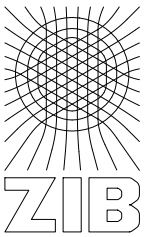
Empirical Analysis of Solving Phases in Mixed Integer Programming

Masterarbeit bei
Prof. Dr. Thorsten Koch

vorgelegt von
Gregor Hendel[1]
Technische Universität Berlin
Fachbereich Mathematik



Berlin, 27. August 2014

[1]Konrad Zuse Zentrum für Informationstechnik Berlin, hendel@zib.de

Hiermit versichere ich die selbstständige und eigenhändige Anfertigung dieser Arbeit an Eides statt.

(Ort, Datum)	(Unterschrift)

Zusammenfassung

Viele Planungs- und Entscheidungsprobleme aus Industrie und Wirtschaft lassen sich abstrakt als Gemischt-ganzzahliges Optimierungsproblem (engl.: *Mixed Integer Program, MIP*) formulieren. Moderne Lösungssoftware für allgemeine MIPs kombiniert das sogenannte Branch-and-Bound-Verfahren mit einer Vielzahl von Zusatzkomponenten wie zum Beispiel Primalheuristiken und Schnittebenenverfahren. Der Löseprozess lässt sich in die folgenden Phasen unterteilen: Am Anfang steht die *Zulässigkeitsphase*, in der der Löser eine zulässige Lösung für das Problem sucht. Nach der ersten Lösung setzt die *Verbesserungsphase* ein. Diese dauert an, bis eine Optimallösung für das vorgegebene Problem gefunden wurde. Die verbleibende Zeit verbringt der Löser in der *Beweisphase* der Optimalität. In der vorliegenden Arbeit befassen wir uns mit Fragestellungen, die das Ausnutzen von Phaseninformationen für den Einsatz der eingangs erwähnten Löserkomponenten betreffen:

1. Ist es möglich, den gesamten Löseprozess durch den Einsatz phasenspezifischer Einstellungen und Komponenten in jeder Phase zu beschleunigen?

2. Gibt es Kriterien, auf deren Grundlage sich ein praxistauglicher Übergang zwischen der Verbesserungsphase und der Beweisphase ermöglichen lässt?

Zur Beantwortung betrachten wir die Lösephasen zunächst getrennt voneinander. Hier identifizieren wir Komponenten und dazugehörige Parameter des MIP-Lösers SCIP, von deren Einsatz wir auf der Grundlage von Kenntnissen aus der einschlägigen MIP-Literatur eine Verkürzung der Phasendauer versprechen. Die besten individuellen Einstellungen für die einzelnen Phasen aus einer Reihe von Experimenten werden anschließend in einem phasen-basierenden Löser kombiniert. Dieser kann die Lösezeit von SCIP um bis zu 11 % auf einer Reihe von schweren Instanzen verbessern und drei zusätzliche Instanzen lösen. Die Erkennung des Übergangs zwischen der Verbesserungs- und der Beweisphase erfolgt bis zu diesem Zeitpunkt exakt.

Anschließend entwickeln wir Kriterien für einen heuristischen Phasenübergang. Insgesamt drei verschiedene solcher heuristischen Ansätze werden in dieser Arbeit vorgestellt. Das erste Kriterium nutzt eine stetige Approximation des diskreten primalen Löseverlaufs mit Hilfe einer logarithmischen Regressionskurve. Die anderen beiden Kriterien nutzen globale Informationen über den Zustand des Suchbaumes für einen heuristischen Phasenübergang. Besonders erwähnt werden sollte hier

ein auf (noch zu definierenden) *Knotenrängen* basierendes Kriterium, mit dessen Hilfe wir einen phasen-basierenden Löser realisieren konnten, der ein ähnlich gutes Resultat wie der exakte phasen-basierende Löser hinsichtlich der Gesamtlösezeit erzielt. Die Auswertungen zeigen, dass alle Kriterien dazu tendieren, den wahren Phasenübergang in die Beweisphase zu unterschätzen. Dieser Umstand kann als Indikator dienen, ob die beste gefundene Lösung beim Erreichen des Zeitlimits schon optimal ist oder nicht. Dies trifft insbesondere auf das *best-estimate*-Kriterium zu.

Im Rahmen dieser Arbeit wurden eine Knotenauswahlregel, eine Branchingregel und zwei Primalheuristiken aus der aktuellen Literatur in SCIP integriert. Darüber hinaus stellen wir in dieser Arbeit zwei neuartige Modifikationen der Standard-Branchingregel von SCIP vor. Neben Entwicklungen am Löser stellen wir außerdem mit dem `ipet`-Modul (**I**nteractive **P**ython **e**valuation **t**ools) ein Python-Paket zur Arbeit mit Benchmark-Daten zur Verfügung. Das Paket verfügt über eine Benutzeroberfläche, die einige Funktionen zum Einlesen von Daten sowie deren Visualisierung und Speicherung bereitstellt.

