimize solver performance variability, we used three random seeds and measured *work units*, in addition to running time. Additionally, since this manuscript focuses on the dual side of the solution process, Gurobi was provided with the optimal solution to all problems during testing.

5.2 Benchmark Set Generation

To create realistic homogeneous families of MILP problems, we generated 55 variations (50 for training, 5 for testing) of each MIPLIB 2017 benchmark instance [31] by perturbing objective coefficients and right-hand-side values. We outline the high-level approach below. See Appendix C for details.

Right-Hand-Side (RHS) Changes. If a constraint has at least one continuous variable or is an inequality, then we perturb the right-hand-side value b_i to $b_i \cdot r$ where r is uniformly sampled from $[0.9, 1.1]$. If the constraint is an equality constraint with integer variables only, then we apply additive perturbation of $\{-1, 0, 1\}$. Note that these perturbations may fail due to two reasons: (i) none of the constraints satisfy the requirements; (ii) the perturbed problem becomes infeasible. If this happens (on any of the 55 generated variations), we disable RHS perturbation for this particular instance.

Objective Changes. We perturb each objective coefficient c_j to $c_j \cdot r$ where r is uniformly randomly selected from $[0.75, 1.25]$. Note that, in some MIPLIB instances, the objective function contains a single variable, in which case the above perturbation does not produce any change.

Number of Perturbed Instances. Out of the 240 benchmark instances in MIPLIB 2017, we discarded 20 instances due to infeasibility, being very large, or being very difficult (requires more than 4 hours to achieve an integer feasible solution). Out of the remaining 220 instances, we discovered that on 34 instances the perturbation rules failed to create different variations, leaving us with 186 instances where we were either successful in changing the RHS values (8 instances) or objective coefficients (105 instances) or both (73 instances).

Perturbation Quality. We solved the five test variations of the 186 instances, then observed optimal objective function values and solution diversity, focusing on the integer part of the optimal solution. Out of 186 instances, 178 showed at least two distinct optimal solutions (integer part). Notably, 157 instances had five distinct solutions, indicating that the rules were overall successful at generating instances with distinct optimal solutions and values. For only 8 instances, the rules produced variations with identical optimal solutions (integer part).

Changes in Problem Difficulty. A potential concern with the proposed perturbation rules was that the resulting instance variations might become easier or trivial to solve. On the contrary we discovered that the variations, on average, became more difficult to solve. In particular, the arithmetic (geometric resp.)

mean of solution times increased from 911 seconds to 1868 seconds (174 seconds to 294 seconds resp.) and similar observations were made for the number of nodes in the branch-and-bound tree.

5.3 Expert Method Performance

Before benchmarking the performance of the learning heuristic, we begin by evaluating a theoretical *expert method* to gauge what potential speedups are achievable with this approach and to identify which MIPLIB 2017 instances could potentially benefit. Given a *test* instance, the *expert* method first runs the cut collection algorithm to generate GMI cuts, then invokes Gurobi to solve the instance, providing the cuts to the solver though a cut callback. Specifically, all collected cuts are simultaneously provided exactly once, when the cut callback is called for the first time. However, the time spent collecting the cuts is not counted towards the running time of the expert method, to simulate the ideal results that a "perfect" learning heuristic could achieve.

Eliminated Instances. While benchmarking the expert method, 48 additional were eliminated, mostly due to the cut collection procedure returning zero cuts, but also due to some memory/time limits and a few numerical errors. See Table 9 for a more detailed breakdown. Zero cuts are typically generated due to dual degeneracy, as it is possible that there is no improvement in dual bound after a single round of cuts and the dual multipliers corresponding to the cuts are then zero. In addition to the instances listed in Table 9, on three instances we observed numerical issues during the benchmark procedure after cut collection.

Cut Performance within Gurobi. On the remaining 135 instances, we evaluate the effectiveness of the cuts based on Gurobi's runtime, the number of branch-and-bound nodes, and work units, a deterministic measure reflecting time spent on optimization. For each of the five test variations and three random seeds, we solved each instance using Gurobi both with and without cuts, gathering average statistics. To classify instances with respect to cut performance, we calculated speedup as the ration between average work units with and without cuts. Instances were labeled *positive* if the average speedup was above 1.01x, *negative* if below 0.99x, and *neutral* otherwise. Our results showed that 50 instances were positive, 36 were neutral and 49 were negative. See Table 10 in the Appendix for more detailed results. These findings suggest that, *if we could ignore the computational cost of generating them*, these cuts could improve solver performance for a substantial number of MIPLIB 2017 instances. However, when accounting for the time required to collect the cuts, most positive instances become negative, which motivates our learning approach.

5.4 Learning Heuristic Performance

In this subsection, we evaluate the performance of the learning heuristic, focusing on the 50 benchmark instances in which the *expert* provided positive results.

Table 1: Average performance over all 50 benchmark instances.

Method	Time (s)	Work	Nodes	Cuts	Speedup Time	Speedup Work
ml:near:1	1,110.08	2,024.94	571,120.15	122.52	1.03	1.11
expert	1,154.88	1,924.89	701,039.03	122.17	1.02	1.20
baseline	1,219.13	2,035.39	719,464.26	—	1.00	1.00
ml:near:10	1,124.44	1,950.01	564,413.93	1,233.31	0.93	1.28
ml:far:10	1,059.20	1,849.97	560,983.46	1,237.80	0.91	1.32
ml:rand:10	1,111.68	1,918.12	583,215.51	1,235.16	0.91	1.30
ml:near:50	1,160.96	1,807.49	490,234.73	6,147.18	0.65	1.49
exp+col	1,556.93	1,924.89	701,039.03	122.17	0.46	1.20

Learning Setting. Recall that, for each of the 50 benchmark instances considered, we have 50 training and 5 test variations. During an offline training phase, we ran the cut collection method on all training variations and recorded the generated cuts. More specifically, for each generated cut, we stored the basis and corresponding tableau row that was used to generate the cut, which can be seen as an alternative method of storing the necessary constraint aggregation multipliers. During the test phase, as before, the cuts were generated and provided to Gurobi via a cut callback function. Also, as before, each problem was solved three times with different random seeds and average statistics were collected.

Evaluated Methods. As discussed in Section 4, we would like to test two aspects of our learning heuristics. First, the number of cuts to add, which, in our case, depends primarily on the number k of training variations from which to select aggregation multipliers. To cover a wide range of settings, while keeping the time required to run the computational experiments manageable, we chose $k = \{1, 10, 50\}$ variations. Second, the strategy for selecting training variations. Specifically, we considered: (i) k-closest training variations to the test instance, (ii) k-farthest variations, and (iii) k-random variations. Based on these criteria, we tested the following configurations: (i) 1-closest, (ii) 10-closest, (iii) 10-farthest, (iv) 10-random, and (v) 50-closest, effectively using all training variations to select aggregation multipliers. To measure instance similarity, each instance variation was represented by a feature vector consisting of right-hand-side values and objective coefficients and distances were measured using the 2-norm. Since these features can vary significantly in magnitude, we applied standard scaling by removing the mean and scaling to unit variance.

Table 1 summarizes the average performance of the evaluated methods across all 50 benchmark instances. For each method, *time* includes not only the time Gurobi requires to solve the problem, but also the additional time spent predicting multipliers, converting the test problem to standard form, computing multiple tableaux, and generating cuts before the optimization process begins. In contrast, *work units* measures only the computational effort spent by Gurobi,

Table 2: Average performance over 14 hard benchmark instances.

	Time (s)	Work	Nodes	Cuts	Speedup	
Method					Time	Work
ml:far:10	3,581.44	6,378.97	1,904,188.59	1,190.49	1.17	1.22
ml:rand:10	3,686.43	6,517.60	1,982,708.30	1,191.15	1.15	1.25
ml:near:10	3,826.53	6,742.74	1,915,780.40	1,192.53	1.11	1.12
expert	3,981.47	6,650.68	2,406,732.23	116.19	1.11	1.12
ml:near:50	3,648.58	6,240.77	1,652,977.92	5,847.93	1.08	1.40
ml:near:1	3,791.68	6,950.06	1,938,210.00	118.19	1.08	1.02
baseline	4,200.30	7,012.50	2,464,156.55	—	1.00	1.00
exp+col	4,568.36	6,650.68	2,406,732.23	116.19	0.91	1.12

in a deterministic way. The table also includes the average number of branch-and-bound nodes explored and average speedups for both time and work units. Speedups were calculated by first computing them for each test variation individually, then averaging these results across variations. The *baseline* row in the table refers to standard Gurobi, with an empty cut callback function. Additionally, since the *expert*'s time excludes cut collection time, we introduce an *exp+col* row, which adds cut collection time to the *time* column while keeping all other entries unchanged. Table 2 shows similar results, but focuses on 14 hard benchmark instances, defined as those requiring at least five minutes for default Gurobi to solve. More detailed results are presented in Tables 13, 14, 11, and 12, in the Appendix. We can make the following observations:

1. *All variations of the learning heuristic are quite effective at improving Gurobi's running time* (measured in work units), with the best variant achieving an average speedup of 1.49x across all 50 instances and a 1.40x speedup on the 14 hard instances. Every variation, including the simple 1-closest, outperformed Gurobi's default, and most surpassed even the expert method! With few exceptions, when the expert method performs well, the learning heuristics achieve similarly strong results.
2. *More cuts are typically better* (with respect to work units). Although there is typically a concern that providing too many cuts to a MILP solver may hinder its performance, we did not observe this issue in our experiments. In fact, the best performance was obtained by 50-closest, which adds cuts based on aggregation multipliers from all the training instances.
3. *Instance similarity does not appear to be a critical factor*. All three selection strategies (closest, farthest and random) showed similar performance, as long as cuts were generated from 10 training variations. If anything, 10-farthest performed the best in our experiments among these three strategies.
4. *The heuristic performs well even when right-hand-side (RHS) values are perturbed*. When the RHS remains unchanged, one may expect the learning heuristics to perform well, as the feasible region of the problems remain

the same. However, the real test is how the heuristic handles varying RHS values. Of the 50 instances tested, 20 included RHS perturbations, and we observed similar speedup trends in these 20 instances as in the full set of 50. Specifically, the arithmetic average of work units for baseline, expert, 10-farthest, 1-closest, 10-closest, 50-closest, and 10-random is 2535.55, 2414.54, 2242.60, 2312.10, 2343.43, 2177.83, and 2327.89 respectively.

5. *The learning heuristic is effective even when considering total running time, particularly for challenging instances.* The previous discussion points focus on work units, which measure only the effort spent by Gurobi. When we also include the computational time required for predicting and generating cuts, some new insights emerge. First, adding cut collection time to the expert method completely nullifies its running time improvements, with an average speedup of just 0.46x across all 50 benchmark instances. Even for hard instances, the *exp+col* method still underperforms the baseline, with an average speedup of 0.91x. Second, although the 50-closest variant provides the best results in terms of work units, it is less efficient in total running time than other methods, due to the extensive data processing it requires, such as reading additional training data files and computing tableaux for more linear programming bases. Methods with lower preprocessing demands, like 10-farthest and 10-random, offer a more balanced performance. Nonetheless, all learning heuristic variants outperform the baseline on hard instances, with time speedups between 1.08x and 1.17x. Third, our cut generation implementation still suffers from significant overhead, limiting the effectiveness of methods like 10-farthest to hard instances only. In fact, the only method able to achieve a positive average time speedup across all 50 benchmark instances is 1-closest, which requires the least preprocessing. Much of this overhead comes from recomputing the problem's standard form and multiple tableaux, as we lack access to solver internals containing this information. If these methods were directly integrated into Gurobi, we anticipate much lower overhead, making the method potentially useful even for easy instances.

6 Conclusion and Future Research

In this work, we demonstrated that historical data can be effectively used to generate strong cutting planes and accelerate the performance of a commercial state-of-the-art MILP solver. There are several open theoretical and computational research directions. On the theory side, there are questions regarding generalizing Theorem 1 (and similar result for split cuts with the same proof; see Theorem 4 in Appendix). We have established finite list of aggregation multipliers only when the right-hand-side of the instance family lives in a lattice generated by rational vectors. What happens when we consider all right-hand-sides is an open question. Finally, examining extensions of such results for more generals families of cut from the simplex tableau, the so-called group cuts [44] is another interesting direction of research.

On the computational side, we have seen that it is possible to get substantial improvements over default Gurobi, just by re-using the good aggregation multipliers determined for historical instances. This results could perhaps be improved by examining some interesting extensions. First, with more computational power and effort, it may be possible to work with a better expert, which for example generates cuts even from infeasible basis. Second, in our study we have explored changing the right-side within a margin of $\pm 10\%$ and the objective coefficient by $\pm 25\%$. Perhaps, it is possible to explore more dramatic changes.

Acknowledgments

Argonne National Laboratory's work was supported by the U.S. Department of Energy, Office of Electricity, under contract DE-AC02-06CH11357. Santanu S. Dey would like to gratefully acknowledge the support of ONR grant #N000142212632.

References

1. Tobias Achterberg, Thorsten Koch, and Alexander Martin. Miplib 2003. *Operations Research Letters*, 34(4):361–372, 2006.
2. Alejandro Marcos Alvarez, Quentin Louveaux, and Louis Wehenkel. A machine learning-based approximation of strong branching. *INFORMS Journal on Computing*, 29(1):185–195, 2017.
3. Kent Andersen, Gérard Cornuéjols, and Yanjun Li. Split closure and intersection cuts. *Mathematical programming*, 102(3):457–493, 2005.
4. 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.
5. Gennadiy Averkov. On finitely generated closures in the theory of cutting planes. *Discrete Optimization*, 9(4):209–215, 2012.
6. Egon Balas. Disjunctive programming. *Annals of discrete mathematics*, 5:3–51, 1979.
7. Egon Balas, Sebastian Ceria, Gérard Cornuéjols, and N Natraj. Gomory cuts revisited. *Operations Research Letters*, 19(1):1–9, 1996.
8. Egon Balas and Anureet Saxena. Optimizing over the split closure. *Mathematical Programming*, 113(2):219–240, 2008.
9. Maria-Florina F Balcan, Siddharth Prasad, Tuomas Sandholm, and Ellen Vitercik. Structural analysis of branch-and-cut and the learnability of gomory mixed integer cuts. *Advances in Neural Information Processing Systems*, 35:33890–33903, 2022.
10. Timo Berthold, Matteo Francobaldi, and Gregor Hendel. Learning to use local cuts. *arXiv preprint arXiv:2206.11618*, 2022.
11. Dimitris Bertsimas and Bartolomeo Stellato. Online mixed-integer optimization in milliseconds. *INFORMS Journal on Computing*, 34(4):2229–2248, 2022.
12. Robert E Bixby. Solving real-world linear programs: A decade and more of progress. *Operations research*, 50(1):3–15, 2002.
13. Merve Bodur, Alberto Del Pia, Santanu S Dey, Marco Molinaro, and Sebastian Pokutta. Aggregation-based cutting-planes for packing and covering integer programs. *Mathematical Programming*, 171:331–359, 2018.
14. Alberto Caprara and Adam N Letchford. On the separation of split cuts and related inequalities. *Mathematical Programming*, 94:279–294, 2003.

15. Hongyu Cheng and Amitabh Basu. Learning cut generating functions for integer programming. *arXiv preprint arXiv:2405.13992*, 2024.
16. Hongyu Cheng, Sammy Khalife, Barbara Fiedorowicz, and Amitabh Basu. Sample complexity of algorithm selection using neural networks and its applications to branch-and-cut. *arXiv preprint https://arxiv.org/abs/2402.02328*, 2024.
17. Didier Chételat and Andrea Lodi. Continuous cutting plane algorithms in integer programming. *Operations Research Letters*, 51(4):439–445, 2023.
18. Diego Cifuentes, Santanu S Dey, and Jingye Xu. Sensitivity analysis for mixed binary quadratic programming. In *International Conference on Integer Programming and Combinatorial Optimization*, pages 446–459. Springer, 2024.
19. Michele Conforti, Gérard Cornuéjols, Giacomo Zambelli, Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Integer programming*. Springer, 2014.
20. William Cook, Ravindran Kannan, and Alexander Schrijver. Chvátal closures for mixed integer programming problems. *Mathematical Programming*, 47(1):155–174, 1990.
21. Gérard Cornuéjols et al. Revival of the gomory cuts in the 1990's. *Annals of Operations Research*, 149(1):63–66, 2007.
22. Gérard Cornuéjols, François Margot, and Giacomo Nannicini. On the safety of gomory cut generators. *Mathematical Programming Computation*, 5:345–395, 2013.
23. Sanjeeb Dash and Marcos Goycoolea. A heuristic to generate rank-1 gmi cuts. *Mathematical Programming Computation*, 2(3-4):231–257, 2010.
24. Sanjeeb Dash, Oktay Günlük, and Andrea Lodi. Mir closures of polyhedral sets. *Mathematical Programming*, 121:33–60, 2010.
25. Sanjeeb Dash, Oktay Günlük, and Diego A Morán R. On the polyhedrality of cross and quadrilateral closures. *Mathematical Programming*, 160:245–270, 2016.
26. Santanu S Dey and Marco Molinaro. Theoretical challenges towards cutting-plane selection. *Mathematical Programming*, 170:237–266, 2018.
27. Arnaud Deza and Elias B Khalil. Machine learning for cutting planes in integer programming: A survey. *arXiv preprint arXiv:2302.09166*, 2023.
28. Gabriele Dragotto, Stefan Clarke, Jaime Fernández Fisac, and Bartolomeo Stellato. Differentiable cutting-plane layers for mixed-integer linear optimization. *arXiv preprint arXiv:2311.03350*, 2023.
29. Matteo Fischetti and Domenico Salvagnin. A relax-and-cut framework for gomory mixed-integer cuts. *Mathematical Programming Computation*, 3:79–102, 2011.
30. Matteo Fischetti and Domenico Salvagnin. Approximating the split closure. *INFORMS Journal on Computing*, 25(4):808–819, 2013.
31. 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*, 13(3):443–490, 2021.
32. Ralph Gomory and RAND CORP SANTA MONICA CA. *An Algorithm for the Mixed Integer Problem: Notes Linear Programming and Extensions-Part 54*. Rand, 1960.
33. Ralph E Gomory. An algorithm for integer solutions to linear programs. *Recent advances in mathematical programming*, 64(260-302):14, 1963.
34. Oscar Guaje, Arnaud Deza, Aleksandr M Kazachkov, and Elias B Khalil. Machine learning for optimization-based separation: the case of mixed-integer rounding cuts. *arXiv preprint arXiv:2408.08449*, 2024.