Danksagungen

Ganz besonders herzlich bedanken möchte ich mich bei Professor Koch für die Themenstellung, die Begutachtung und zahlreiche Tipps zum Verfassen der vorliegenden Arbeit. Gleiches gilt auch für meinen Zweitgutachter Timo Berthold, dem ich herzlich für seine allzeit offene Tür und viele interessante Diskussionen sowohl zum Thema als auch darüber hinaus danken möchte. Beim Verfassen der einzelnen Kapitel wurde ich mit vielen hilfreichen Kommentare durch meine Freunde Heide Hoppmann, Richard Sieg, Kai Hennig und Michael Winkler tatkräftig unterstützt, dafür ein herzliches Dankeschön. Außerdem bedanke ich mich bei meinen Kollegen und Schlagzeugern Tobias Achterberg, Matthias Miltenberger, Gerald Gamrath und Ambros Gleixner sowie meinen Bürokollegen Yuji Shinano und Stephen J Maher für die angenehme Arbeitsatmosphäre und viele interessante Diskussionen zwischendurch.

Meine Freundin Katharina hat ebenso einen großen Anteil am Entstehen dieser Arbeit durch eine ihr eigene Mischung aus aufmunternden Worten, viel Geduld, und sogar Korrekturarbeit, die ich in Sushi-Einladungen gar nicht werde aufwiegen können.

Contents

Chapter 1

Introduction

Numerous planning tasks in areas such as, e.g., infrastructure design, crew scheduling, or various industrial problems have become too difficult to be solved efficiently by hand. Many of the questions arising in those fields can be formulated in terms of Mixed Integer Programs (MIPs). Sophisticated MIP solving software packages combine the so-called branch-and-bound approach with a variety of additional techniques such as, e.g., primal heuristics, presolving techniques, and cutting plane separation. A lot of algorithmic decisions within a MIP solver are subject to user parameter choices. These parameters are usually tuned to yield a good overall performance on a variety of MIP problems when statically applied during the entire search process.

In this thesis, we consider a decomposition of the solving process into a sequence of three solving phases, each with a different phase objective. The objective of the first phase is to find a feasible solution. During the second phase, a sequence of incumbent solutions gets constructed until the incumbent is eventually optimal. Proving optimality is the central objective of the remaining third phase. Our aim is to construct a phase-based solver that dynamically reacts on phase transitions with an appropriate setting, based on the MIP solver Scip [sci]. Two main questions naturally arise from the concept of solving phases:

1. Can we make better use of the Scip components inside such a dynamic, phase-based solver that reacts on phase transitions with an adaptation of its settings, compared to a static setting throughout the solving process?

2. What possible heuristic criteria could be used to benefit of this concept in practice, where it is in general not feasible to detect the optimality of an incumbent solution prior to the termination of the solving process?

We give answers to these two questions by means of an empirical analysis. First, we identify specific Scip components from which we expect the largest influence on the individual phase objectives. We extended Scip with the necessary functionalities to detect and react on the phase transitions. Starting from a test set of publicly available MIP instances, we collect individual problem sets for each phase that we expect to be affected by our changes. During those experiments,

the detection of the phase transition between the second and the third phase is performed by means of an oracle that knows the optimal solutions for all our test instances.

Research on a heuristic replacement measure of the mentioned oracle is the second focus of this thesis. We introduce three different heuristic criteria that are all original work as to the author's knowledge. The first criterion uses a log-linear regression of the primal progress as continuous approximation of the solving process. The other two criteria use global information about the search tree state of the solver. One criterion uses the *best-estimate* by Benichou et al. [BGG$^+$71] for estimating the objective of the best attainable solution in a subtree; The incumbent is assumed to be optimal when all open nodes have a best-estimate that is not better than the current incumbent objective. The best-estimate is also the basis for our notion of *node ranks*, which are the basis for a third heuristic criterion. We present computational results, which indicate that the use of the node rank criterium for heuristic phase transition yields speed-ups similar to those obtained with a phase-based solver that exactly detects the phase transitions.

This thesis is organized as follows: First, we introduce notation and give definitions of MIP and branch-and-bound in Chapter 2. We present an overview of the components that are already included in SCIP. Furthermore, Chapter 2 contains a categorization of the algorithmic components based on their individual impact on the primal and dual progress of the solver. This distinction served as a starting point for later considerations regarding the solving phase experiments.

In Chapter 3, we discuss the performance measures that we use in this thesis together with a review of the development of empirical algorithmic science. In the second part of this chapter, we give an overview of non-parametric hypothesis tests that we use to separate algorithmic improvement from random noise caused by the phenomenon of performance variability [Dan08, KAA$^+$11].

Chapter 4 introduces the decomposition of the solving process into a sequence of three phases. A formal definition is followed by a discussion of the most important techniques for each phase. We also describe newly implemented components from the recent MIP-literature in detail. Furthermore, we introduce two modifications to the reliability pseudo-cost branching rule for the third phase. Both modifications are original developments by the author of this thesis. Finally, we give a detailed presentation of the aforementioned heuristic criteria that we use for the phase transition. The chapter also includes a presentation of historical work on tree size estimates.

Chapter 5 gives an overview of the results that we obtained from computational experiments with our modifications to SCIP. In the order of the previous chapter, we first evaluate the proposed changes individually for each phase. Then, we analyze the heuristic criteria w.r.t. their predictive performance. Finally, we combine the best individual phase search algorithms in a phase-based solver that uses the heuristic phase-transition criteria, which is compared to its exact counterpart, and SCIP with default settings.

In Chapter 6, we introduce IPET, the **I**nteractive **P**ython **e**valuation **t**ool. IPET is a graphical user interface written in the Python programming language by the

author of this thesis. It facilitates the handling of SCIP benchmark data by collecting more than 1000 distinct data fields from large log files, which are usually generated during computational experiments with SCIP. Additional data can be quickly specified for reading through the interface. Furthermore, the interface offers several ways for visualizing the data in tables or graphically. Finally, several file formats such as e.g., CSV and LaTeX, are supported to export the data for publications or for further evaluation with other software tools. The chapter serves as a quick tutorial that covers the most important functionalities of the user interface. We also give recipes for the use of the underlying Python library inside custom scripts. Finally, our findings are briefly summarized in Chapter 7.

We show instance-wise experimental outcome for every of the conducted experiments for Chapters 2 and 5 in the appendix.

Chapter 2

Components of the branch-and-bound MIP solver SCIP

This chapter familiarizes the reader with techniques utilized for solving general Mixed Integer Programs (MIP). The MIP solving algorithm that we focus on is the LP-based branch-and-bound method, which is part of all state-of-the-art MIP solvers, both commercial ones such as, e.g., CPLEX [cpl], GUROBI [gur], XPRESS [fic] or solvers such as CBC [cbc] and SCIP [sci]. There exist numerous algorithms to support the branch-and-bound search such as, e.g., primal heuristics or cutting plane algorithms, which we call "components" in this section. In fact, the use of such auxiliary components increases the solving abilities of the branch-and-bound search for many MIP models. After formal definitions of a Mixed Integer Program and the branch-and-bound method, we introduce the MIP solver SCIP, which we use for the computations of this thesis and give an overview of the components included in SCIP.

2.1 Mixed integer programming and branch-and-bound

The following definition of a Mixed Integer Program introduces important notation used throughout this thesis.

Definition 1 (Mixed Integer Program)**.** *Let $n, m \in \mathbb{N}$, $l, u \in \mathbb{R}^n_\infty$, $A \in \mathbb{R}^{m \times n}$ a real matrix, $b \in \mathbb{R}^m$, and $c \in \mathbb{R}^n$. Let further $\mathcal{I} \dot\cup \mathcal{C}$ be a disjoint partition of $\{1, \ldots, n\}$.* A Mixed Integer Program (MIP) *is a minimization problem of the form*

$$\begin{aligned} & \min c^t x && (2.1)\\ s.t. \quad & Ax \leq b && (2.2)\\ & l \leq x \leq u && (2.3)\\ & x_j \in \mathbb{Z} \quad \forall j \in \mathcal{I} && (2.4) \end{aligned}$$

We call $\mathcal{C}$ and $\mathcal{I}$ the *continuous* and *integer* variables, respectively. Lower and upper bounds on the variables are denoted by l, u. The restrictions (2.2), (2.3) and (2.4) of a MIP P are called *(linear) constraints*, *bound constraints*, and *integrality restrictions*, respectively. A vector $y \in \mathbb{R}^n$ is called a *(feasible) solution* of P, if it satisfies all constraints of P. The set of solutions is the *solution space of* P, denoted by $\mathcal{S}(P)$. A solution $y^{\text{opt}} \in \mathcal{S}(P)$ with minimum objective value is called *optimal solution*, and its objective value is the optimal objective value $c^{\text{opt}} := c^t y^{\text{opt}}$ of P. P is called *feasible* if it has a solution, and *infeasible* otherwise. If a MIP P only contains continuous variables, i.e. $\mathcal{I} = \emptyset$, we speak of P as a *linear program (LP)*. For a MIP P, we obtain its *LP-relaxation* P^{LP} by ignoring the integrality restrictions (2.4):

$$P^{\text{LP}} := \min\{c^t x : \; Ax \leq b, l \leq x \leq u\} \qquad \text{(LP-relaxation)}$$

Mixed integer programming in the stated, general form has been shown to be $\mathcal{NP}$-hard [GJ79], although some of its variants are solvable in polynomial time, in particular LP [Kha79]. Whenever the LP-relaxation P^{LP} is neither infeasible nor unbounded, the objective value of an optimal solution y^{LP} of P^{LP} provides a lower bound to the optimal objective value of P (if existent). This fact is used within the basic variant of the *branch-and-bound* procedure, which starts by solving the LP-relaxation of a MIP P to optimality. If the found, optimal LP-solution y^{LP} is feasible for P, it is also optimal for P. Otherwise, there exists an integer variable $j \in \mathcal{I}$ whose LP-solution value y_j^{LP} is fractional. We call such variables *fractionals* and use $\mathcal{F}$ to denote the set of fractionals. Whenever $\mathcal{F} \neq \emptyset$, P is split into subproblems $P_1, \ldots, P_k$, $k \in \mathbb{N}$, in a way that

1. it is ensured that y^{LP} is not feasible for any of the sub-LP-relaxations P_l^{LP}, $1 \leq l \leq k$, but
2. for every solution y that is feasible for P, there exists a $1 \leq l \leq k$ such that y is feasible for P_l.