35. Prateek Gupta, Maxime Gasse, Elias Khalil, Pawan Mudigonda, Andrea Lodi, and Yoshua Bengio. Hybrid models for learning to branch. *Advances in neural information processing systems*, 33:18087–18097, 2020.
36. 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. *Pattern Recognition*, 123:108353, 2022.
37. Emma S Johnson, Shabbir Ahmed, Santanu S Dey, and Jean-Paul Watson. A k-nearest neighbor heuristic for real-time dc optimal transmission switching. *arXiv preprint arXiv:2003.10565*, 2020.
38. Burak Kocuk, Hyemin Jeon, Santanu S Dey, Jeff Linderoth, James Luedtke, and Xu Andy Sun. A cycle-based formulation and valid inequalities for dc power transmission problems with switching. *Operations Research*, 64(4):922–938, 2016.
39. Dabeen Lee. On the np-hardness of deciding emptiness of the split closure of a rational polytope in the 0, 1 hypercube. *Discrete Optimization*, 32:11–18, 2019.
40. Hugues Marchand and Laurence A Wolsey. Aggregation and mixed integer rounding to solve mips. *Operations research*, 49(3):363–371, 2001.
41. 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.
42. George L Nemhauser and Laurence A Wolsey. A recursive procedure to generate all cuts for 0–1 mixed integer programs. *Mathematical Programming*, 46(1):379–390, 1990.
43. Max B Paulus, Giulia Zarpellon, Andreas Krause, Laurent Charlin, and Chris Maddison. Learning to cut by looking ahead: Cutting plane selection via imitation learning. In *International conference on machine learning*, pages 17584–17600. PMLR, 2022.
44. Jean-Philippe P Richard and Santanu S Dey. The group-theoretic approach in mixed integer programming. *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 727–801, 2010.
45. Álinson Santos Xavier, Feng Qiu, Xiaoyi Gu, Berkay Becu, and Santanu S. Dey. MIPLearn: An Extensible Framework for Learning- Enhanced Optimization, February 2024.
46. Alexander Schrijver. *Theory of linear and integer programming.* John Wiley & Sons, 1998.
47. 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.
48. Mark Turner, Thorsten Koch, Felipe Serrano, and Michael Winkler. Adaptive cut selection in mixed-integer linear programming. *arXiv preprint arXiv:2202.10962*, 2022.
49. Juan Pablo Vielma. A constructive characterization of the split closure of a mixed integer linear program. *Operations Research Letters*, 35(1):29–35, 2007.
50. Zhihai Wang, Xijun Li, Jie Wang, Yufei Kuang, Mingxuan Yuan, Jia Zeng, Yongdong Zhang, and Feng Wu. Learning cut selection for mixed-integer linear programming via hierarchical sequence model. *arXiv preprint arXiv:2302.00244*, 2023.
51. Laurence A Wolsey. The b-hull of an integer program. *Discrete Applied Mathematics*, 3(3):193–201, 1981.
52. Álinson S Xavier, Feng Qiu, and Shabbir Ahmed. Learning to solve large-scale security-constrained unit commitment problems. *INFORMS Journal on Computing*, 33(2):739–756, 2021.

53. Gu Xiaoyi, S Dey, ÁS Xavier, et al. Exploiting instance and variable similarity to improve learning-enhanced branching. *arXiv*, 2022.

A Proofs

Given a vector $u \in \mathbb{R}^k$, let $u^+, u^- \in \mathbb{R}^k_+$ be the vectors defined as:

$$u_i^+ = \begin{cases} u_i \text{ if } u_i \geq 0 \\ 0 \text{ if } u_i \leq 0. \end{cases} \quad u_i^- = \begin{cases} -u_i \text{ if } u_i < 0 \\ 0 \text{ if } u_i \geq 0. \end{cases}$$

Given two inequalities

$$\eta^1 x \geq 1, \tag{6}$$

and

$$\eta^2 x \geq 1 \tag{7}$$

valid for IP presented in (2), we say that (6) dominates (7) if $\eta^1 \leq \eta^2$. This is because all the variables in IP are non-negative.

As discussed in Section 2, split cuts and GMI cuts are very closely related. Our presentation in this section is from Section 5.1 in [19].

Split cuts for IP*.* Considering a mixed integer set IP in standard form as described in (2), any non-dominated split cut for IP can be generated as follows. Given a vector $(\lambda, v) \in \mathbb{R}^m \times \mathbb{R}^n$, we will say it is *legitimate* if it satisfies the following three conditions:

(1.) $\lambda^\top A_j - v_j \in \mathbb{Z}$ for $j \in J$,
(2.) $\lambda^\top A_j - v_j = 0$ for $j \in [n] \setminus J$,
(3.) $\lambda^\top b \notin \mathbb{Z}$, where $f = \lambda^\top b - \lfloor \lambda^\top b \rfloor$.

For a legitimate $(\lambda, v) \in \mathbb{R}^m \times \mathbb{R}^n$, the inequality

$$\frac{v^+ x}{f} + \frac{v^- x}{1-f} \geq 1, \tag{8}$$

is a split inequality valid for IP defined in (2) obtained using the disjunction

$$(\pi^\top x \leq \pi_0) \vee (\pi^\top x \geq \pi_0 + 1),$$

where $\pi = \lambda^\top A - v$ and $\pi_0 = \lfloor \lambda^\top b \rfloor$.

Connection to GMICs. Using the fact that variables in IP given in (2) are non-negative, one can easily show the following: (Section 5.1.4 in [19]) Given a fixed $\hat{\lambda} \in \mathbb{R}^m$ with $\hat{\lambda}^\top b \notin \mathbb{Z}$, one can find one optimal value of $\hat{v}$ such that $(\hat{\lambda}, \hat{v})$ is legitimate, where optimality implies that an inequality of the form (8) corresponding to any legitimate pair $(\hat{\lambda}, v)$ is dominated by the inequality (8) corresponding to $(\hat{\lambda}, \hat{v})$. This inequality corresponding to $(\hat{\lambda}, \hat{v})$ is precisely the GMIC as described in (1) with $\alpha_j = \hat{\lambda}^\top A_j$ for $j \in [n]$ and $\beta = \hat{\lambda}^\top b$, that is the inequality $GMIC(\text{IP}_{\hat{\lambda}})$. Thus, all the rank-1 GMICs for the set IP are special case of (8), where the v vectors is selected optimally for a given λ.

We next show that by a argument similar to that presented in [24] (our presentation is based on [19]), it is possible to prove that there is a *finite set of aggregation multipliers* which yield the GMI closure for an infinite family of instances with varying right-hand-sides. In order to present this result we need the notation presented below.

We begin with a result presented in [19], originally shown in [3], before presenting our main result.

Lemma 1. *(Corollary 5.6 in [19]) If the inequality (8) corresponding to a* (λ, v) *is undominated among all possible inequalities of the form (8), then the support of* (λ, v) *corresponds to a basis, that is, if* $\lambda_{i_1}, \ldots, \lambda_{i_k}$ *are non-zero and* $v_{j_1}, \ldots, v_{j_l}$ *are non-zero, then* $\{a_{i_1}, \ldots, a_{i_k}, e_{j_1}, \ldots, e_{j_l}\} \in \mathbb{R}^n$ *are linearly independent, where* a_i *is the* i^{th} *row of* A *and* e_j *is the* j^{th} *standard basis vector.*

Theorem 2. *Let* Γ *be the lattice generated by rational vectors* $b^1, \ldots, b^k \in \mathbb{Q}^m$. *Consider the infinite family of instances corresponding to* Γ *as described in (5). Then there exists a finite set* $\Lambda \subseteq \mathbb{R}^m$, *such that the GMIC closure of every instance* $\text{IP}(\gamma)$ *can be obtained using aggregation multipliers in* Λ, *that is:*

$$\mathcal{G}(\text{IP}(\gamma)) = \bigcap_{\lambda \in \Lambda} \text{GMIC}\left(\text{IP}(\gamma)_\lambda\right) \quad \forall \gamma \in \Gamma.$$

Proof. We will show that there exists a finite set $\tilde{\Lambda} = \{(\lambda^1, v^1), \ldots, (\lambda^g, v^g)\}$ with its elements satisfying the first two conditions of being legitimate (if third is not satisfied for some $b + \gamma$, then no cut using this λ is generated for this instance) and satisfying Lemma 1, such that any inequality (8) generated from any legitimate (λ, v) is dominated by an inequality (8) corresponding to (λ^i, v^i) for some $i \in [g]$. Let the projection of $\tilde{\Lambda}$ onto the λ-component be the finite set Λ. As discussed above, given a fixed $\hat{\lambda}$, we can then select a corresponding $\hat{v}$ to this $\hat{\lambda}$, which leads to an inequality of the type (8) that dominates all other inequalities of the type (8) corresponding to legitimate vector $(\hat{\lambda}, v)$ – this is how all GMICs are generated. Thus, this shows that the GMIC closure is obtained by generating the GMICs corresponding to the elements in the finite list Λ.

Let $u = (\lambda, v)$ be a legitimate vector with a support corresponding to a basis (from Lemma 1) for an instance $\text{IP}(\gamma)$, that is it satisfies (1.), (2.) and $\lambda^\top(b + \gamma) \notin \mathbb{Z}$.

We define the cone

$$\mathcal{C}_{\text{sign}(u)} = \left\{ w = (\theta, \phi) \in \mathbb{R}^m \times \mathbb{R}^n \;\middle|\; \begin{array}{r} \theta^\top A_j - \phi_j = 0 \ \forall j \in [n] \setminus J \\ w_j = 0 \text{ if } u_j = 0 \\ w_j \leq 0 \text{ if } u_j < 0 \\ w_j \geq 0 \text{ if } u_j > 0. \end{array} \right\}.$$

We claim that if there exists $u^1 = (\theta^1, \phi^1), u^2 = (\theta^2, \phi^2) \in \mathcal{C}_{\mathrm{sign}(u)}$ such that $u = u^1 + u^2$, where $u^2 \in \mathbb{Z}^{n+m}$ and $(\theta^2)^\top (b+\gamma) \in \mathbb{Z}$, then the inequality corresponding to u is dominated by the inequality corresponding to u^1. First we verify that u^1 is legitimate that is it satisfies conditions (1.) - (3.). Note that since A is an integral matrix, and $(\theta^2, \phi^2_j) \in \mathbb{Z}^m \times \mathbb{Z}$, for $j \in J$, we have that $\mathbb{Z} \ni \lambda^\top A_j - v_j = (\theta^1 + \theta^2)^\top A_j - \phi^1_j - \phi^2_2$ implies that $(\phi^1)^\top A_j - v_j \in \mathbb{Z}$, that is condition (1.) holds. Also because $u^1 \in C_u$, we have that u^1 satisfies condition (2.). Finally, note that $\lambda^\top (b+\gamma) = (\theta^1 + \theta^2)^\top (b+\gamma)$ and by assumption $(\theta^2)^\top (b+\gamma) \in \mathbb{Z}$, we have $(\theta^1)^\top (b+\gamma) \equiv \lambda^\top (b+\gamma)(\mathrm{mod} 1) \equiv f(\mathrm{mod} 1)$, that is $(\theta^1)^\top (b+\gamma) \notin \mathbb{Z}$, thus u^1 satisfies condition (3.). Finally note that due to constraints defining $\mathcal{C}_{\mathrm{sign}(u)}$, and because $u = u^1 + u^2$, we have that $(\phi^1)^+ \leq v^+$ and $(\phi^1)^- \leq v^-$. Therefore, the inequality corresponding to u^1:

$$\frac{((\phi^1)^+)^\top x}{f} + \frac{((\phi^1)^-)^\top x}{1-f} \geq 1,$$

dominates the inequality corresponding to $u = (\lambda, v)$.

Let $b^1, \ldots, b^k$ be the generators of Γ. Let $M(\Gamma) \in \mathbb{Z}_+$ be the smallest positive integer such that $M(\Gamma) \cdot b^i \in \mathbb{Z}^m$ for all $i \in [k]$. Note that this implies that $M(\Gamma) \cdot \gamma \in \mathbb{Z}^m$ for all $\gamma \in \Gamma$. Let Δ be the maximum among the absolute value of all sub-determinant of the matrix corresponding to the left-hand-side of constraints defining $\mathcal{C}_{\mathrm{sign}(u)}$. Let $H := (m+n) \cdot M(\Gamma) \cdot \Delta$.

We next show that for all legitimate $u \in \mathbb{R}^{m+n}$ there exists $u^1, u^2 \in \mathcal{C}_{\mathrm{sign}(u)}$ such that $u = u^1 + u^2$ and $u^2 \in \mathbb{Z}^{m+n}$, $(\theta^2)^\top (b+\gamma) \in \mathbb{Z}$ for all $\gamma \in \Gamma$, where u^1 satisfies $-H \leq u^1_j \leq H$ for all $j \in [m+n]$ and u^1s support is contained in the support of u. The key to this proof depends on the fact that $\mathcal{C}_{\mathrm{sign}(u)}$ does not depend on the right-hand-side and that $M(\Gamma) \cdot \gamma \in \mathbb{Z}^m$ for all $\gamma \in \Gamma$. First note that $\mathcal{C}_{\mathrm{sign}(u)}$ is a pointed cone and let $r^1, \ldots, r^k$ be the extreme rays of $\mathcal{C}_{\mathrm{sign}(u)}$. By standard arguments (see Chapter 3 in [46]) we can re-scale $r^1, \ldots, r^k$ such that they are integral vectors and $\|r^t\|_\infty \leq \Delta$ for all $t \in [k]$. Let us now further re-scale these vectors by $M(\Gamma)$, that is we assume:

- $r^t \in \mathbb{Z}^{m+n}$ for all $t \in [k]$,
- Each entry of r^t is divisible by $M(\Gamma)$ for all $t \in [k]$,
- $\|r^t\|_\infty \leq M(\Gamma) \cdot \Delta$.

Now since $u \in \mathcal{C}_{\mathrm{sign}(u)}$, we have that $u = \sum_{t=1}^k \psi_t r^t$, where by Carathéodory's Theorem at most $m+n$ of the ψs are positive. Let $u^1 = \sum_{t=1}^k (\psi_t - \lfloor \psi_t \rfloor) \cdot r^t$ and let $u^2 = \sum_{t=1}^k \lfloor \psi_t \rfloor \cdot r^t$. Then clearly $u^1, u^2 \in \mathcal{C}_{\mathrm{sign}(u)}$, $u^2 \in \mathbb{Z}^{m+n}$ and each entry of u^2 is divisible by $M(\Gamma)$ for all $\gamma \in \Gamma$. Thus $(\theta^2)^\top (b+\gamma) \in \mathbb{Z}$. Finally, since at most $n+m$ of ψ_ts are positive, we have that $\|u^1\|_\infty \leq (m+n) \cdot M(\gamma) \cdot \Delta = H$.

Based on the results above, we conclude that non-dominated inequalities (8) for all instances IP(γ) for $\gamma \in \Gamma$ are generated corresponding to legitimate vectors $u = (\lambda, v)$ with a support corresponding to a basis and $\|u\|_\infty \leq H$. Therefore, we obtain that the set:

$$\Pi := \{\pi \in \mathbb{Z}^n \,|\, \pi = \lambda^\top A - v^\top I, \pi_j = 0 \ \forall j \in [n] \setminus J, \|\lambda\|_\infty \leq H, \|v\|_\infty \leq H\},$$

is finite. Thus, all the non-dominated split cuts (8) can be generated using the split disjunctions whose left-hand-side belong to the Π.

Finally, note that for a given $\pi \in \Pi$, the solution to the system $\pi = \lambda^\top A - u^\top I$ where the support of (λ, u) corresponds to a basis is a unique solution. Thus the number of non-dominated inequalities of the form (8) can be obtained from a finite list of $u = (\lambda, v)$s. This completes the proof.