This problem division is called *branching*. In this thesis, we will only consider branching schemes with $k = 2$, where the subproblems P_-, P_+ are obtained by extending P by an additional inequality $x_j \leq \lfloor y_j^{\text{LP}} \rfloor$ or $x_j \geq \lceil y_j^{\text{LP}} \rceil$, respectively, for a fractional j. In our MIP-notation, such bound changes are reflected by adjusting the upper or lower bound of j, $u_j^- \leftarrow \lfloor y_j^{\text{LP}} \rfloor$ and $l_j^+ \leftarrow \lceil y_j^{\text{LP}} \rceil$. Clearly, these bound changes render y^{LP} infeasible for both sub-LP-relaxations, but every feasible solution for P is feasible for exactly one of the two subproblems.

When performing a sequence of branchings, a tree $T = (V, A)$ (the branch-and-bound tree) is created, which has a node for every MIP created through branching, and an arc (Q, Q') if Q' is obtained from Q by branching. We therefore call two MIPs P and Q "nodes" if the relationship between P and Q is important, in particular if Q arose from P by branching. For the *bounding* step of the procedure, the algorithm keeps track of the best feasible solution found so far, the so-called *incumbent* $\hat{y}$. Using the notation $\delta(Q)$ for the best calculated lower bound to the optimal objective value of Q, a node Q can be pruned if $\delta(Q) \geq c^t \hat{y}$ because no

solution can be found in the subtree rooted at Q which is better than the current incumbent. We call $\delta(Q)$ the *dual bound* of Q. For every arc from Q to Q' in the tree, we can immediately set $\delta(Q') \leftarrow \delta(Q)$ because Q' arose from Q by a branching. On the other hand, the dual bound of a node can be increased based on the dual bounds of its children,

$$\delta(Q) = \min_{(Q,Q') \in A} \{\delta(Q')\}.$$

The incumbent is proven to be optimal if $\delta(P) = c^t \hat{y}$ for the root-MIP P of T, and the branch-and-bound procedure can be terminated.

There has been a lot of research conducted on strategies how to find "good" fractionals to branch on, and on the order in which the tree nodes are explored. We give an overview of such branching rules and node selection strategies as well as the available literature in the sections 2.3.1 and 2.3.2, resp.

Other components that support the branch-and-bound procedure are also briefly introduced in respective sections. *Primal heuristics* are algorithms which aim at quickly finding a feasible solution or improving the incumbent. *Cutting planes* are additional inequalities to strengthen the LP-relaxation of a MIP P which are valid for the convex hull of feasible solutions of P. By *node preprocessing*, we denote algorithms to tighten a MIP formulation, e.g., by inferring the redundancy of certain constraints or by shrinking variable domains.

We now introduce the MIP solver SCIP and give a very general overview of its different components. After introducing the components, we will give computational results which reveal that all five components play an important role for the performance of SCIP.

2.2 SCIP– Solving Constraint Integer Programs

The MIP solver which we used for all computations in this thesis is SCIP (Solving Constraint Integer Programs). SCIP was started in 2002 by Tobias Achterberg in the course of his dissertation [Ach07] (see also [Ach09]). In its current version 3.1, it combines various methods for both linear and nonlinear optimization. The plug-in concept of SCIP allows for an integration of custom solver components such as, e.g., additional primal heuristics, see also Section 2.3.3. SCIP is part of the SCIP Optimization Suite, which is developed at Zuse Institute Berlin and at the universities of Darmstadt and Erlangen. For further information and a list of the numerous people who have contributed to SCIP in the past, we refer to [sci]. Furthermore, we use SOPLEX [sop] in version 2.0 inside of SCIP for solving the node LP-relaxations.

Algorithm 1: generic branching rule

Input : Node P with fractional relaxation solution y_P^{LP} and non-empty set of fractionals $\mathcal{F} \neq \emptyset$
Output : branching variable $j^* \in \mathcal{F}$
1 Compute score s_j for all $j \in \mathcal{F}$;
2 **return** $j^* = \underset{j \in \mathcal{F}}{\text{argmax}}\{s_j\}$;

2.3 Components of SCIP

2.3.1 Branching rules

The decision on which fractional variable to branch is crucial for the success of the branch-and-bound search. As outlined in Algorithm 1, a *branching rule* assigns a score s_j to every fractional $j \in \mathcal{F}$. The branching variable is the one maximizing the score. Every branching rule which we present here is characterized by the score function it uses.