The proof above closely follows arguments presented in [19].

Theorem 3. *Let Γ be the lattice generated by rational vectors $b^1, \ldots, b^k \in \mathbb{Q}^m$. Consider the infinite family of instances corresponding to Γ as described in (5). Then there exists a finite set $\Lambda \subseteq \mathbb{R}^m$, such that the GMIC closure of every instance* $\mathrm{IP}(\gamma)$ *can be obtained using aggregation multipliers in Λ, that is:*

$$\mathcal{G}(\mathrm{IP}(\gamma)) = \bigcap_{\lambda \in \Lambda} \mathrm{GMIC}\left(\mathrm{IP}(\gamma)_\lambda\right) \quad \forall \gamma \in \Gamma.$$

Proof. We will show that there exists a finite set $\tilde{\Lambda} = \{(\lambda^1, v^1), \ldots, (\lambda^g, v^g)\}$ with its elements satisfying the first two conditions of being legitimate (if third is not satisfied for some $b + \gamma$, then no cut using this λ is generated for this instance) and satisfying Lemma 1, such that any inequality (8) generated from any legitimate (λ, v) is dominated by an inequality (8) corresponding to (λ^i, v^i) for some $i \in [g]$. Let the projection of $\tilde{\Lambda}$ onto the λ-component be the finite set Λ. As discussed above, given a fixed $\hat{\lambda}$, we can then select a corresponding $\hat{v}$ to this $\hat{\lambda}$, which leads to an inequality of the type (8) that dominates all other inequalities of the type (8) corresponding to legitimate vector $(\hat{\lambda}, v)$ – this is how all GMICs are generated. Thus, this shows that the GMIC closure is obtained by generating the GMICs corresponding to the elements in the finite list Λ.

Let $u = (\lambda, v)$ be a legitimate vector with a support corresponding to a basis (from Lemma 1) for an instance $\mathrm{IP}(\gamma)$, that is it satisfies (1.), (2.) and $\lambda^\top (b + \gamma) \notin \mathbb{Z}$.

We define the cone

$$\mathcal{C}_{\mathrm{sign}(u)} = \left\{ w = (\theta, \phi) \in \mathbb{R}^m \times \mathbb{R}^n \;\middle|\; \begin{array}{r} \theta^\top A_j - \phi_j = 0 \ \forall j \in [n] \setminus J \\ w_j = 0 \text{ if } u_j = 0 \\ w_j \leq 0 \text{ if } u_j < 0 \\ w_j \geq 0 \text{ if } u_j > 0. \end{array} \right\}.$$

We claim that if there exists $u^1 = (\theta^1, \phi^1), u^2 = (\theta^2, \phi^2) \in \mathcal{C}_{\mathrm{sign}(u)}$ such that $u = u^1 + u^2$, where $u^2 \in \mathbb{Z}^{n+m}$ and $(\theta^2)^\top (b + \gamma) \in \mathbb{Z}$, then the inequality corresponding to u is dominated by the inequality corresponding to u^1. First we verify that u^1 is legitimate that is it satisfies conditions (1.) - (3.). Note that since A is an integral matrix, and $(\theta^2, \phi^2_j) \in \mathbb{Z}^m \times \mathbb{Z}$, for $j \in J$, we have that $\mathbb{Z} \ni \lambda^\top A_j - v_j = (\theta^1 + \theta^2)^\top A_j - \phi^1_j - \phi^2_2$ implies that $(\phi^1)^\top A_j - v_j \in \mathbb{Z}$, that is condition (1.) holds. Also because $u^1 \in C_u$, we have that u^1 satisfies condition (2.). Finally, note that $\lambda^\top (b+\gamma) = (\theta^1 + \theta^2)^\top (b+\gamma)$ and by assumption $(\theta^2)^\top (b + \gamma) \in \mathbb{Z}$, we have $(\theta^1)^\top (b + \gamma) \equiv \lambda^\top (b + \gamma)(\mathrm{mod} 1) \equiv f(\mathrm{mod} 1)$, that is $(\theta^1)^\top(b+\gamma) \notin \mathbb{Z}$, thus u^1 satisfies condition (3.). Finally note that due to constraints defining $\mathcal{C}_{\text{sign}(u)}$, and because $u = u^1 + u^2$, we have that $(\phi^1)^+ \leq v^+$ and $(\phi^1)^- \leq v^-$. Therefore, the inequality corresponding to u^1:

$$\frac{((\phi^1)^+)^\top x}{f} + \frac{((\phi^1)^-)^\top x}{1-f} \geq 1,$$

dominates the inequality corresponding to $u = (\lambda, v)$.

Let $b^1, \ldots, b^k$ be the generators of Γ. Let $M(\Gamma) \in \mathbb{Z}_+$ be the smallest positive integer such that $M(\Gamma) \cdot b^i \in \mathbb{Z}^m$ for all $i \in [k]$. Note that this implies that $M(\Gamma) \cdot \gamma \in \mathbb{Z}^m$ for all $\gamma \in \Gamma$. Let Δ be the maximum among the absolute value of all sub-determinant of the matrix corresponding to the left-hand-side of constraints defining $\mathcal{C}_{\text{sign}(u)}$. Let $H := (m+n) \cdot M(\Gamma) \cdot \Delta$.

We next show that for all legitimate $u \in \mathbb{R}^{m+n}$ there exists $u^1, u^2 \in \mathcal{C}_{\text{sign}(u)}$ such that $u = u^1 + u^2$ and $u^2 \in \mathbb{Z}^{m+n}$, $(\theta^2)^\top(b+\gamma) \in \mathbb{Z}$ for all $\gamma \in \Gamma$, where u^1 satisfies $-H \leq u^1_j \leq H$ for all $j \in [m+n]$ and u^1s support is contained in the support of u. The key to this proof depends on the fact that $\mathcal{C}_{\text{sign}(u)}$ does not depend on the right-hand-side and that $M(\Gamma) \cdot \gamma \in \mathbb{Z}^m$ for all $\gamma \in \Gamma$. First note that $\mathcal{C}_{\text{sign}(u)}$ is a pointed cone and let $r^1, \ldots, r^k$ be the extreme rays of $\mathcal{C}_{\text{sign}(u)}$. By standard arguments (see Chapter 3 in [46]) we can re-scale $r^1, \ldots, r^k$ such that they are integral vectors and $\|r^t\|_\infty \leq \Delta$ for all $t \in [k]$. Let us now further re-scale these vectors by $M(\Gamma)$, that is we assume:

- $r^t \in \mathbb{Z}^{m+n}$ for all $t \in [k]$,
- Each entry of r^t is divisible by $M(\Gamma)$ for all $t \in [k]$,
- $\|r^t\|_\infty \leq M(\Gamma) \cdot \Delta$.

Now since $u \in \mathcal{C}_{\text{sign}(u)}$, we have that $u = \sum_{t=1}^k \psi_t r^t$, where by Carathéodory's Theorem at most $m+n$ of the ψs are positive. Let $u^1 = \sum_{t=1}^k (\psi_t - \lfloor \psi_t \rfloor) \cdot r^t$ and let $u^2 = \sum_{t=1}^k \lfloor \psi_t \rfloor \cdot r^t$. Then clearly $u^1, u^2 \in \mathcal{C}_{\text{sign}(u)}$, $u^2 \in \mathbb{Z}^{m+n}$ and each entry of u^2 is divisible by $M(\Gamma)$ for all $\gamma \in \Gamma$. Thus $(\theta^2)^\top(b+\gamma) \in \mathbb{Z}$. Finally, since at most $n+m$ of ψ_ts are positive, we have that $\|u^1\|_\infty \leq (m+n) \cdot M(\gamma) \cdot \Delta = H$.

Based on the results above, we conclude that non-dominated inequalities (8) for all instances IP(γ) for $\gamma \in \Gamma$ are generated corresponding to legitimate vectors $u = (\lambda, v)$ with a support corresponding to a basis and $\|u\|_\infty \leq H$. Therefore, we obtain that the set:

$$\Pi := \{\pi \in \mathbb{Z}^n \,|\, \pi = \lambda^\top A - v^\top I, \pi_j = 0 \; \forall j \in [n] \setminus J, \|\lambda\|_\infty \leq H, \|v\|_\infty \leq H\},$$

is finite. Thus, all the non-dominated split cuts (8) can be generated using the split disjunctions whose left-hand-side belong to the Π.

Finally, note that for a given $\pi \in \Pi$, the solution to the system $\pi = \lambda^\top A - u^\top I$ where the support of (λ, u) corresponds to a basis is a unique solution. Thus the number of non-dominated inequalities of the form (8) can be obtained from a finite list of $u = (\lambda, v)$s. This completes the proof.

The proof above closely follows arguments presented in [19].

Corollary 2. *Consider* $\mathrm{IP}(\gamma)$ *as defined in (5), where we allow* $\gamma \in \mathbb{R}^m$*. Let* $J = [n]$*, that is all variables are integral. Then there exists a finite set of multipliers* Γ *such that*

$$\mathcal{G}(\mathrm{IP}(\gamma)) = \bigcap_{\lambda \in \Lambda} \mathrm{GMIC}\left(\mathrm{IP}(\gamma)_\lambda\right) \quad \forall \gamma \in \mathbb{R}^m.$$

Proof. Let Δ^1 be the finite set obtained from Theorem 1 where $\Gamma = \mathbb{Z}^m$, that is Γ is generated by $e^1, e^2, \ldots, e^m$. Also let $\Delta^2 := \{e^1, e^2, \ldots, e^m\}$. Let $\Delta = \Delta^1 \cup \Delta^2$. We consider two cases:

- $b_i + \gamma_i \notin \mathbb{Z}$ for some $i \in [m]$. In that case, the integer program is infeasible and we obtain the same conclusion by applying the GMIC cut to the i^{th} constraint. This is the GMIC corresponding to the aggregation multiplier $\lambda = e^i$. Thus, this GMIC is from the list Δ of aggregation multipliers, which gives the integer hull (and thus the GMIC closure).
- If $b_i + \gamma_i \in \mathbb{Z}$ for all $i \in [m]$, then the result follows from Theorem 1.

Using techniques similar to the proof of Theorem 1, we can show a similar result about split closures where we define split closure (in an equivalent way) using only the left-hand-sides of the disjunctions [20]:

Theorem 4. *Let* Γ *be the lattice generated by rational vectors* $b^1, \ldots, b^k \in \mathbb{Q}^m$*. Let* $P(\gamma) := \{x \in \mathbb{R}^n \mid Ax \leq b + \gamma\}$ *for* $\gamma \in \Gamma$ *be a family of rational polyhedron and we are interested in split closures for* $Q(\gamma) = P(\gamma) \cap \{x \mid x_j \in \mathbb{Z}, j \in J\}$*. Let* $P(\gamma)^\pi = \mathrm{conv}(P(\gamma) \cap \{x \in \mathbb{R}^n \mid \pi^\top x \in \mathbb{Z}\})$*, where* $\pi \in \mathbb{Z}^n$ *and the support of* π *belongs to* J*. Then there exists a finite set* $\Pi := \{\pi^1, \ldots, \pi^g\}$*, with* $\pi^i \in \mathbb{Z}^n, \pi^i_j = 0 \ \forall j \notin J$ *for all* $i \in [g]$*, such that:*

$$\text{split closure}(P(\gamma)) := \bigcap_{\pi \in \mathbb{Z}^n, \pi_j = 0, j \notin J} P(\gamma)^\pi = \bigcap_{\pi \in \Pi} P(\gamma)^\pi, \ \forall \gamma \in \Gamma.$$

B Cut Collection Algorithm

For approximating the GMIC closure, we modified the relax-and-cut approach developed by Fischetti and Salvagnin [29]. The relax-and-cut approach is a method to generate a large number of simplex tableau corresponding to different basic feasible solutions which yield useful GMICs.

In our implementation, we considerably simplified the algorithm from that originally proposed in [29]. Let $\mathrm{C}(B)$ denote the set of GMICs collected and retained from a tableau associated with a particular basis, B, of LP-relaxation of (2). That is

$$\mathrm{C}(B) := \left\{(\alpha^j_B)^\top x \geq \alpha^j_{0B}, \quad j = 1, \cdots, q(B)\right\}.$$

Our method iteratively goes through the following steps:

1. Assuming we have already visited basis $B^1, \dots, B^k$, we have thus collected and retained the cuts in the set $\bigcup_{l=1}^{k} \mathrm{C}(B^l)$. Let $c^\top x$ be the objective function of our IP. We solve the LP where we add all these GMICs, that is we solve:

$$\begin{aligned} \min\ & c^\top x \\ & Ax = b, x \geq 0, \\ & (\alpha^j_{B^l})^\top x \geq \alpha^j_{0B^l}\ \forall j = 1, \dots, q(B^l),\ \forall l \in [k]. \end{aligned} \tag{9}$$

2. Let $u^j_{B^l}$ be an optimal dual solution of the above LP corresponding to the constraint $(\alpha^j_{B^l})^\top x \geq \alpha^j_{0B^l}$. We now construct the following Lagrangian relaxation to the previous LP:

$$\begin{aligned} \min\ & c^\top x + \textstyle\sum_{l\in[k]} \sum_{j\in q(B^l)} u^j_{B^l} (\alpha^j_{0B^l} - (\alpha^j_{B^l})^\top x) \\ & Ax = b, x \geq 0. \end{aligned} \tag{10}$$

   Notice that the optimal objective function value of both the above LPs is equal.
3. We solve (10), and let B^{k+1} be the optimal basis. We collect a subset of GMICs corresponding to this optimal tableaux, call these cuts $C(B^{k+1})$. See details of which cuts are collected from a tableaux below. For cuts selected from the previously visited basis, we discard those with zero dual value, i.e. we update $C(B)^l$ for $l \in \{1, \dots, k\}$ by removing cut j if $u^j_{B^l} = 0$ and update $q(B^l)$ accordingly. Then we go to step 1, with $k \leftarrow k + 1$.

The method is initialized by the GMICs from the optimal tableaux of the LP relaxation. We terminate if the optimal objective function value of (9) for two consecutive iterations does not change. Otherwise, we repeat the above iteration for at most K iterations. In our experiments, we set $K = 10$.

Finally, after K rounds, we solve (9) and select the cuts with positive dual multipliers. These GMICs are our approximation of the GMIC closure.

Rule for selection of GMICs from a tableaux. We sort the right-hand-sides of the tableaux with respect to the amount of fractionality (defined for $u \in \mathbb{R}$ as $\min\{u - \lfloor u \rfloor, \lceil u \rceil - u\}$). Let $\hat{q}$ be the the number of rows with basic integer variables and the fractionality being greater than 0.001. We let $q(B) = \min\{\hat{q}, 500\}$. We then select the GMICs corresponding to the top $q(B)$ rows in terms of fractionality.

Remark 3. Our method is considerably simpler to implement than the one proposed in Fischetti-Salvagnin [29]. In particular, the paper [29] presents various variants where potentially more basic feasible solutions are visited by solving the Lagrangian dual of (9) and generating cuts from all/subset of the optimal tablueax corresponding to solving (10) with u's not being the optimal dual multiplier - but being the intermediary values obtained while solving the Lagrangian Dual of (9). However, as the paper states, this method requires significant engineering of the gradient updates to solve the Lagrangian dual. Moreover, as we see in the next section, we found that the cuts discovered by our simple implementation already improved the performance of a sizable number of instances.

C Benchmark Set Generation

Right-hand-side changes. We perturbed the right-hand-sides in two steps. In the first step we used simple rules to produce a preliminary perturbation of the right-hand-sides. In the second step, we checked if the first step produced any changes or not, or whether the perturbation led to infeasibilities.

Step 1: Preliminary right-hand-side perturbation. We apply the following simple rules for perturbing the right-hand-side values:

- **Rule 1:** If a constraint contains only two variables and exactly one of them is discrete, then perturb the right-hand-side value b_i to $b_i \cdot r$ where r is uniformly randomly selected from $[0.9, 1.1]$.
- **Rule 2:** If a constraint contains more than two variables and at least one of them is continuous, then perturb the right-hand-side value b_i to $b_i \cdot r$ where r is uniformly randomly selected from $[0.9, 1.1]$. We distinguish this rule from the previous, since we wanted to track how often the changes are made to constraints with two variables.
- **Rule 3:** If a constraint contains only discrete variables, is not an equality constraint and the right-hand-side value is not equal to 1, then perturb the right-hand-side value b_i to $b_i \cdot r$ where r is uniformly randomly selected from $[0.9, 1.1]$.
- **Rule 4:** If a constraint contains only discrete variables and is an equality constraint, then we apply additive perturbation of $\{-1, 0, 1\}$ to the right-hand-side.

The above rules were designed based on the following guiding principles:

1. Prevent integer infeasibility to occur locally at the constraint level as far as possible: **Rule 1** and **Rule 2** perturb the right-hand-side multiplicatively since a continuous variable is present. **Rule 3** perturbs the right-hand-side multiplicatively since the constraint is in inequality form. These perturbations may overlook global implicit constraints on the RHS values. For example, the sum of demands should be lesser than sum of supplies when the constraints represent a network flow.
2. Preserve typical combinatorial constraints: special structures such as set partitioning, set covering, and set packing constraints are not changed; see **Rule 3** and **Rule 4**.

Step 2: Checking the preliminary right-hand-side changes. The perturbations in the previous step may fail due to two reasons:

1. Firstly, none of the constraints satisfy the requirements of **Rules 1-Rule 4**. In this case, we cannot make any right-hand-side changes to this MIPLIB instance.
2. Otherwise, we generate 5 random perturbations. We, then check if these perturbations are integer feasible using Gurobi. If any perturbation is infeasible, then we do not make any right-hand-side changes to this MIPLIB instance.

Objective changes. We perturb each objective coefficient c_j to $c_j \cdot r$ where r is uniformly randomly selected from $[0.75, 1.25]$. Note that, in some MIPLIB instances, the support of the objective function is on only one variable. Clearly, in this case, the above perturbation does not produce any change. Therefore, for such MIPLIB instances, we do not make any changes to the objective function.

D Supplementary Tables

Table 3: List of eliminated instances and reason for elimination.

instance	reason
blp-ar98	no changes
bnatt400	no changes
bnatt500	infeasible
brazil3	no changes
cryptanalysiskb128n5obj14	infeasible
cryptanalysiskb128n5obj16	no changes
csched008	no changes
dano3_3	no changes
dano3_5	no changes
fhnw-binpack4-4	infeasible
fhnw-binpack4-48	no changes
fiball	no changes
germanrr	no changes
gfd-schedulen180f7d50m30k18	no changes
highschool1-aigio	no changes
hypothyroid-k1	no changes
leo1	no changes
mcsched	no changes
neos-2075418-temuka	infeasible
neos-3004026-krka	no changes
neos-3402454-bohle	infeasible
neos-3988577-wolgan	infeasible
neos-5052403-cygnet	too large
neos-5104907-jarama	too large
neos-5114902-kasavu	too large
neos-848589	too large
neos859080	infeasible
ns1116954	no changes
ns1952667	no changes
physiciansched3-3	no changes
physiciansched6-2	no changes
piperout-08	no changes
piperout-27	no changes
proteindesign121hz512p9	no changes
proteindesign122trx11p8	no changes
radiationm18-12-05	no changes
radiationm40-10-02	no changes
rococoB10-011000	no changes
rococoC10-001000	no changes
roi5alpha10n8	too large

Continued on next page

Table 3: List of eliminated instances and reason for elimination.

instance	reason
roll3000	no changes
s100	too large
s250r10	too large
savsched1	too large
square41	too large
square47	too large
supportcase12	too large
supportcase19	too large
supportcase22	too hard
swath1	no changes
swath3	no changes
traininstance2	no changes
triptim1	no changes
wachplan	no changes

Table 4: Instance size statistics and average number of changes of each type applied to generate the 55 instance variations.

Instance	Variables	Constraints	Obj. Changes	RHS Changes					Changes
				Type 1	Type 2	Type 3	Type 4	Total	
30n20b8	18,380	576	2	0	0	0	0	0	2
50v-10	2,013	233	1,647	0	0	0	0	0	1,647
CMS750_4	11,697	16,381	751	0	6,446	1,688	0	8,134	8,885
academictimetablesmall	28,926	23,294	255	0	0	0	0	0	255
air05	7,195	426	7,195	0	0	0	0	0	7,195
app1-1	2,480	4,926	23	0	0	0	0	0	23
app1-2	26,871	53,467	266	0	0	0	0	0	266
assign1-5-8	156	161	26	0	0	0	0	0	26
atlanta-ip	48,738	21,732	31,296	0	0	0	0	0	31,296
b1c1s1	3,872	3,904	1,568	0	272	0	0	272	1,840
bab2	147,912	17,245	116,840	0	0	0	0	0	116,840
bab6	114,240	29,904	73,940	0	0	0	0	0	73,940
beasleyC3	2,500	1,750	1,250	0	0	0	0	0	1,250
binkar10_1	2,298	1,026	1,742	0	0	0	0	0	1,742
blp-ic98	13,640	717	1	0	0	627	0	627	628
bppc4-08	1,456	111	2	0	0	0	0	0	2
buildingenergy	154,978	277,594	68,740	0	8,760	0	0	8,760	77,500
cbs-cta	24,793	10,112	968	0	0	0	0	0	968
chromaticindex1024-7	73,728	67,583	4	0	0	0	0	0	4
chromaticindex512-7	36,864	33,791	4	0	0	0	0	0	4
cmflsp50-24-8-8	16,392	3,520	15,068	0	0	0	0	0	15,068
co-100	48,417	2,187	47,217	0	0	0	0	0	47,217

Continued on next page

Table 4: Instance size statistics and average number of changes of each type applied to generate 55 perturbations.

Instance	Variables	Constraints	Obj. changes	RHS changes					Changes
				Type 1	Type 2	Type 3	Type 4	Total	
cod105	1,024	1,024	1,024	0	0	1,024	0	1,024	2,048
comp07-2idx	17,264	21,235	14,056	0	0	420	131	551	14,607
comp21-2idx	10,863	14,038	8,546	0	0	0	0	0	8,546
cost266-UUE	4,161	1,446	4,161	0	0	0	0	0	4,161
csched007	1,758	351	50	0	0	0	0	0	50
cvs16r128-89	3,472	4,633	2,048	0	0	105	0	105	2,153
decomp2	14,387	10,765	3,331	0	0	0	0	0	3,331
drayage-100-23	11,090	4,630	7,705	0	4,160	0	0	4,160	11,865
drayage-25-23	11,090	4,630	7,705	0	4,160	0	0	4,160	11,865
dws008-01	11,096	6,064	196	0	0	0	0	0	196
eil33-2	4,516	32	4,516	0	0	0	0	0	4,516
eilA101-2	65,832	100	65,832	0	0	0	0	0	65,832
enlight_hard	200	100	100	0	0	0	0	0	100
ex10	17,680	69,608	17,680	0	0	0	0	0	17,680
ex9	10,404	40,962	10,404	0	0	0	0	0	10,404
exp-1-500-5-5	990	550	990	0	250	50	0	300	1,290
fast0507	63,009	507	63,009	0	0	496	0	496	63,505
fastxgemm-n2r6s0t2	784	5,998	208	0	0	0	0	0	208
gen-ip002	41	24	41	0	0	24	0	24	65
gen-ip054	30	27	30	0	0	26	0	26	56
glass-sc	214	6,119	214	0	0	6,119	0	6,119	6,333
glass4	322	396	19	0	351	0	0	351	370

Continued on next page

Table 4: Instance size statistics and average number of changes of each type applied to generate 55 perturbations.

Instance	Variables	Constraints	Obj. changes	RHS changes					Changes
				Type 1	Type 2	Type 3	Type 4	Total	
gmu-35-40	1,205	424	687	0	0	379	0	379	1,066
gmu-35-50	1,919	435	1,022	0	0	406	0	406	1,428
graph20-20-1rand	2,183	5,587	37	0	0	1,369	0	1,369	1,406
graphdraw-domain	254	865	54	0	0	0	0	0	54
h80x6320d	12,640	6,558	12,640	0	0	0	0	0	12,640
ic97_potential	728	1,046	280	0	0	0	0	0	280
icir97_tension	2,494	1,203	408	0	0	0	0	0	408
irish-electricity	61,728	104,259	15,552	0	0	0	0	0	15,552
irp	20,315	39	20,315	0	0	0	0	0	20,315
istanbul-no-cutoff	5,282	20,346	77	0	10,067	1	0	10,068	10,145
k1mushroom	8,211	16,419	1	0	0	16,418	0	16,418	16,419
lectsched-5-obj	21,805	38,884	2,679	0	0	0	0	0	2,679
leo2	11,100	593	1	0	0	583	0	583	584
lotsize	2,985	1,920	2,390	0	0	0	0	0	2,390
mad	220	51	20	0	0	10	0	10	30
map10	164,547	328,818	1,676	0	0	1	0	1	1,677
map16715-04	164,547	328,818	1,676	0	0	1	0	1	1,677
markshare2	74	7	14	0	7	0	0	7	21
markshare_4_0	34	4	4	0	4	0	0	4	8
mas74	151	13	151	0	12	1	0	13	164
mas76	151	12	151	0	11	1	0	12	163
mc11	3,040	1,920	1,520	0	0	0	0	0	1,520

Continued on next page

Table 4: Instance size statistics and average number of changes of each type applied to generate 55 perturbations.

Instance	Variables	Constraints	Obj. changes	RHS changes					Changes
				Type 1	Type 2	Type 3	Type 4	Total	
mik-250-20-75-4	270	195	270	0	75	0	0	75	345
milo-v12-6-r2-40-1	2,688	5,628	210	0	0	0	0	0	210
momentum1	5,174	42,680	2,807	0	0	0	0	0	2,807
mushroom-best	8,468	8,580	8,350	0	0	8,124	0	8,124	16,474
mzzv11	10,240	9,499	205	0	0	5,760	0	5,760	5,965
mzzv42z	11,717	10,460	194	0	0	5,680	0	5,680	5,874
n2seq36q	22,480	2,565	8,100	0	0	0	0	0	8,100
n3div36	22,120	4,484	22,120	0	0	4,484	0	4,484	26,604
n5-3	2,550	1,062	2,500	0	0	0	0	0	2,500
neos-1122047	5,100	57,791	100	0	0	0	0	0	100
neos-1171448	4,914	13,206	2,457	0	5,142	630	0	5,772	8,229
neos-1171737	2,340	4,179	1,170	0	2,469	390	0	2,859	4,029
neos-1354092	13,702	3,135	59	0	0	0	0	0	59
neos-1445765	20,617	2,147	1,000	0	0	1	0	1	1,001
neos-1456979	4,605	6,770	4,245	0	0	0	0	0	4,245
neos-1582420	10,100	10,180	9,900	0	0	0	0	0	9,900
neos-2657525-crna	524	342	32	0	0	60	0	60	92
neos-2746589-doon	50,936	31,530	50,400	0	0	0	0	0	50,400
neos-2978193-inde	20,800	396	20,736	0	0	0	0	0	20,736
neos-2987310-joes	27,837	29,015	24,786	0	567	0	0	567	25,353
neos-3024952-loue	3,255	3,705	3,075	0	0	445	0	445	3,520
neos-3046615-murg	274	498	2	16	1	240	0	257	259

Continued on next page

Table 4: Instance size statistics and average number of changes of each type applied to generate 55 perturbations.

Instance	Variables	Constraints	Obj. changes	RHS changes					Changes
				Type 1	Type 2	Type 3	Type 4	Total	
neos-3083819-nubu	8,644	4,725	3,184	0	0	0	0	0	3,184
neos-3216931-puriri	3,555	5,989	1,686	0	0	0	0	0	1,686
neos-3381206-awhea	2,375	479	475	0	0	0	4	4	479
neos-3402294-bobin	2,904	591,076	24	0	0	0	0	0	24
neos-3555904-turama	37,461	146,493	12,487	0	0	0	0	0	12,487
neos-3627168-kasai	1,462	1,655	3	0	462	70	0	532	535
neos-3656078-kumeu	14,870	17,656	2	0	0	0	0	0	2
neos-3754480-nidda	253	402	253	0	0	0	0	0	253
neos-4300652-rahue	33,003	76,992	10,374	0	0	0	0	0	10,374
neos-4338804-snowy	1,344	1,701	21	0	0	0	0	0	21
neos-4387871-tavua	4,004	4,554	2,000	0	0	0	0	0	2,000
neos-4413714-turia	190,402	2,303	1	0	1	1	0	2	3
neos-4532248-waihi	86,842	167,322	1	0	83,230	0	0	83,230	83,231
neos-4647030-tutaki	12,600	8,382	1,400	0	0	0	0	0	1,400
neos-4722843-widden	77,723	113,555	4,248	0	0	0	0	0	4,248
neos-4738912-atrato	6,216	1,947	1,056	0	0	0	0	0	1,056
neos-4763324-toguru	53,593	106,954	53,131	0	0	0	1	1	53,132
neos-4954672-berkel	1,533	1,848	252	0	450	0	0	450	702
neos-5049753-cuanza	242,736	322,248	228,096	0	0	0	0	0	228,096
neos-5093327-huahum	40,640	51,840	7,808	0	0	0	0	0	7,808
neos-5107597-kakapo	3,114	6,498	24	0	6,396	0	0	6,396	6,420
neos-5188808-nattai	14,544	29,452	2,376	0	0	0	0	0	2,376

Continued on next page

Table 4: Instance size statistics and average number of changes of each type applied to generate 55 perturbations.

Instance	Variables	Constraints	Obj. changes	RHS changes					Changes
				Type 1	Type 2	Type 3	Type 4	Total	
neos-5195221-niemur	14,546	42,256	2	0	0	0	0	0	2
neos-631710	167,056	169,576	556	0	0	1,665	0	1,665	2,221
neos-662469	18,235	1,085	18,235	0	0	0	0	0	18,235
neos-787933	236,376	1,897	1,764	0	0	133	0	133	1,897
neos-827175	32,504	14,187	11,102	0	0	0	0	0	11,102
neos-860300	1,385	850	1,385	0	0	510	0	510	1,895
neos-873061	175,288	93,360	87,644	0	2	0	0	2	87,646
neos-911970	888	107	48	0	0	0	0	0	48
neos-933966	31,762	12,047	31,762	0	3,243	4,988	0	8,231	39,993
neos-950242	5,760	34,224	24	0	0	0	0	0	24
neos-957323	57,756	3,757	57,576	0	0	1,523	0	1,523	59,099
neos-960392	59,376	4,744	1,575	0	0	1,645	0	1,645	3,220
neos17	535	486	485	0	486	0	0	486	971
neos5	63	63	63	0	60	3	0	63	126
neos8	23,228	46,324	32	0	0	23,020	0	23,020	23,052
net12	14,115	14,021	39	0	0	0	0	0	39
netdiversion	129,180	119,589	39,798	0	0	0	0	0	39,798
nexp-150-20-8-5	20,115	4,620	17,880	0	0	0	0	0	17,880
ns1208400	2,883	4,289	3	0	0	0	0	0	3
ns1644855	40,200	40,698	10,000	0	10,000	0	0	10,000	20,000
ns1760995	17,956	615,388	2	0	0	0	0	0	2
ns1830653	1,629	2,932	747	0	0	0	0	0	747

Continued on next page

Table 4: Instance size statistics and average number of changes of each type applied to generate 55 perturbations.

Instance	Variables	Constraints	Obj. changes	RHS changes					Changes
				Type 1	Type 2	Type 3	Type 4	Total	
nu25-pr12	5,868	2,313	5,868	0	0	23	0	23	5,891
nursesched-medium-hint03	34,248	14,062	26,854	0	0	0	0	0	26,854
nursesched-sprint02	10,250	3,522	6,925	0	0	0	0	0	6,925
nw04	87,482	36	87,482	0	0	0	0	0	87,482
opm2-z10-s4	6,250	160,633	6,212	0	0	8	0	8	6,220
p200x1188c	2,376	1,388	1,188	0	0	0	0	0	1,188
peg-solitaire-a3	4,552	4,587	37	0	0	0	0	0	37
pg	2,700	125	2,600	0	125	0	0	125	2,725
pg5_34	2,600	225	2,600	0	25	0	0	25	2,625
pk1	86	45	1	0	15	0	0	15	16
qap10	4,150	1,820	2,610	0	0	0	0	0	2,610
rail01	117,527	46,843	15,760	0	0	0	0	0	15,760
rail02	270,869	95,791	34,359	0	0	0	0	0	34,359
rail507	63,019	509	63,019	0	0	498	0	498	63,517
ran14x18-disj-8	504	447	504	0	0	0	0	0	504
rd-rplusc-21	622	125,899	2	0	0	0	0	0	2
reblock115	1,150	4,735	1,150	0	0	20	0	20	1,170
rmatr100-p10	7,359	7,260	7,359	0	7,259	0	1	7,260	14,619
rmatr200-p5	37,816	37,617	37,816	0	37,616	0	1	37,617	75,433
rocI-4-11	6,839	10,883	4	0	0	0	0	0	4
rocII-5-11	11,523	26,897	177	0	0	0	0	0	177
roi2alpha3n4	6,816	1,251	174	0	0	0	2	2	176

Continued on next page

Table 4: Instance size statistics and average number of changes of each type applied to generate 55 perturbations.