A good branching rule keeps the size of the overall search tree after termination as small as possible at affordable computational cost. More often than not, these two goals are conflicting; while the *least-infeasible*- and *most-infeasible* rules require a computational effort that is linear in the number of fractionals but are considered weak in keeping the tree size small, the *strong branching* rule yields relatively small trees at a high computational cost at every node of the tree. Some of the branching rules presented in this section use the following definition of up- and down-fractionalities:

Definition 2 (Fractionalities)**.** *For a fractional $j \in \mathcal{F}$ in an LP-relaxation solution y^{LP}, we define its* up-fractionality *as*

$$f_j^+ := \lceil (y^{LP})_j \rceil - (y^{LP})_j$$

and its down-fractionality *as*

$$f_j^- := (y^{LP})_j - \lfloor (y^{LP})_j \rfloor = 1 - f_j^+.$$

Note that fractionalities are always positive. The *least-infeasible* and *most-infeasible* rules are based solely on the notion of fractionalities to determine the branching variable.

least-infeasible and *most-infeasible* rules

The *most-infeasible* rule chooses the fractional $j \in \mathcal{F}$ whose fractionality is closest to 0.5 by considering the score

$$s_j = \min\{f_j^+, f_j^-\}.$$

The score of the *least-infeasible* rule prefers variables with an almost integral solution value:

$$s_j = \max\{f_j^+, f_j^-\}.$$

Computational experiments conducted in [AKM04, Ach07] indicate a weak performance for either of these two rules. One of the reason for this poor performance might be the narrow view at a node provided by fractionalities alone. They neither respect the problem structure, nor do they consider objective improvements in the children of a node. Note, however, that Berthold et al. [BGS14] showed in a recent experiment that considering the combined fractionalities of several LP solutions decreases the number of branch-and-bound nodes required by the *most-infeasible* rule by around 50%.

strong branching

The *strong branching* rule focuses on the gain in the objective function. It was first proposed in the context of the Traveling Salesman Problem [ABCC95] and first applied to general MIP within CPLEX [cpl]. The *strong branching* score of a fractional $j \in \mathcal{F}_P$ at a node P with LP relaxation solution value $\bar{c}_P := c^t y_P^{\text{LP}}$ is

$$s_j^{\text{str}} = \max\{\bar{c}_{P_-^j} - \bar{c}_P, \epsilon\} \cdot \max\{\bar{c}_{P_+^j} - \bar{c}_P, \epsilon\}$$

with $\epsilon = 10^{-6}$. Here, P_-^j and P_+^j denote the two subproblems obtained after branching on j. The information is collected through actually solving the relaxations of the children. It chooses the fractional which provides the locally best gain in the dual bound of the children LP relaxations. The use of the product in the above formula attempts to balance the two individual scores for the two branching directions, in order to balance the size of the resulting subtrees. At every node P, a series of $2 \cdot |\mathcal{F}_P|$ child relaxations needs to be solved, which makes *strong branching* the computationally most expensive branching rule presented here. SCIP comes with three branching rules that apply a *strong branching* approach: the first one, which performs look-aheads on the full set of fractionals $\mathcal{F}$, is called *full strong branching*. The *strong branching* rule, however, imposes further restrictions on the size of the candidate set $\mathcal{F}' \subseteq \mathcal{F}$ for which *strong branching* is actually performed. The look-aheads are only applied for at most $|\mathcal{F}'| = 100$ candidates, and stopped whenever the best candidate w.r.t. the *strong branching* score has not changed after 8 consecutive candidate evaluations. A further restriction regards the number of iterations performed for a single *strong branching* subproblem, which are limited to twice the average LP iterations per LP used so far, but at most 500 iterations.

Although *strong branching* is generally attributed to produce the smallest trees among all branching rules on single variables, the computational overhead for solving the high number of additionally required LP relaxations makes *strong branching* alone too expensive, even with the additional limitations discussed above. It is therefore often used in combination with *pseudo-cost branching* as an initial branching rule when sufficient pseudo-cost information is not available yet.

pseudo-cost branching* and *reliability pseudo-cost branching

The *pseudo-costs* of a variable are a typical measure to estimate its impact on the children's lower bounds after branching. Their definition and first use for MIP date back to [BGG+71]. For a node P with LP solution value $\bar{c}_P$ and a fractional variable $j \in \mathcal{F}_P$, let P_- and P_+ be the two child problems of P obtained after branching on j downwards and upwards, resp. Let $\bar{c}_{P_-}$, $\bar{c}_{P_+}$ be the values of the respective child LP relaxations. With the variable fractionalities $f_j^+(P) := \lceil (y_P^{\text{LP}})_j \rceil - (y_P^{\text{LP}})_j$ and $f_j^-(P) := (y_P^{\text{LP}})_j - \lfloor (y_P^{\text{LP}})_j \rfloor$, the *unit gains* in the respective direction are defined as

$$\varsigma_j^-(P) := \frac{\bar{c}_{P_-} - \bar{c}_P}{f_j^-} \quad \text{and} \quad \varsigma_j^+(P) := \frac{\bar{c}_{P_+} - \bar{c}_P}{f_j^+}.$$

Note that the unit gains are well defined because the fractionalities are positive. Besides, the unit gains are always nonnegative.