Instance	Variables	Constraints	Obj. changes	RHS changes					Changes
				Type 1	Type 2	Type 3	Type 4	Total	
satellites2-40	35,378	20,916	741	0	0	0	0	0	741
satellites2-60-fs	35,378	16,516	741	0	0	0	0	0	741
sct2	5,885	2,151	5	0	0	0	0	0	5
seymour	1,372	4,944	1,372	0	0	4,542	0	4,542	5,914
seymour1	1,372	4,944	1,372	18	4,134	408	0	4,560	5,932
sing326	55,156	50,781	23,983	0	168	0	0	168	24,151
sing44	59,708	54,745	26,050	0	168	0	0	168	26,218
snp-02-004-104	228,350	126,512	215,696	0	0	0	0	0	215,696
sorrell3	1,024	169,162	1,024	0	0	0	0	0	1,024
sp150x300d	600	450	300	0	0	0	0	0	300
sp97ar	14,101	1,761	14,101	0	0	0	0	0	14,101
sp98ar	15,085	1,435	15,085	0	0	0	0	0	15,085
splice1k1	3,253	6,505	1	0	0	6,504	0	6,504	6,505
supportcase10	14,770	165,684	70	0	0	0	0	0	70
supportcase18	13,410	240	120	0	0	0	0	0	120
supportcase26	436	870	40	0	792	0	0	792	832
supportcase33	20,203	20,489	45	0	0	0	0	0	45
supportcase40	16,440	38,192	15,480	0	0	0	0	0	15,480
supportcase42	19,466	18,439	18,439	1,028	0	0	0	1,028	19,467
supportcase6	130,052	771	130,051	0	0	0	0	0	130,051
supportcase7	138,844	6,532	147	0	3,147	23	0	3,170	3,317
tbfp-network	72,747	2,436	70,343	0	0	0	0	0	70,343

Continued on next page

Table 4: Instance size statistics and average number of changes of each type applied to generate 55 perturbations.

Instance	Variables	Constraints	Obj. changes	RHS changes					Changes
				Type 1	Type 2	Type 3	Type 4	Total	
thor50dday	106,261	53,360	53,130	0	49	0	0	49	53,179
timtab1	397	171	128	0	0	0	0	0	128
tr12-30	1,080	750	720	0	370	0	0	370	1,090
traininstance6	10,218	12,309	1	0	0	52	0	52	53
trento1	7,687	1,265	7,687	0	1,259	1	0	1,260	8,947
uccase12	62,529	121,161	22,332	0	0	0	0	0	22,332
uccase9	33,242	49,565	13,975	0	0	0	0	0	13,975
uct-subprob	2,256	1,973	1,345	0	0	0	0	0	1,345
unitcal_7	25,755	48,939	9,408	0	0	0	0	0	9,408
var-smallemery-m6j6	5,608	13,416	2	0	0	3	0	3	5

Table 5: Number of instances having t distinct optimal solutions (integer part) across five perturbations.

Solutions (t)	Num. of instances
1	8
2	6
3	7
4	8
5	157

Table 6: Number of obj. and RHS changes applied to each instance, and number of distinct optimal solutions and values obtained across five perturbations.

Instance	Obj. changes	RHS changes	Solutions	Opt. values
30n20b8	2	0	5	5
50v-10	1,647	0	5	5
CMS750_4	751	8,134	5	5
academictimetablesmall	255	0	5	1
air05	7,195	0	5	5
app1-1	23	0	5	5
app1-2	266	0	1	5
assign1-5-8	26	0	5	5
atlanta-ip	31,296	0	5	5
b1c1s1	1,568	272	5	5
bab2	116,840	0	5	5
bab6	73,940	0	5	5
beasleyC3	1,250	0	5	5
binkar10_1	1,742	0	5	5
blp-ic98	1	627	5	5
bppc4-08	2	0	5	5
buildingenergy	68,740	8,760	5	5
cbs-cta	968	0	5	1
chromaticindex1024-7	4	0	1	5
chromaticindex512-7	4	0	1	5
cmflsp50-24-8-8	15,068	0	5	5
co-100	47,217	0	5	5
cod105	1,024	1,024	1	1
comp07-2idx	14,056	551	5	5
comp21-2idx	8,546	0	5	5
cost266-UUE	4,161	0	5	5
csched007	50	0	5	5
cvs16r128-89	2,048	105	5	5
decomp2	3,331	0	1	5
drayage-100-23	7,705	4,160	5	5
drayage-25-23	7,705	4,160	5	5
dws008-01	196	0	5	5
eil33-2	4,516	0	4	5
eilA101-2	65,832	0	4	5
enlight_hard	100	0	1	5
ex10	17,680	0	5	5
ex9	10,404	0	5	5
exp-1-500-5-5	990	300	5	5
fast0507	63,009	496	5	5

Continued on next page

Table 6: Number of obj. and RHS changes applied to each instance, and number of distinct optimal solutions and values obtained across five perturbations.

Instance	Obj. changes	RHS changes	Solutions	Opt. values
fastxgemm-n2r6s0t2	208	0	5	5
gen-ip002	41	24	5	5
gen-ip054	30	26	5	5
glass-sc	214	6,119	5	5
glass4	19	351	5	5
gmu-35-40	687	379	5	5
gmu-35-50	1,022	406	5	5
graph20-20-1rand	37	1,369	5	5
graphdraw-domain	54	0	5	5
h80x6320d	12,640	0	5	5
ic97_potential	280	0	5	5
icir97_tension	408	0	5	5
irish-electricity	15,552	0	5	5
irp	20,315	0	5	5
istanbul-no-cutoff	77	10,068	1	5
k1mushroom	1	16,418	5	5
lectsched-5-obj	2,679	0	5	5
leo2	1	583	5	5
lotsize	2,390	0	5	5
mad	20	10	5	5
map10	1,676	1	5	5
map16715-04	1,676	1	4	5
markshare2	14	7	5	5
markshare_4_0	4	4	5	5
mas74	151	13	5	5
mas76	151	12	5	5
mc11	1,520	0	5	5
mik-250-20-75-4	270	75	5	5
milo-v12-6-r2-40-1	210	0	4	5
momentum1	2,807	0	5	5
mushroom-best	8,350	8,124	3	5
mzzv11	205	5,760	4	4
mzzv42z	194	5,680	5	5
n2seq36q	8,100	0	5	5
n3div36	22,120	4,484	5	5
n5-3	2,500	0	5	5
neos-1122047	100	0	5	5
neos-1171448	2,457	5,772	5	5
neos-1171737	1,170	2,859	5	5

Continued on next page

Table 6: Number of obj. and RHS changes applied to each instance, and number of distinct optimal solutions and values obtained across five perturbations.

Instance	Obj. changes	RHS changes	Solutions	Opt. values
neos-1354092	59	0	5	5
neos-1445765	1,000	1	5	5
neos-1456979	4,245	0	5	5
neos-1582420	9,900	0	5	5
neos-2657525-crna	32	60	5	4
neos-2746589-doon	50,400	0	5	5
neos-2978193-inde	20,736	0	5	5
neos-2987310-joes	24,786	567	2	5
neos-3024952-loue	3,075	445	5	5
neos-3046615-murg	2	257	5	5
neos-3083819-nubu	3,184	0	5	5
neos-3216931-puriri	1,686	0	4	5
neos-3381206-awhea	475	4	5	5
neos-3402294-bobin	24	0	5	5
neos-3555904-turama	12,487	0	5	5
neos-3627168-kasai	3	532	5	5
neos-3656078-kumeu	2	0	5	5
neos-3754480-nidda	253	0	5	5
neos-4300652-rahue	10,374	0	5	5
neos-4338804-snowy	21	0	5	5
neos-4387871-tavua	2,000	0	5	5
neos-4413714-turia	1	2	5	5
neos-4532248-waihi	1	83,230	5	5
neos-4647030-tutaki	1,400	0	5	5
neos-4722843-widden	4,248	0	5	5
neos-4738912-atrato	1,056	0	5	5
neos-4763324-toguru	53,131	1	5	5
neos-4954672-berkel	252	450	5	5
neos-5049753-cuanza	228,096	0	5	5
neos-5093327-huahum	7,808	0	2	5
neos-5107597-kakapo	24	6,396	5	5
neos-5188808-nattai	2,376	0	5	5
neos-5195221-niemur	2	0	5	5
neos-631710	556	1,665	5	5
neos-662469	18,235	0	5	5
neos-787933	1,764	133	5	5
neos-827175	11,102	0	5	5
neos-860300	1,385	510	3	5
neos-873061	87,644	2	5	5

Continued on next page

Table 6: Number of obj. and RHS changes applied to each instance, and number of distinct optimal solutions and values obtained across five perturbations.

Instance	Obj. changes	RHS changes	Solutions	Opt. values
neos-911970	48	0	5	5
neos-933966	31,762	8,231	5	5
neos-950242	24	0	5	5
neos-957323	57,576	1,523	5	5
neos-960392	1,575	1,645	2	2
neos17	485	486	3	5
neos5	63	63	5	5
neos8	32	23,020	1	1
net12	39	0	5	5
netdiversion	39,798	0	5	5
nexp-150-20-8-5	17,880	0	5	5
ns1208400	3	0	5	5
ns1644855	10,000	10,000	5	5
ns1760995	2	0	5	5
ns1830653	747	0	2	5
nu25-pr12	5,868	23	5	5
nursesched-medium-hint03	26,854	0	5	5
nursesched-sprint02	6,925	0	5	5
nw04	87,482	0	5	5
opm2-z10-s4	6,212	8	5	5
p200x1188c	1,188	0	5	5
peg-solitaire-a3	37	0	5	5
pg	2,600	125	5	5
pg5_34	2,600	25	5	5
pk1	1	15	5	5
qap10	2,610	0	3	5
rail01	15,760	0	5	5
rail02	34,359	0	5	5
rail507	63,019	498	5	5
ran14x18-disj-8	504	0	5	5
rd-rplusc-21	2	0	4	5
reblock115	1,150	20	5	5
rmatr100-p10	7,359	7,260	5	5
rmatr200-p5	37,816	37,617	5	5
rocI-4-11	4	0	5	5
rocII-5-11	177	0	2	5
roi2alpha3n4	174	2	5	5
satellites2-40	741	0	5	5
satellites2-60-fs	741	0	5	5

Continued on next page

Table 6: Number of obj. and RHS changes applied to each instance, and number of distinct optimal solutions and values obtained across five perturbations.

Instance	Obj. changes	RHS changes	Solutions	Opt. values
sct2	5	0	5	5
seymour	1,372	4,542	5	5
seymour1	1,372	4,560	5	5
sing326	23,983	168	5	5
sing44	26,050	168	5	5
snp-02-004-104	215,696	0	5	5
sorrell3	1,024	0	5	5
sp150x300d	300	0	5	5
sp97ar	14,101	0	5	5
sp98ar	15,085	0	5	5
splice1k1	1	6,504	5	5
supportcase10	70	0	2	5
supportcase18	120	0	5	5
supportcase26	40	792	3	5
supportcase33	45	0	5	5
supportcase40	15,480	0	5	5
supportcase42	18,439	1,028	5	5
supportcase6	130,051	0	5	5
supportcase7	147	3,170	3	5
tbfp-network	70,343	0	5	5
thor50dday	53,130	49	5	5
timtab1	128	0	4	5
tr12-30	720	370	5	5
traininstance6	1	52	5	5
trento1	7,687	1,260	5	5
uccase12	22,332	0	5	5
uccase9	13,975	0	5	5
uct-subprob	1,345	0	5	5
unitcal_7	9,408	0	5	5
var-smallemery-m6j6	2	3	3	5

Table 7: Average running time and nodes required to solve original and perturbed MIPLIB 2017 instances.

Instance	MIP Time (s)		B&B Nodes	
	Original	Perturbed	Original	Perturbed
30n20b8	1.1	7.8	271.0	168.2
50v-10	0.0	8,802.6	0.0	6,998,394.2
CMS750_4	1.9	30.6	368.0	4,114.8
academictimetablesmall	314.7	665.5	1,956.0	5,159.8
air05	3.0	6.2	569.0	1,095.8
app1-1	0.3	0.4	1.0	90.8
app1-2	1.7	59.5	1.0	1,219.6
assign1-5-8	0.0	5,026.3	0.0	11,980,898.8
atlanta-ip	461.4	309.9	3,283.0	2,374.6
b1c1s1	1,757.0	710.4	86,111.0	29,430.6
bab2	3,309.2	1,359.9	10,544.0	14,578.2
bab6	1,774.8	491.7	22,060.0	4,746.6
beasleyC3	0.4	1.1	1.0	433.4
binkar10_1	4.0	0.3	5,343.0	195.6
blp-ic98	31.9	9.3	5,296.0	2,837.0
bppc4-08	0.0	8,022.4	0.0	313,054.4
buildingenergy	73.7	202.6	107.0	1,373.2
cbs-cta	0.3	0.3	1.0	1.0
chromaticindex1024-7	0.0	339.8	0.0	1,924.8
chromaticindex512-7	0.0	183.3	0.0	3,510.2
cmflsp50-24-8-8	11.1	9,239.6	286.0	304,385.0
co-100	92.9	71.6	4,790.0	1,292.2
cod105	33.8	0.0	7.0	0.0
comp07-2idx	21.8	93.6	325.0	1,305.8
comp21-2idx	2,938.0	4,883.5	29,555.0	56,161.2
cost266-UUE	905.6	771.1	68,142.0	73,696.6
csched007	182.7	221.8	28,162.0	40,873.6
cvs16r128-89	14,400.0	0.0	299,303.0	14.8
decomp2	0.1	0.1	0.0	0.0
drayage-100-23	0.1	0.3	0.0	1.8
drayage-25-23	0.1	0.4	0.0	4.8
dws008-01	4.0	3,192.4	538.0	533,524.2
eil33-2	0.0	5.1	0.0	4,676.0
eilA101-2	0.0	10,176.9	0.0	32,810.8
enlight_hard	0.0	0.0	0.0	0.0
ex10	12.4	14.5	1.0	1.0
ex9	4.2	4.0	0.0	0.0
exp-1-500-5-5	0.0	3.1	0.0	3,666.8

Continued on next page

Table 7: Average running time and nodes required to solve original and perturbed MIPLIB 2017 instances.

Instance	MIP Time (s)		B&B Nodes	
	Original	Perturbed	Original	Perturbed
fast0507	28.6	10,830.0	1,151.0	973,617.6
fastxgemm-n2r6s0t2	113.8	179.1	102,468.0	58,755.8
gen-ip002	325.1	31.6	4,348,424.0	494,690.4
gen-ip054	604.6	0.5	17,642,730.0	10,239.4
glass-sc	1,271.9	479.1	296,173.0	114,010.0
glass4	8.2	0.0	11,181.0	0.4
gmu-35-40	0.0	0.0	0.0	1.0
gmu-35-50	0.0	0.0	0.0	2.8
graph20-20-1rand	13,055.7	0.0	1,418,345.0	32.6
graphdraw-domain	90.3	112.1	76,531.0	156,299.0
h80x6320d	9.7	5.7	2,263.0	887.8
ic97_potential	2,111.5	91.7	837,733.0	36,729.2
icir97_tension	122.7	124.0	58,276.0	61,710.8
irish-electricity	187.8	339.5	10,750.0	22,667.4
irp	0.7	1.4	57.0	266.6
istanbul-no-cutoff	21.2	0.0	124.0	0.0
k1mushroom	44.7	106.8	1.0	2,438.0
lectsched-5-obj	25.9	77.9	1,060.0	9,581.8
leo2	443.9	241.0	78,538.0	98,782.2
lotsize	2,339.3	1,122.9	102,755.0	45,441.2
mad	900.8	351.8	6,482,472.0	2,300,337.8
map10	94.5	107.7	1,240.0	1,332.6
map16715-04	0.0	110.2	0.0	1,685.8
markshare2	5,825.8	14,400.0	89,421,453.0	146,860,300.6
markshare_4_0	0.2	26.9	1,725.0	419,233.6
mas74	163.5	152.2	3,024,502.0	2,188,836.0
mas76	14.6	10.5	313,562.0	194,354.4
mc11	3.8	3.0	802.0	528.6
mik-250-20-75-4	0.0	0.1	0.0	4.0
milo-v12-6-r2-40-1	0.1	469.4	0.0	176,944.6
momentum1	739.4	266.7	65,750.0	19,948.4
mushroom-best	110.5	73.2	10,646.0	5,386.2
mzzv11	3.0	0.0	1.0	0.4
mzzv42z	3.1	0.0	1.0	1.0
n2seq36q	1.2	8.0	1.0	909.0
n3div36	372.4	55.1	85,420.0	20,232.8
n5-3	0.0	4.2	0.0	999.6
neos-1122047	3.2	8.0	1.0	1.0

Continued on next page

Table 7: Average running time and nodes required to solve original and perturbed MIPLIB 2017 instances.

Instance	MIP Time (s)		B&B Nodes	
	Original	Perturbed	Original	Perturbed
neos-1171448	0.9	11.6	1.0	2,067.8
neos-1171737	1.6	0.5	1.0	480.2
neos-1354092	2,323.1	1,725.8	27,884.0	10,299.0
neos-1445765	6.1	6.9	312.0	433.4
neos-1456979	15.8	65.9	1,252.0	4,962.2
neos-1582420	6.0	20.4	718.0	5,316.2
neos-2657525-crna	6,329.0	8,703.1	26,105,810.0	34,491,198.0
neos-2746589-doon	56.7	16.4	775.0	5,650.0
neos-2978193-inde	314.6	1.0	153,875.0	49.8
neos-2987310-joes	1.1	0.5	1.0	0.2
neos-3024952-loue	219.2	14,400.0	112,701.0	4,034,190.0
neos-3046615-murg	12.8	4,191.3	22,028.0	4,261,382.6
neos-3083819-nubu	0.3	0.3	183.0	739.8
neos-3216931-puriri	104.7	122.3	1,029.0	1,348.2
neos-3381206-awhea	0.9	2,261.5	806.0	9,984,238.8
neos-3402294-bobin	4.8	5.0	1.0	1.0
neos-3555904-turama	369.7	13,688.0	1,743.0	15,079.4
neos-3627168-kasai	3,552.6	12,673.7	3,065,789.0	22,983,206.8
neos-3656078-kumeu	1,195.3	1,181.2	146,466.0	111,987.2
neos-3754480-nidda	3,073.9	1.2	5,360,514.0	761.2
neos-4300652-rahue	19.4	23.8	9.0	10.8
neos-4338804-snowy	2,148.3	3,893.4	3,337,594.0	6,005,116.6
neos-4387871-tavua	13,783.9	8,325.1	138,205.0	104,636.4
neos-4413714-turia	11.0	18.6	1.0	1.0
neos-4532248-waihi	134.3	256.7	971.0	619.4
neos-4647030-tutaki	69.6	55.7	2,959.0	4,523.4
neos-4722843-widden	8.9	13.1	2,425.0	3,443.8
neos-4738912-atrato	20.5	7.2	4,151.0	2,398.8
neos-4763324-toguru	93.7	237.7	1,763.0	3,364.2
neos-4954672-berkel	6,050.7	12,038.7	1,769,827.0	3,752,385.8
neos-5049753-cuanza	303.1	1,056.1	751.0	981.2
neos-5093327-huahum	6,633.8	3,708.1	359,843.0	200,749.8
neos-5107597-kakapo	26.4	14,400.0	11,955.0	523,772.8
neos-5188808-nattai	227.0	241.6	11,131.0	12,283.8
neos-5195221-niemur	995.3	1,804.8	9,802.0	12,808.4
neos-631710	2,564.0	31.6	10,202.0	139.2
neos-662469	27.1	190.8	588.0	7,153.0
neos-787933	0.2	0.2	1.0	1.0

Continued on next page

Table 7: Average running time and nodes required to solve original and perturbed MIPLIB 2017 instances.

Instance	MIP Time (s)		B&B Nodes	
	Original	Perturbed	Original	Perturbed
neos-827175	0.1	1.6	1.0	1.0
neos-860300	8.9	7.2	1,071.0	1,195.0
neos-873061	259.2	634.0	10,735.0	8,283.0
neos-911970	12.8	2.4	67,433.0	2,170.0
neos-933966	1.8	0.0	1.0	1.0
neos-950242	10.1	17.6	1.0	750.4
neos-957323	1.6	0.1	1.0	1.0
neos-960392	15.9	0.0	3.0	1.0
neos17	0.7	1.9	2,282.0	4,038.2
neos5	128.7	3.0	509,842.0	15,629.2
neos8	0.3	0.1	1.0	0.0
net12	54.2	44.2	859.0	939.2
netdiversion	149.9	221.6	91.0	250.2
nexp-150-20-8-5	0.0	90.8	0.0	1,954.0
ns1208400	4.0	7.4	1.0	292.8
ns1644855	159.1	14,400.0	1.0	5,184.6
ns1760995	870.1	1,190.7	2,684.0	3,264.0
ns1830653	26.1	31.9	4,533.0	5,527.2
nu25-pr12	0.2	0.3	65.0	75.4
nursesched-medium-hint03	2,661.7	2,519.6	15,698.0	16,777.0
nursesched-sprint02	2.0	4.7	1.0	4.6
nw04	0.9	1.1	54.0	32.4
opm2-z10-s4	2,598.8	2,355.9	4,342.0	3,450.2
p200x1188c	0.1	0.1	1.0	1.0
peg-solitaire-a3	65.3	14,295.1	1,566.0	321,896.0
pg	2.3	0.2	1,312.0	46.4
pg5_34	11.8	7.0	5,992.0	3,347.4
pk1	18.4	22.4	202,365.0	98,672.0
qap10	19.5	18.6	7.0	30.6
rail01	271.2	293.8	78.0	90.8
rail02	6,008.9	7,844.6	441.0	242.4
rail507	55.1	9,408.7	2,094.0	714,715.8
ran14x18-disj-8	0.0	555.8	0.0	307,958.6
rd-rplusc-21	5.6	182.1	1.0	30,082.4
reblock115	566.9	3,447.7	347,852.0	3,974,610.4
rmatr100-p10	13.2	391.6	1,818.0	6,112.8
rmatr200-p5	4,800.6	14,400.0	27,209.0	7,835.4
rocI-4-11	1.3	104.7	1.0	15,238.8

Continued on next page

Table 7: Average running time and nodes required to solve original and perturbed MIPLIB 2017 instances.

Instance	MIP Time (s)		B&B Nodes	
	Original	Perturbed	Original	Perturbed
rocII-5-11	1.5	2,234.2	1.0	91,038.6
roi2alpha3n4	207.3	325.4	13,599.0	11,734.4
satellites2-40	0.8	36.6	0.0	46.2
satellites2-60-fs	558.2	64.2	418.0	104.6
sct2	7.9	30.8	3,310.0	12,264.0
seymour	7,676.4	38.3	538,626.0	4,039.6
seymour1	23.1	11.6	2,020.0	802.4
sing326	1,415.5	14,400.0	8,418.0	60,247.6
sing44	839.0	5,491.5	6,676.0	57,967.8
snp-02-004-104	10.2	42.8	1.0	1,373.6
sorrell3	11,533.5	13,637.5	182,638.0	200,777.0
sp150x300d	0.0	0.0	1.0	1.0
sp97ar	541.4	1,372.7	46,239.0	158,529.6
sp98ar	111.3	132.3	11,477.0	19,815.4
splice1k1	14,400.0	14,400.1	12,978.0	344,355.4
supportcase10	1,070.2	5,786.1	550.0	3,110.0
supportcase18	703.0	6,329.3	585,273.0	4,308,341.2
supportcase26	110.9	0.0	94,136.0	1.0
supportcase33	23.2	16.3	1,473.0	1,540.8
supportcase40	487.2	405.4	32,777.0	26,366.8
supportcase42	144.3	150.2	20,075.0	8,982.8
supportcase6	124.9	85.4	1,924.0	1,015.8
supportcase7	9.4	1.3	30.0	1.0
tbfp-network	144.8	1,984.7	285.0	28,995.2
thor50dday	1,444.6	567.3	26,551.0	6,443.2
timtab1	34.8	57.9	29,876.0	44,905.6
tr12-30	0.0	154.3	0.0	194,284.4
traininstance6	3.7	0.2	9,222.0	545.8
trento1	177.9	5,945.9	3,564.0	264,790.2
uccase12	4.6	9.4	1.0	1.0
uccase9	942.3	9,464.9	27,823.0	403,509.0
uct-subprob	1,385.7	1,151.6	130,281.0	114,494.6
unitcal_7	15.6	11.2	1.0	22.6
var-smallemery-m6j6	446.4	524.8	316,569.0	337,186.6
Arithmetic Mean	911.3	1,868.0	908,303.6	1,470,437.2
Shifted Geo. Mean	174.0	294.6	2,699.8	5,805.0

Table 8: Cut collection statistics for the five test variations.

Instance	Time (s)	MIP Gap (%)			Success (%)
		Initial	Final	Delta	
30n20b8	16.27	99.47	50.00	49.46	100.00
50v-10	26.82	15.18	6.93	8.25	100.00
CMS750_4	—	—	—	—	0.00
academictimetablesmall	—	—	—	—	0.00
air05	56.55	2.41	2.17	0.24	100.00
app1-1	34.78	700.19	410.10	290.08	100.00
app1-2	187.40	546.55	543.87	2.68	100.00
assign1-5-8	35.86	12.95	11.85	1.11	100.00
atlanta-ip	713.05	10.79	10.54	0.25	100.00
b1c1s1	42.67	87.21	64.57	22.64	100.00
bab2	4,591.69	18.21	8.57	9.64	100.00
bab6	2,058.39	12.68	8.87	3.80	100.00
beasleyC3	37.22	94.62	35.45	59.17	100.00
binkar10_1	33.10	1.93	1.74	0.19	100.00
blp-ic98	80.99	6.44	4.34	2.10	100.00
bppc4-08	—	—	—	—	0.00
buildingenergy	2,522.52	0.02	0.02	0.00	100.00
cbs-cta	—	—	—	—	0.00
chromaticindex1024-7	—	—	—	—	0.00
chromaticindex512-7	—	—	—	—	0.00
cmflsp50-24-8-8	796.02	2.73	2.58	0.16	100.00
co-100	722.73	65.98	59.15	6.83	100.00
cod105	137.22	—	—	—	100.00
comp07-2idx	56.64	49.17	46.02	3.15	80.00
comp21-2idx	47.73	100.00	93.49	6.51	100.00
cost266-UUE	28.26	19.13	10.25	8.88	100.00
csched007	18.72	23.99	17.48	6.51	100.00
cvs16r128-89	232.00	224.37	222.66	1.71	100.00
decomp2	—	—	—	—	0.00
drayage-100-23	32.32	87.21	86.22	0.99	100.00
drayage-25-23	33.46	87.01	86.05	0.97	100.00
dws008-01	24.21	98.66	95.00	3.65	100.00
eil33-2	26.54	25.32	23.25	2.07	100.00
eilA101-2	253.35	22.12	21.65	0.47	100.00
enlight_hard	20.80	100.00	57.90	42.10	100.00
ex10	13,964.53	15.01	15.00	0.01	20.00
ex9	4,698.90	11.90	11.87	0.02	100.00
exp-1-500-5-5	13.72	65.89	41.35	24.54	100.00

Continued on next page

Table 8: Cut collection statistics for the five test variations.

Instance	Time (s)	MIP Gap (%)			Success (%)
		Initial	Final	Delta	
fast0507	922.88	35.93	32.94	3.00	100.00
fastxgemm-n2r6s0t2	21.13	90.43	90.35	0.08	100.00
gen-ip002	19.79	1.19	1.09	0.10	100.00
gen-ip054	20.09	2.33	1.97	0.36	100.00
glass-sc	26.26	55.06	54.55	0.51	100.00
glass4	—	—	—	—	0.00
gmu-35-40	22.19	162.20	11.10	151.10	100.00
gmu-35-50	21.46	204.65	14.10	190.55	100.00
graph20-20-1rand	—	—	—	—	0.00
graphdraw-domain	20.85	36.78	36.32	0.46	100.00
h80x6320d	28.87	16.51	7.99	8.52	100.00
ic97_potential	21.61	13.33	8.68	4.65	100.00
icir97_tension	—	—	—	—	0.00
irish-electricity	3,679.44	31.69	27.74	3.95	20.00
irp	35.74	1.30	0.29	1.01	100.00
istanbul-no-cutoff	52.89	79.36	75.76	3.60	60.00
k1mushroom	1,087.67	728.48	726.28	2.20	60.00
lectsched-5-obj	—	—	—	—	0.00
leo2	41.25	20.75	19.09	1.66	100.00
lotsize	31.13	76.73	59.54	17.19	100.00
mad	—	—	—	—	0.00
map10	326.73	19.82	18.77	1.05	20.00
map16715-04	219.13	140.12	139.22	0.91	40.00
markshare2	—	—	—	—	0.00
markshare_4_0	—	—	—	—	0.00
mas74	20.04	36.90	36.51	0.39	100.00
mas76	19.77	10.66	10.57	0.09	100.00
mc11	24.68	94.56	41.36	53.20	100.00
mik-250-20-75-4	19.27	18.94	5.56	13.38	100.00
milo-v12-6-r2-40-1	62.87	37.10	19.70	17.41	80.00
momentum1	119.66	32.70	17.41	15.28	80.00
mushroom-best	17.48	99.97	99.94	0.03	80.00
mzzv11	138.86	29,001.11	28,348.70	652.41	100.00
mzzv42z	59.67	1,594.07	1,557.16	36.90	100.00
n2seq36q	103.82	1.11	0.41	0.70	100.00
n3div36	48.17	28.12	25.50	2.62	100.00
n5-3	15.79	63.91	25.57	38.34	100.00
neos-1122047	116.34	0.86	0.79	0.07	100.00
neos-1171448	63.08	4.94	2.61	2.33	100.00

Continued on next page

Table 8: Cut collection statistics for the five test variations.

Instance	Time (s)	MIP Gap (%)			Success (%)
		Initial	Final	Delta	
neos-1171737	32.02	38.19	33.99	4.20	100.00
neos-1354092	262.94	70.94	4.87	66.07	100.00
neos-1445765	43.76	42.39	34.97	7.42	100.00
neos-1456979	22.76	13.96	9.32	4.64	100.00
neos-1582420	44.31	4.41	3.90	0.52	100.00
neos-2657525-crna	—	—	—	—	0.00
neos-2746589-doon	203.80	1.64	1.48	0.15	80.00
neos-2978193-inde	32.68	3.22	1.62	1.60	100.00
neos-2987310-joes	118.21	209,305.81	49,331.07	159,974.74	100.00
neos-3024952-loue	44.52	149.72	122.55	27.17	100.00
neos-3046615-murg	22.51	87.46	81.82	5.64	100.00
neos-3083819-nubu	24.55	0.39	0.22	0.17	100.00
neos-3216931-puriri	270.00	35.21	35.12	0.09	100.00
neos-3381206-awhea	20.91	10.08	10.08	0.00	100.00
neos-3402294-bobin	—	—	—	—	0.00
neos-3555904-turama	166.21	19.46	19.03	0.43	100.00
neos-3627168-kasai	25.44	4.80	3.41	1.39	100.00
neos-3656078-kumeu	35.81	69.87	65.37	4.51	20.00
neos-3754480-nidda	19.90	1,061.84	629.97	431.88	100.00
neos-4300652-rahue	—	—	—	—	0.00
neos-4338804-snowy	—	—	—	—	0.00
neos-4387871-tavua	21.28	69.43	66.46	2.97	100.00
neos-4413714-turia	—	—	—	—	0.00
neos-4532248-waihi	95.99	100.00	2.04	97.96	100.00
neos-4647030-tutaki	171.32	0.84	0.01	0.83	100.00
neos-4722843-widden	300.76	105.38	69.57	35.81	100.00
neos-4738912-atrato	56.56	77.87	46.23	31.64	100.00
neos-4763324-toguru	516.77	32.27	31.78	0.49	100.00
neos-4954672-berkel	26.34	55.35	45.25	10.10	100.00
neos-5049753-cuanza	1,378.43	18.22	1.30	16.93	100.00
neos-5093327-huahum	116.54	36.87	32.48	4.38	100.00
neos-5107597-kakapo	31.54	100.00	98.31	1.69	80.00
neos-5188808-nattai	—	—	—	—	0.00
neos-5195221-niemur	—	—	—	—	0.00
neos-631710	—	—	—	—	0.00
neos-662469	343.41	0.06	0.05	0.01	100.00
neos-787933	370.60	92.58	85.16	7.42	100.00
neos-827175	—	—	—	—	0.00
neos-860300	35.62	51.77	34.09	17.68	100.00

Continued on next page

Table 8: Cut collection statistics for the five test variations.

Instance	Time (s)	MIP Gap (%)			Success (%)
		Initial	Final	Delta	
neos-873061	357.64	95.32	13.44	81.88	100.00
neos-911970	24.84	61.96	61.93	0.03	100.00
neos-933966	1,246.12	98.82	98.82	0.00	100.00
neos-950242	53.38	75.48	71.25	4.24	100.00
neos-957323	747.55	493.47	433.74	59.74	100.00
neos-960392	161.84	8,633.93	20.11	8,613.82	100.00
neos17	21.26	99.55	84.78	14.77	100.00
neos5	19.96	9.72	8.74	0.98	100.00
neos8	131.60	—	—	—	100.00
net12	77.45	91.69	78.12	13.57	100.00
netdiversion	994.98	5.20	4.30	0.90	100.00
nexp-150-20-8-5	37.52	92.22	67.41	24.81	100.00
ns1208400	—	—	—	—	0.00
ns1644855	—	—	—	—	0.00
ns1760995	2,612.19	147.35	145.71	1.64	100.00
ns1830653	41.60	71.13	61.29	9.83	100.00
nu25-pr12	30.41	2.42	0.83	1.59	100.00
nursesched-medium-hint03	559.68	71.40	61.01	10.40	100.00
nursesched-sprint02	65.32	9.11	3.55	5.56	100.00
nw04	93.19	2.76	0.41	2.35	100.00
opm2-z10-s4	4,699.37	39.74	39.44	0.30	100.00
p200x1188c	36.50	61.68	61.24	0.44	100.00
peg-solitaire-a3	—	—	—	—	0.00
pg	32.61	28.14	7.94	20.20	100.00
pg5_34	32.68	16.18	3.99	12.19	100.00
pk1	34.38	39.50	37.68	1.82	80.00
qap10	652.83	5.79	5.53	0.26	100.00
rail01	2,218.39	30.05	24.45	5.60	100.00
rail02	—	—	—	—	0.00
rail507	1,162.84	35.62	32.76	2.86	100.00
ran14x18-disj-8	18.37	9.94	9.40	0.54	100.00
rd-rplusc-21	—	—	—	—	0.00
reblock115	58.88	7.12	5.68	1.44	100.00
rmatr100-p10	35.38	88.86	71.15	17.70	100.00
rmatr200-p5	395.23	86.22	77.41	8.81	100.00
rocI-4-11	—	—	—	—	0.00
rocII-5-11	—	—	—	—	0.00
roi2alpha3n4	45.51	27.05	19.08	7.98	100.00
satellites2-40	159.26	56.24	54.17	2.07	100.00

Continued on next page

Table 8: Cut collection statistics for the five test variations.