Definition 3 (Pseudo-costs)**.** *For an integer variable $j \in \mathcal{I}$ of a MIP P, let $\eta_j^-, \eta_j^+ > 0$ denote the number of problems Q for which j was selected as branching variable and the child nodes Q_-, Q_+ have been solved and were feasible, and let σ_j^-, σ_j^+ be the sums of obtained unit gains over all these problems. We define the* pseudo-costs *of j in the respective directions as*

$$\Psi_j^- = \frac{\sigma_j^-}{\eta_j^-} \text{ and } \Psi_j^+ = \frac{\sigma_j^+}{\eta_j^+}. \tag{2.5}$$

If there is a direction for which the number of problems is 0*, we call j* uninitialized *in this direction and set the corresponding pseudo-costs to* 0*.*

The pseudo-costs Ψ_j^- and Ψ_j^+ measure the average unit gain observed after branching on j downwards and upwards, resp. Note that pseudo-costs are subject to change over time. Every time that a child problem is solved and feasible, the pseudo-costs of the corresponding variable are updated. The disadvantage of pseudo-costs is that all variables are uninitialized at the beginning of the search such that pseudo-costs provide no information. The initialization to 0 is not the only possible variant, neither is the choice of averaging the unit gains through the complete history. Alternative suggestions include to take the most recent observed unit gain after branching on a variable as pseudo-costs instead, or the very first observation, see [LS97] for discussion, experiments, and further literature.

If sufficient pseudo-cost information is available, the estimated gain for branching up on j is $\Psi_j^+ \cdot f_j^+(P)$, and the estimated gain after branching down on j is $\Psi_j^- \cdot f_j^-(P)$. The pseudo-cost score of j is obtained via the product score function,

$$s_j^{\text{ps}} = \max\{\Psi_j^- \cdot f_j^-(P), \epsilon\} \cdot \max\{\Psi_j^+ \cdot f_j^+(P), \epsilon\},$$

$\epsilon = 10^{-6}$, which attempts to balance the size of the two subtrees obtained after branching.

The *pseudo-cost branching* rule is an effective replacement of the *strong branching* rule at later stages of the search but lacks information at the beginning. For that reason, some combinations of *pseudo-cost branching* and *strong branching* have been developed, some of which use a *strong branching* initialization on uninitialized variables, and pseudo-costs for every initialized candidate, or *strong branching* at the topmost x levels of the tree, and *pseudo-cost branching* at deeper levels. The best combination introduced so far is based on the notion of *reliability* [AKM04]: Given a reliability parameter $\eta_{\text{rel}} > 0$, the pseudo-costs of a variable are considered unreliable as long as $\min\{\eta_j^-, \eta_j^+\} < \eta_{\text{rel}}$. On unreliable candidates, *strong branching* is performed with the same restrictions on LP iterations and the number of look-aheads without a new best candidate as in *strong branching*. This rule is used by *reliability pseudo-cost branching*.

By the writing of this thesis, η_{rel} is set to 5. Setting $\eta_{\text{rel}} = 1$ resembles *pseudo-cost branching* with *strong branching* initialization. The η_{rel} is dynamically adjusted at every node depending on the proportion of LP iterations during *strong branching* and the total Simplex iterations regularly spent on node evaluation, see [Ach07] for further details.

The use of a reliability parameter η_{rel} is superior to *pseudo-cost branching* supported by *strong branching* at the topmost levels of the tree because it better concentrates *strong branching* on variables with very little branching information. In Section 4.3.3, we present a modification to this notion of reliability which we base on the observed variance in the branching history of the candidate variables.

inference branching

While the *strong branching* and the *pseudo-cost branching* prefer variables with a large impact on the dual bounds of the child nodes, the *inference branching* rule considers the impact on the domain size of the subproblems. Similarly to pseudo-costs, an inference score is kept for every variable based on history information. Let $j \in \mathcal{F}$ be a fractional, for which a total of ϕ_j^- and ϕ_j^+ deductions on other integer variables was observed after branching on j downwards and upwards, resp. Let θ_j^- and θ_j^+ count the numbers of such branchings so far. The inference score s_j of j balances the average number of deductions in both directions

$$s_j^{\text{inf}} = \max\left\{\frac{\phi_j^-}{\theta_j^-}, \epsilon\right\} \cdot \max\left\{\frac{\phi_j^+}{\theta_j^+}, \epsilon\right\},$$

with $\epsilon = 10^{-6}$. Similarly to pseudo-costs, there is no inference history available at the beginning of the search. Initial inference values for binary variables are obtained from the probing presolver, see [Ach07] for further details.