Instance	Time (s)	MIP Gap (%)			Success (%)
		Initial	Final	Delta	
satellites2-60-fs	177.65	56.24	55.03	1.21	100.00
sct2	39.92	0.06	0.03	0.02	40.00
seymour	53.70	26.50	25.42	1.07	100.00
seymour1	31.20	9.66	7.51	2.15	100.00
sing326	367.07	0.76	0.60	0.15	100.00
sing44	378.68	0.47	0.38	0.08	100.00
snp-02-004-104	253.46	6.43	1.78	4.65	100.00
sorrell3	386.86	2,828.71	2,718.01	110.70	100.00
sp150x300d	35.39	92.89	45.18	47.71	100.00
sp97ar	122.87	2.98	2.06	0.92	100.00
sp98ar	132.80	2.41	1.68	0.74	100.00
splice1k1	1,408.60	601.47	550.36	51.11	100.00
supportcase10	440.76	51.57	51.23	0.33	100.00
supportcase18	37.73	4.10	4.10	0.00	100.00
supportcase26	29.86	41.49	2.56	38.92	100.00
supportcase33	106.97	51.90	33.23	18.68	80.00
supportcase40	46.69	6.77	4.62	2.15	100.00
supportcase42	142.99	3.53	1.61	1.92	100.00
supportcase6	1,104.89	9.26	4.10	5.16	100.00
supportcase7	799.98	710.80	289.30	421.49	100.00
tbfp-network	893.59	9.60	9.02	0.58	100.00
thor50dday	4,672.92	91.02	86.28	4.74	100.00
timtab1	23.31	96.49	65.73	30.76	100.00
tr12-30	22.73	88.78	23.38	65.40	100.00
traininstance6	—	—	—	—	0.00
trento1	519.80	95.01	93.23	1.79	100.00
uccase12	1,165.25	0.08	0.04	0.03	100.00
uccase9	488.59	1.99	1.36	0.62	80.00
uct-subprob	28.22	24.66	21.07	3.59	100.00
unitcal_7	139.34	1.27	0.76	0.51	100.00
var-smallemery-m6j6	143.02	4.47	4.15	0.32	100.00

Table 9: Number of test variations for which the cut collection procedure failed, grouped by failure reason.

Instance	Memory	Numerics	Time Limit	Zero Cuts	Total
CMS750_4	—	—	—	5	5
academictimetablesmall	—	—	—	5	5
bppc4-08	—	—	—	5	5
cbs-cta	—	—	—	5	5
chromaticindex1024-7	—	—	—	5	5
chromaticindex512-7	—	—	—	5	5
comp07-2idx	—	—	—	1	1
decomp2	—	—	—	5	5
ex10	—	—	4	—	4
glass4	—	—	—	5	5
graph20-20-1rand	—	—	—	5	5
icir97_tension	—	—	—	5	5
irish-electricity	—	4	—	—	4
istanbul-no-cutoff	—	—	—	2	2
k1mushroom	—	2	—	—	2
lectsched-5-obj	—	—	—	5	5
mad	—	—	—	5	5
map10	—	4	—	—	4
map16715-04	—	3	—	—	3
markshare2	—	—	—	5	5
markshare_4_0	—	—	—	5	5
milo-v12-6-r2-40-1	—	1	—	—	1
momentum1	—	1	—	—	1
mushroom-best	—	—	—	1	1
neos-2657525-crna	—	—	—	5	5
neos-2746589-doon	—	1	—	—	1
neos-3402294-bobin	—	—	—	5	5
neos-3656078-kumeu	—	4	—	—	4
neos-4300652-rahue	—	—	—	5	5
neos-4338804-snowy	—	—	—	5	5
neos-4413714-turia	—	—	—	5	5
neos-5107597-kakapo	—	—	—	1	1
neos-5188808-nattai	—	—	—	5	5
neos-5195221-niemur	—	—	—	5	5
neos-631710	5	—	—	—	5
neos-827175	—	—	—	5	5
ns1208400	—	—	—	5	5
ns1644855	—	—	5	—	5
peg-solitaire-a3	—	—	—	5	5

Continued on next page

Table 9: Number of test variations for which the cut collection procedure failed, grouped by failure reason.

Instance	Memory	Numerics	Time Limit	Zero Cuts	Total
pk1	—	—	—	1	1
rail02	5	—	—	—	5
rd-rplusc-21	—	—	—	5	5
rocI-4-11	—	—	—	5	5
rocII-5-11	—	—	—	5	5
sct2	—	—	—	3	3
supportcase33	—	1	—	—	1
traininstance6	—	—	—	5	5
uccase9	—	1	—	—	1
Total	10	22	9	149	190

Table 10: Average performance of expert method over five test variations. Instances with failed variations have been discarded.

	Work		Nodes		Gap (%)		Speedup	Outcome
Method	baseline	expert	baseline	expert	baseline	expert	expert	
Instance								
30n20b8	18.24	5.29	253.33	27.67	0.00	0.00	3.448	Positive
neos-4532248-waihi	288.41	120.78	2,546.53	1,268.60	0.00	0.00	2.388	Positive
supportcase6	185.38	107.46	1,185.60	843.60	0.00	0.00	1.725	Positive
exp-1-500-5-5	4.02	2.42	3,992.53	2,142.53	0.00	0.00	1.664	Positive
neos-5049753-cuanza	1,298.35	820.43	1,087.87	978.67	0.00	0.00	1.583	Positive
mik-250-20-75-4	0.13	0.09	371.60	108.40	0.00	0.00	1.362	Positive
irp	0.46	0.35	82.27	78.27	0.00	0.00	1.340	Positive
supportcase40	602.30	463.11	37,040.20	37,672.33	0.00	0.00	1.301	Positive
bab2	4,206.36	3,256.79	36,313.27	27,594.80	0.00	0.00	1.292	Positive
bab6	330.50	259.09	744.27	407.53	0.00	0.00	1.276	Positive
pg5_34	4.79	3.87	1,419.53	1,137.93	0.00	0.00	1.239	Positive
nw04	1.20	0.99	14.73	1.00	0.00	0.00	1.208	Positive
neos-3754480-nidda	2.35	1.99	647.27	727.13	0.00	0.00	1.178	Positive
nursesched-sprint02	2.64	2.25	1.00	1.00	0.00	0.00	1.173	Positive
neos-1354092	58.31	51.06	7.47	3.00	0.00	0.00	1.142	Positive
fastxgemm-n2r6s0t2	244.00	214.42	52,686.53	46,502.47	0.00	0.00	1.138	Positive
air05	7.29	6.42	1,060.87	833.20	0.00	0.00	1.135	Positive
nursesched-medium-hint03	3,299.66	2,923.77	11,265.60	9,345.07	0.00	0.00	1.129	Positive
neos-3046615-murg	7,176.69	6,429.83	4,334,974.60	3,867,064.00	0.00	0.00	1.116	Positive
rail01	262.33	237.67	28.47	26.00	0.00	0.00	1.104	Positive
mas74	74.85	68.72	1,300,120.73	1,194,661.93	0.00	0.00	1.089	Positive

Continued on next page

Table 10: Average performance of expert method over five test variations. Instances with failed variations have been discarded.

	Work		Nodes		Gap (%)		Speedup	Outcome
Method	baseline	expert	baseline	expert	baseline	expert	expert	
Instance								
neos-4738912-atrato	4.17	3.85	1,308.07	1,178.53	0.00	0.00	1.083	Positive
blp-ic98	10.98	10.25	2,466.40	2,400.40	0.00	0.00	1.072	Positive
n3div36	61.89	57.86	13,820.60	13,280.20	0.00	0.00	1.070	Positive
sp97ar	1,522.83	1,425.20	124,273.60	110,659.93	0.00	0.00	1.069	Positive
neos-662469	143.74	135.51	3,149.53	2,842.73	0.00	0.00	1.061	Positive
uct-subprob	1,880.82	1,774.99	103,934.07	93,074.40	0.00	0.00	1.060	Positive
unitcal_7	8.70	8.21	6.47	4.80	0.00	0.00	1.059	Positive
neos-3627168-kasai	13,251.73	12,536.84	20,520,749.73	20,252,337.40	0.00	0.00	1.057	Positive
neos-860300	12.58	11.96	1,066.73	972.40	0.00	0.00	1.052	Positive
gen-ip054	0.43	0.41	8,658.73	8,608.80	0.00	0.00	1.051	Positive
seymour1	26.82	25.63	713.80	691.80	0.00	0.00	1.046	Positive
app1-2	73.56	71.22	600.87	622.80	0.00	0.00	1.033	Positive
neos-4954672-berkel	20,000.00	19,371.91	3,827,904.93	3,701,741.00	0.02	0.02	1.032	Positive
neos-4763324-toguru	505.87	490.42	3,557.60	3,377.40	0.00	0.00	1.032	Positive
cmflsp50-24-8-8	12,843.31	12,466.90	207,442.47	206,097.93	0.00	0.00	1.030	Positive
neos-1171448	16.33	15.88	1,884.53	1,862.47	0.00	0.00	1.028	Positive
neos5	5.02	4.89	13,780.13	13,909.60	0.00	0.00	1.026	Positive
co-100	176.61	172.35	1,603.40	1,694.53	0.00	0.00	1.025	Positive
h80x6320d	7.42	7.25	979.13	927.87	0.00	0.00	1.024	Positive
comp21-2idx	5,499.45	5,387.66	34,112.07	35,944.80	0.00	0.00	1.021	Positive
roi2alpha3n4	426.91	418.39	11,222.80	10,961.60	0.00	0.00	1.020	Positive

Continued on next page

Table 10: Average performance of expert method over five test variations. Instances with failed variations have been discarded.

	Work		Nodes		Gap (%)		Speedup	Outcome
Method Instance	baseline	expert	baseline	expert	baseline	expert	expert	
csched007	440.15	431.73	43,134.67	43,790.53	0.00	0.00	1.020	Positive
dws008-01	5,324.96	5,235.69	297,084.87	258,689.27	0.00	0.00	1.017	Positive
trento1	8,836.17	8,713.31	149,324.73	144,679.00	0.00	0.00	1.014	Positive
neos-2978193-inde	2.26	2.23	69.20	60.93	0.00	0.00	1.013	Positive
neos-1171737	0.49	0.49	363.40	388.73	0.00	0.00	1.012	Positive
neos-3216931-puriri	179.09	177.00	1,436.13	1,317.33	0.00	0.00	1.012	Positive
supportcase42	6.86	6.78	36.33	36.33	0.00	0.00	1.012	Positive
50v-10	12,432.34	12,303.09	4,812,683.73	4,948,372.67	0.01	0.01	1.011	Positive
var-smallemery-m6j6	966.06	957.76	330,968.87	358,278.93	0.00	0.00	1.009	Neutral
reblock115	4,662.00	4,624.91	4,082,020.60	3,945,952.93	0.00	0.00	1.008	Neutral
drayage-100-23	0.37	0.37	1.00	1.00	0.00	0.00	1.007	Neutral
neos-1445765	13.18	13.13	280.73	267.93	0.00	0.00	1.004	Neutral
fast0507	13,844.59	13,797.69	933,105.60	924,178.87	0.00	0.00	1.003	Neutral
timtab1	70.26	70.05	43,233.67	47,713.53	0.00	0.00	1.003	Neutral
neos-4647030-tutaki	30.30	30.23	264.20	261.80	0.00	0.00	1.002	Neutral
rmatr100-p10	696.52	696.51	5,376.67	5,472.67	0.00	0.00	1.000	Neutral
rmatr200-p5	20,000.01	20,000.01	8,735.87	8,375.27	0.11	0.11	1.000	Neutral
gmu-35-40	0.00	0.00	1.00	1.00	0.00	0.00	1.000	Neutral
drayage-25-23	0.46	0.46	3.33	1.40	0.00	0.00	1.000	Neutral
neos-957323	0.17	0.17	1.00	1.00	0.00	0.00	1.000	Neutral
neos-933966	0.05	0.05	1.00	1.00	0.00	0.00	1.000	Neutral

Continued on next page

Table 10: Average performance of expert method over five test variations. Instances with failed variations have been discarded.

Method	Work		Nodes		Gap (%)		Speedup	Outcome
Instance	baseline	expert	baseline	expert	baseline	expert	expert	
neos-960392	0.03	0.03	1.00	1.00	0.00	0.00	1.000	Neutral
ex9	3.82	3.82	0.00	0.00	0.00	0.00	1.000	Neutral
neos8	0.06	0.06	0.00	0.00	0.00	0.00	1.000	Neutral
cvs16r128-89	0.02	0.02	1.00	1.00	0.00	0.00	1.000	Neutral
gmu-35-50	0.00	0.00	1.80	2.93	0.00	0.00	1.000	Neutral
cod105	0.01	0.01	0.00	0.00	0.00	0.00	1.000	Neutral
neos-3024952-loue	20,000.00	20,000.00	3,096,604.53	3,019,827.40	0.04	0.04	1.000	Neutral
supportcase7	1.29	1.29	1.00	1.00	0.00	0.00	1.000	Neutral
supportcase26	0.02	0.02	0.00	0.00	0.00	0.00	1.000	Neutral
mzzv42z	0.03	0.03	0.20	0.20	0.00	0.00	1.000	Neutral
neos-2987310-joes	0.77	0.77	0.20	0.20	0.00	0.00	1.000	Neutral
mzzv11	0.02	0.02	0.00	0.00	0.00	0.00	1.000	Neutral
sing326	20,000.01	20,000.01	70,550.87	71,697.40	0.00	0.00	1.000	Neutral
ns1760995	79.79	79.81	1.00	1.00	0.00	0.00	1.000	Neutral
eilA101-2	15,124.17	15,146.40	30,002.40	28,593.80	0.02	0.03	0.999	Neutral
buildingenergy	81.45	81.66	2.33	2.20	0.00	0.00	0.997	Neutral
neos-1122047	14.92	14.98	1.00	1.00	0.00	0.00	0.996	Neutral
b1c1s1	1,366.98	1,372.73	29,093.67	30,053.93	0.00	0.00	0.996	Neutral
neos-787933	0.46	0.47	1.00	1.00	0.00	0.00	0.996	Neutral
binkar10_1	0.35	0.36	200.00	170.40	0.00	0.00	0.994	Neutral
neos-5093327-huahum	4,774.34	4,809.58	173,008.27	156,362.20	0.00	0.00	0.993	Neutral

Continued on next page

Table 10: Average performance of expert method over five test variations. Instances with failed variations have been discarded.

Method	Work		Nodes		Gap (%)		Speedup	Outcome
Instance	baseline	expert	baseline	expert	baseline	expert	expert	
neos-3555904-turama	19,834.82	20,000.00	7,158.93	7,279.07	0.12	0.13	0.992	Neutral
neos17	2.95	2.98	3,544.20	3,552.73	0.00	0.00	0.989	Negative
satellites2-60-fs	75.78	76.65	128.47	104.93	0.00	0.00	0.989	Negative
sorrell3	19,609.94	19,838.65	129,172.87	133,570.20	0.07	0.07	0.988	Negative
uccase12	7.52	7.64	1.00	1.00	0.00	0.00	0.984	Negative
n5-3	7.81	7.98	1,046.73	762.87	0.00	0.00	0.979	Negative
qap10	30.17	30.87	24.80	26.27	0.00	0.00	0.977	Negative
supportcase10	7,108.03	7,296.36	2,897.67	2,980.20	0.00	0.00	0.974	Negative
ns1830653	37.31	38.52	3,740.27	3,790.80	0.00	0.00	0.969	Negative
assign1-5-8	7,746.01	8,038.48	8,846,355.07	9,585,410.13	0.00	0.00	0.964	Negative
rail507	10,594.29	11,042.93	634,507.67	667,217.27	0.00	0.00	0.959	Negative
nu25-pr12	0.27	0.28	74.13	65.47	0.00	0.00	0.959	Negative
glass-sc	1,224.76	1,278.07	115,443.93	120,406.33	0.00	0.00	0.958	Negative
ic97_potential	114.06	119.16	39,857.93	41,029.80	0.00	0.00	0.957	Negative
gen-ip002	43.49	45.62	457,816.87	463,323.27	0.00	0.00	0.953	Negative
net12	80.58	84.98	641.67	618.93	0.00	0.00	0.948	Negative
tr12-30	172.32	182.01	152,344.80	160,146.87	0.00	0.00	0.947	Negative
neos-950242	27.31	28.90	767.53	775.87	0.00	0.00	0.945	Negative
n2seq36q	6.01	6.39	557.27	635.87	0.00	0.00	0.941	Negative
thor50dday	1,478.55	1,585.79	10,156.20	10,056.67	0.00	0.00	0.932	Negative
neos-1456979	28.72	31.02	2,523.20	2,982.33	0.00	0.00	0.926	Negative

Continued on next page

Table 10: Average performance of expert method over five test variations. Instances with failed variations have been discarded.

Method	Work		Nodes		Gap (%)		Speedup	Outcome
Instance	baseline	expert	baseline	expert	baseline	expert	expert	
tbfp-network	2,628.70	2,840.34	24,350.13	25,805.80	0.00	0.00	0.925	Negative
sp98ar	161.74	175.76	18,480.60	19,876.80	0.00	0.00	0.920	Negative
pg	0.19	0.21	34.60	31.93	0.00	0.00	0.917	Negative
neos-4387871-tavua	15,375.25	16,824.75	78,364.73	77,523.00	0.01	0.02	0.914	Negative
ran14x18-disj-8	1,027.11	1,134.98	288,543.33	322,673.53	0.00	0.00	0.905	Negative
neos-3083819-nubu	0.27	0.30	599.80	742.13	0.00	0.00	0.896	Negative
leo2	324.71	365.80	77,151.53	89,866.00	0.00	0.00	0.888	Negative
graphdraw-domain	158.94	179.11	160,303.87	144,203.27	0.00	0.00	0.887	Negative
atlanta-ip	570.91	650.28	2,349.13	2,861.87	0.00	0.00	0.878	Negative
seymour	69.97	79.94	3,645.87	4,221.00	0.00	0.00	0.875	Negative
mc11	3.11	3.56	488.80	458.53	0.00	0.00	0.871	Negative
mas76	4.98	5.81	126,141.53	153,998.27	0.00	0.00	0.857	Negative
neos-911970	1.98	2.31	1,299.73	1,744.33	0.00	0.00	0.856	Negative
cost266-UUE	1,552.67	1,907.05	85,829.73	82,118.93	0.00	0.00	0.814	Negative
p200x1188c	0.12	0.15	1.00	1.00	0.00	0.00	0.810	Negative
sing44	3,222.24	4,005.08	29,702.67	38,115.53	0.00	0.00	0.805	Negative
sp150x300d	0.01	0.02	1.00	1.00	0.00	0.00	0.800	Negative
nexp-150-20-8-5	160.25	200.93	1,645.13	1,573.27	0.00	0.00	0.798	Negative
satellites2-40	69.64	88.87	130.00	93.60	0.00	0.00	0.784	Negative
eil33-2	10.61	13.55	4,509.73	4,702.00	0.00	0.00	0.783	Negative
neos-3381206-awhea	1,695.76	2,228.19	3,323,889.93	4,090,279.73	0.00	0.00	0.761	Negative

Continued on next page

Table 10: Average performance of expert method over five test variations. Instances with failed variations have been discarded.

Method Instance	Work baseline	 expert	Nodes baseline	 expert	Gap (%) baseline	 expert	Speedup expert	Outcome
netdiversion	71.35	95.00	59.47	57.33	0.00	0.00	0.751	Negative
supportcase18	10,741.68	14,460.71	3,582,093.07	2,467,351.00	0.03	0.03	0.743	Negative
app1-1	0.76	1.07	1.00	6.67	0.00	0.00	0.716	Negative
lotsize	2,826.84	4,125.24	67,638.47	98,598.13	0.00	0.00	0.685	Negative
beasleyC3	0.58	0.85	69.47	180.93	0.00	0.00	0.674	Negative
snp-02-004-104	10.74	16.28	1.00	257.53	0.00	0.00	0.660	Negative
neos-1582420	14.59	24.97	3,239.27	4,309.53	0.00	0.00	0.584	Negative
neos-873061	1,386.06	2,652.10	9,970.87	13,170.53	0.00	0.00	0.523	Negative
enlight_hard	0.00	0.00	0.00	0.00	0.00	0.00	—	Neutral
Arithmetic Mean	2,324.63	2,368.70	467,229.43	462,953.60	0.00	0.00	1.027	—

Table 11: Average work units of different methods on the 50 benchmark instances.

				Work			
Method	baseline	expert	ml:far:10	ml:near:1	ml:near:10	ml:near:50	ml:rand:10
Instance							
30n20b8	18.24	5.29	4.84	5.29	4.84	4.84	4.67
neos-4532248-waihi	288.41	120.78	120.78	120.78	120.78	120.78	120.78
supportcase6	185.38	107.46	109.41	118.19	112.15	105.69	118.24
exp-1-500-5-5	4.02	2.42	0.78	3.51	1.02	0.39	0.88
neos-5049753-cuanza	1,298.35	820.43	558.99	1,211.00	584.02	362.67	529.75
mik-250-20-75-4	0.13	0.09	0.08	0.08	0.09	0.09	0.08
irp	0.46	0.35	0.43	0.46	0.44	0.40	0.44
supportcase40	602.30	463.11	528.64	467.38	543.01	563.78	525.53
bab2	4,206.36	3,256.79	2,357.06	6,684.54	5,677.42	1,519.54	1,524.04
bab6	330.50	259.09	346.32	878.22	288.99	189.81	1,872.41
pg5_34	4.79	3.87	3.99	4.68	3.64	4.58	4.26
nw04	1.20	0.99	0.99	1.08	0.99	0.99	0.98
neos-3754480-nidda	2.35	1.99	2.00	2.28	1.97	2.06	2.20
nursesched-sprint02	2.64	2.25	1.97	2.11	2.04	1.97	2.01
neos-1354092	58.31	51.06	24.33	78.20	24.53	24.30	23.48
fastxgemm-n2r6s0t2	244.00	214.42	371.06	448.85	366.57	286.34	288.11
air05	7.29	6.42	7.57	7.24	7.31	7.78	7.09
nursesched-medium-hint03	3,299.66	2,923.77	3,767.27	5,254.99	2,902.49	3,201.57	4,162.71
neos-3046615-murg	7,176.69	6,429.83	6,900.68	7,915.76	7,251.60	6,818.07	6,770.32
rail01	262.33	237.67	144.25	243.37	131.71	121.42	133.12
mas74	74.85	68.72	68.72	68.72	68.72	68.72	68.72

Continued on next page

Table 11: Average work units of different methods on the 50 benchmark instances.

	Work						
Method	baseline	expert	ml:far:10	ml:near:1	ml:near:10	ml:near:50	ml:rand:10
Instance							
neos-4738912-atrato	4.17	3.85	4.55	3.97	3.57	4.84	4.33
blp-ic98	10.98	10.25	10.86	10.18	11.18	11.01	11.10
n3div36	61.89	57.86	58.08	63.42	62.86	62.65	60.31
sp97ar	1,522.83	1,425.20	1,252.22	1,527.44	1,437.84	1,412.39	1,547.47
neos-662469	143.74	135.51	137.26	105.91	108.84	161.08	137.17
uct-subprob	1,880.82	1,774.99	1,676.15	1,693.15	1,665.84	1,679.80	1,947.67
unitcal_7	8.70	8.21	8.90	8.38	8.47	8.11	8.40
neos-3627168-kasai	13,251.73	12,536.84	9,382.52	8,684.10	10,184.75	7,093.65	10,403.42
neos-860300	12.58	11.96	10.73	10.28	11.40	11.40	12.26
gen-ip054	0.43	0.41	0.41	0.41	0.40	0.41	0.42
seymour1	26.82	25.63	24.90	28.60	24.54	24.38	24.60
app1-2	73.56	71.22	70.11	68.72	66.54	69.56	70.05
neos-4954672-berkel	20,000.00	19,371.91	18,687.19	19,788.81	19,524.47	19,676.67	19,510.77
neos-4763324-toguru	505.87	490.42	458.11	464.22	454.78	519.89	451.11
cmflsp50-24-8-8	12,843.31	12,466.90	12,532.65	12,575.32	12,563.44	12,862.92	12,380.33
neos-1171448	16.33	15.88	16.63	16.61	15.40	18.21	16.05
neos5	5.02	4.89	5.17	5.19	5.19	4.91	4.84
co-100	176.61	172.35	158.73	173.61	178.25	170.95	176.52
h80x6320d	7.42	7.25	7.67	7.71	8.22	8.04	7.91
comp21-2idx	5,499.45	5,387.66	5,315.60	5,745.68	5,944.23	5,717.34	5,387.90
roi2alpha3n4	426.91	418.39	399.15	397.20	398.52	431.24	448.74

Continued on next page

Table 11: Average work units of different methods on the 50 benchmark instances.

	Work						
Method	baseline	expert	ml:far:10	ml:near:1	ml:near:10	ml:near:50	ml:rand:10
Instance							
csched007	440.15	431.73	436.38	423.44	393.33	392.07	412.22
dws008-01	5,324.96	5,235.69	5,350.57	5,014.84	5,340.49	5,461.96	5,563.49
trento1	8,836.17	8,713.31	8,695.45	8,652.27	8,721.58	8,681.41	8,641.24
neos-2978193-inde	2.26	2.23	2.23	2.34	2.28	2.03	2.26
neos-1171737	0.49	0.49	0.49	0.46	0.49	0.49	0.50
neos-3216931-puriri	179.09	177.00	167.40	165.84	204.90	154.51	155.89
supportcase42	6.86	6.78	7.36	6.68	7.20	7.56	7.38
50v-10	12,432.34	12,303.09	12,300.63	12,085.50	12,057.22	12,319.07	12,351.77
Arithmetic Average	2,035.39	1,924.89	1,849.97	2,024.94	1,950.01	1,807.49	1,918.12

Table 12: Average speedup (with respect to work units) of different methods on the 50 benchmark instances.

	Speedup					
Method	expert	ml:far:10	ml:near:1	ml:near:10	ml:near:50	ml:rand:10
Instance						
30n20b8	3.45	3.77	3.45	3.77	3.77	3.90
neos-4532248-waihi	2.39	2.39	2.39	2.39	2.39	2.39
supportcase6	1.73	1.69	1.57	1.65	1.75	1.57
exp-1-500-5-5	1.66	5.15	1.14	3.96	10.44	4.55
neos-5049753-cuanza	1.58	2.32	1.07	2.22	3.58	2.45
mik-250-20-75-4	1.36	1.60	1.56	1.49	1.47	1.51
irp	1.34	1.08	1.00	1.05	1.17	1.05
supportcase40	1.30	1.14	1.29	1.11	1.07	1.15
bab2	1.29	1.78	0.63	0.74	2.77	2.76
bab6	1.28	0.95	0.38	1.14	1.74	0.18
pg5_34	1.24	1.20	1.02	1.32	1.05	1.12
nw04	1.21	1.21	1.11	1.22	1.21	1.22
neos-3754480-nidda	1.18	1.17	1.03	1.19	1.14	1.07
nursesched-sprint02	1.17	1.34	1.25	1.29	1.34	1.31
neos-1354092	1.14	2.40	0.75	2.38	2.40	2.48
fastxgemm-n2r6s0t2	1.14	0.66	0.54	0.67	0.85	0.85
air05	1.14	0.96	1.01	1.00	0.94	1.03
nursesched-medium-hint03	1.13	0.88	0.63	1.14	1.03	0.79
neos-3046615-murg	1.12	1.04	0.91	0.99	1.05	1.06
rail01	1.10	1.82	1.08	1.99	2.16	1.97
mas74	1.09	1.09	1.09	1.09	1.09	1.09

Continued on next page

Table 12: Average speedup (with respect to work units) of different methods on the 50 benchmark instances.

Method Instance	Speedup expert	ml:far:10	ml:near:1	ml:near:10	ml:near:50	ml:rand:10
neos-4738912-atrato	1.08	0.92	1.05	1.17	0.86	0.96
blp-ic98	1.07	1.01	1.08	0.98	1.00	0.99
n3div36	1.07	1.07	0.98	0.98	0.99	1.03
sp97ar	1.07	1.22	1.00	1.06	1.08	0.98
neos-662469	1.06	1.05	1.36	1.32	0.89	1.05
uct-subprob	1.06	1.12	1.11	1.13	1.12	0.97
unitcal_7	1.06	0.98	1.04	1.03	1.07	1.04
neos-3627168-kasai	1.06	1.41	1.53	1.30	1.87	1.27
neos-860300	1.05	1.17	1.22	1.10	1.10	1.03
gen-ip054	1.05	1.04	1.05	1.06	1.06	1.03
seymour1	1.05	1.08	0.94	1.09	1.10	1.09
app1-2	1.03	1.05	1.07	1.11	1.06	1.05
neos-4954672-berkel	1.03	1.07	1.01	1.02	1.02	1.03
neos-4763324-toguru	1.03	1.10	1.09	1.11	0.97	1.12
cmflsp50-24-8-8	1.03	1.02	1.02	1.02	1.00	1.04
neos-1171448	1.03	0.98	0.98	1.06	0.90	1.02
neos5	1.03	0.97	0.97	0.97	1.02	1.04
co-100	1.02	1.11	1.02	0.99	1.03	1.00
h80x6320d	1.02	0.97	0.96	0.90	0.92	0.94
comp21-2idx	1.02	1.03	0.96	0.93	0.96	1.02
roi2alpha3n4	1.02	1.07	1.07	1.07	0.99	0.95

Continued on next page

Table 12: Average speedup (with respect to work units) of different methods on the 50 benchmark instances.

	Speedup					
Method	expert	ml:far:10	ml:near:1	ml:near:10	ml:near:50	ml:rand:10
Instance						
csched007	1.02	1.01	1.04	1.12	1.12	1.07
dws008-01	1.02	1.00	1.06	1.00	0.97	0.96
trento1	1.01	1.02	1.02	1.01	1.02	1.02
neos-2978193-inde	1.01	1.01	0.96	0.99	1.11	1.00
neos-1171737	1.01	1.02	1.06	1.00	1.01	0.99
neos-3216931-puriri	1.01	1.07	1.08	0.87	1.16	1.15
supportcase42	1.01	0.93	1.03	0.95	0.91	0.93
50v-10	1.01	1.01	1.03	1.03	1.01	1.01
Arithmetic Average	1.20	1.32	1.11	1.28	1.49	1.31

Table 13: Average time of different methods on the 14 hard benchmark instances.

Method	Time (s)						
	baseline	exp+col	ml:far:10	ml:near:1	ml:near:10	ml:near:50	ml:rand:10
Instance							
50v-10	6,504.97	6,576.95	6,258.86	6,273.33	6,080.06	6,363.89	6,198.39
bab2	2,982.14	6,945.93	2,442.47	3,894.55	4,546.77	2,340.32	1,860.93
cmflsp50-24-8-8	5,561.02	6,239.72	5,121.11	5,118.99	5,127.74	5,382.54	5,099.62
comp21-2idx	3,304.79	3,294.17	3,085.97	3,274.97	3,391.48	3,297.29	3,143.48
dws008-01	3,125.89	3,014.31	2,892.82	2,692.95	2,916.08	2,907.35	2,985.44
neos-3046615-murg	4,390.31	4,169.14	3,888.79	4,437.08	4,062.52	3,853.29	3,887.18
neos-3627168-kasai	12,509.69	11,740.51	8,366.29	7,792.25	9,355.04	6,514.19	9,445.87
neos-4954672-berkel	11,622.77	11,254.44	9,899.46	10,485.50	10,313.65	10,594.96	10,393.03
neos-5049753-cuanza	956.51	2,017.10	694.37	1,024.16	779.54	2,263.72	657.60
nursesched-medium-hint03	1,800.14	2,163.83	2,049.98	2,627.00	1,551.07	1,775.10	2,164.72
sp97ar	876.20	933.90	630.56	755.62	710.02	755.65	777.16
supportcase40	388.69	367.56	328.92	285.58	311.37	460.80	331.54
trento1	3,731.88	4,204.84	3,606.52	3,518.97	3,534.85	3,674.12	3,634.36
uct-subprob	1,049.20	1,034.66	874.07	902.56	891.27	896.84	1,030.74
Arithmetic Average	4,200.30	4,568.36	3,581.44	3,791.68	3,826.53	3,648.58	3,686.43

Table 14: Average speedup (with respect to time) of different methods on the 14 hard benchmark instances.

Method / Instance	Speedup: exp+col	ml:far:10	ml:near:1	ml:near:10	ml:near:50	ml:rand:10
50v-10	0.99	1.04	1.04	1.07	1.02	1.05
bab2	0.43	1.22	0.77	0.66	1.27	1.60
cmflsp50-24-8-8	0.89	1.09	1.09	1.08	1.03	1.09
comp21-2idx	1.00	1.07	1.01	0.97	1.00	1.05
dws008-01	1.04	1.08	1.16	1.07	1.08	1.05
neos-3046615-murg	1.05	1.13	0.99	1.08	1.14	1.13
neos-3627168-kasai	1.07	1.50	1.61	1.34	1.92	1.32
neos-4954672-berkel	1.03	1.17	1.11	1.13	1.10	1.12
neos-5049753-cuanza	0.47	1.38	0.93	1.23	0.42	1.45
nursesched-medium-hint03	0.83	0.88	0.69	1.16	1.01	0.83
sp97ar	0.94	1.39	1.16	1.23	1.16	1.13
supportcase40	1.06	1.18	1.36	1.25	0.84	1.17
trento1	0.89	1.03	1.06	1.06	1.02	1.03
uct-subprob	1.01	1.20	1.16	1.18	1.17	1.02
Arithmetic Average	0.91	1.17	1.08	1.11	1.08	1.15