reliability pseudo/inference branching

The default branching rule of SCIP combines the individual scores s_j^{rel} and s_j^{inf} of *reliability pseudo-cost branching* and *inference branching* in a weighted sum. In addition, two scores s_j^{cut} and s_j^{conf} represent the number of times that branching on

j led to infeasible children and information obtained from conflict analysis [Ach07], respectively. Let $\bar{s}^{\text{conf}}$, $\bar{s}^{\text{inf}}$, $\bar{s}^{\text{rel}}$, and $\bar{s}^{\text{cut}}$ denote the average scores over all integer variables. The score used by *reliability pseudo/inference branching* is composed as

$$s_j = \omega^{\text{rel}} \cdot f\left(\frac{s_j^{\text{rel}}}{\bar{s}^{\text{rel}}}\right) + \omega^{\text{inf}} \cdot f\left(\frac{s_j^{\text{inf}}}{\bar{s}^{\text{inf}}}\right) + \omega^{\text{cut}} \cdot f\left(\frac{s_j^{\text{cut}}}{\bar{s}^{\text{cut}}}\right) + \omega^{\text{conf}} \cdot f\left(\frac{s_j^{\text{conf}}}{\bar{s}^{\text{conf}}}\right). \tag{2.6}$$

The function $f(x) = \frac{x}{x+1}$ is used to map all score values to a unified scale, namely the interval $[0, 1)$. By default, SCIP uses weights of $\omega^{\text{rel}} = 1$, $\omega^{\text{inf}} = 10^{-4}$, and $s^{\text{conf}} = s^{\text{cut}} = 10^{-2}$. This choice of score weights makes *reliability pseudo/inference branching* very similar to *reliability pseudo-cost branching*, because the small weights of the remaining score functions rather provide an alternative on problems without an objective function.

In this thesis, we test a modification to the *reliability pseudo/inference branching* weights; instead of using fixed weights, we adjust the weights dynamically based on the actual development observed in the branch-and-bound tree, see Section 4.3.3 for further details.

2.3.2 Node selection strategies

The guidance into a good search tree region is crucial for the performance of a branch-and-bound search. Node selection strategies have to deal with two conflicting goals: While feasible solutions are usually located at deeper levels of the search tree, open nodes with better dual bounds usually lie in shallower regions. By diving deep into the tree to search for feasible solutions, one risks to explore many superfluous nodes, i.e. nodes which could have been pruned with the knowledge of a better incumbent. In this section, we present node selection rules together with a discussion of their strengths.

dfs and *breadth-first* node selection

The *depth-first-search* (*dfs*) strategy tends to stay as deep in the tree as possible by preferring child nodes over siblings. The *breadth-first* node selection rule does the opposite by exploring all open subproblems at the topmost level of the search tree before continuing with subproblems located one level deeper. Both node selection strategies have in common that they do not exploit additional node quality information such as lower bounds.

An advantage of *dfs* consists of the small distance between two processed nodes in the search tree; since the associated subproblems differ only slightly, usually by one reduced variable domain in the case of a child node compared to its parent node, *dfs* requires little adjustments from one processed node to the next (and only few LP iterations) which reduces the overall node processing time. Moreover, *dfs* benefits from the lower memory requirements. In the (most common) case of a 2-way branching scheme, the number of open nodes does not exceed $2d^{\max} + 1$, where $d^{\max}$ denotes the maximum depth of the branch-and-bound tree. Another

advantage of *dfs* is its potential in quickly generating conflict clauses (see, e.g., [Ach07]).

bfs and best-estimate search

The *best-first-search* (*bfs*) selects the dual-bound defining node to be explored next. Unlike *dfs* methods, *bfs* is the node selection rule which takes the minimum number of nodes until a problem is solved to proven optimality. Since the distance between one solved subproblem and the next can be large, the processing costs per node are usually significant higher compared to *dfs* when *bfs* is used.

In practice, the *bfs* node selection rule inside SCIP uses both *bfs* and *dfs* in a hybrid fashion: After a backtrack, a dual-bound defining node P' with $\delta(P') = \delta(P)$ is selected. The node selection rule iteratively chooses child nodes Q of the previously explored node, as long as their dual bounds do not deviate too much from the overall dual bound $\delta(P)$. The deviation of $\delta(Q) - \delta(P)$ is measured w.r.t. the maximum possible distance $c^t\hat{y} - \delta(P)$, whenever there exists an incumbent $\hat{y} \neq \emptyset$, and a backtrack is performed if the deviation exceeds a threshold. This threshold is subject to a user parameter, a default of 25 % is used.

The *best-estimate search* (*bes*) works very similar to *bfs*, except that it uses the *best-estimate* [BGG+71] (see also Section 4.4.1) of a node Q that estimates the objective value of the best solution in the subtree rooted at Q. Like *bfs*, the *best-estimate search* performs limited diving until the current node estimates along the diving path deviate from the global dual bound by more than 25 %. Additionally, *bes* periodically selects a dual-bound defining node instead of the best-estimate node. For more details about the best-estimate to estimate solution values inside sub-trees, see Section 4.4.1.

2.3.3 Primal heuristics

Primal heuristics are incomplete search algorithms that aim at quickly finding feasible solutions. They can be further classified by the strategies they apply into rounding, propagation, diving, and LNS heuristics [FL10, Hen11, Ber06]. For an overview of general MIP primal heuristics, we refer to [Ber06, FL10]. Rounding and propagation heuristics are discussed, e.g., in [Hen11, ABH12].

Diving heuristics explore an auxiliary search tree in a *dfs*-fashion. They fasten the search for primal solutions by providing a branch-and-bound solver with alternatives to the default branching rule. The main advantage is that the dives are only conducted in a virtual tree but the branching decisions of the heuristics do not create subproblems in the branch-and-bound search tree. Nevertheless, future branching decisions of *reliability pseudo/inference branching* might be influenced by pseudo-costs and/or inference information collected during the execution of diving heuristics. For an overview of existing diving heuristics in SCIP, we refer to [Ach07, Ber06].

Large Neighborhood Search (LNS) heuristics solve auxiliary MIPs by making use of reformulations of the MIP P at hand. The reformulations are chosen in such

a way that every solution in the auxiliary search space can be transformed to a solution for P. An auxiliary MIP is also called *sub-MIP* because it is formulated and solved during the solving process of another MIP. Although the reformulations are typically smaller than P w.r.t. the number of variables and constraints, solving them is still $\mathcal{NP}$-hard in general. In order to make good use of LNS heuristics inside of a branch-and-bound solver, strict limits on the solving process of the sub-MIP should be imposed.

The use of primal heuristics inside a complete branch-and-bound solver mostly yields good-quality solutions earlier during search than provided by feasible node solutions alone. However, as soon as an incumbent is optimal, primal heuristics cannot contribute to the search anymore, but may themselves consume a significant amount of time, especially LNS heuristics.

Basically, primal heuristics of SCIP are applied only at certain nodes of the tree corresponding to their `frequency`-parameter: let $f > 0$ be the value of the parameter, and let $d^{\text{off}} \geq 0$ be an offset for the depth, then the primal heuristic is called at every node P whose depth d_P satisfies

1. $d_P \geq d^{\text{off}}$ and
2. $d_P - d^{\text{off}} \equiv 0 \mod f$.

Primal heuristics with a frequency $f = 0$ are only called at depth d^{off}, typically at the root node, and those with $f = -1$ are never applied. Certain heuristics are only applicable in special situations; diving heuristics are only called after the node selection has temporarily finished a dive without reaching a leaf-node. Hence, they are never applied if the node selection rule is *dfs* or *restartdfs*.

The zoo of primal heuristics within SCIP has constantly grown. Instead of describing them all, we restrict ourselves to the LNS heuristic *Proximity search* [FM14] in Section 4.3.2 and the diving heuristic *Distribution diving* [PC11] in Section 4.3.1 as examples of the two aforementioned classes of LNS and diving heuristics, resp. These were recently implemented in SCIP by the author of this thesis. We also describe parameters which are common to all primal heuristics of the specific category, although each of the heuristics may slightly vary in their concrete usage of the parameter. In Chapter 5, we test different values for the parameters in order to improve the phase performance.

2.3.4 Cutting plane separation

A linear inequality $a^t x \leq b$ that is violated by the LP-solution of a node P, but satisfied by all (optimal) solutions of P, is called a *cutting plane*. Cutting planes can be used to strengthen the LP-relaxation at P and its descendants. Cutting planes can be inferred from aggregating multiple constraints of the problem into a new constraint, as it is done, e.g., in strong Chvatal-Gomory cuts [Chv73], or from combinatorial information such as clique inequalities [JP82]. Gomory [Gom58] was among the first to investigate cutting planes for integer programs. Gomory could show that his cutting plane approach could in principle be used to solve every

integer program with rational data in a finite number of steps. For an overview of cutting planes within SCIP, we refer to [Ach07, Wol06].

When using cutting plane separation inside a MIP solver, a generation of too many cutting planes can be disadvantageous for the solving process: every cutting plane is an additional constraint which affects the (re-)solving time of the LP-relaxation at all descendants of a node.

Inside SCIP, cutting plane separators are organized partly as subroutines of the constraint handlers and partly as separate plug-ins, all of which are (by default) only applied at the root node P_0 or completely disabled. A `frequency` parameter, which is similar to the one for primal heuristics, can be used to enable cutting plane separation inside the tree for individual plug-ins. Furthermore, the separation of cutting planes is organized in rounds: During each round of the cut separation loop, every active separation routine produces a set of cutting planes which enter the separation storage. Let y_P^{LP} be the current node solution, and let R denote the set of all generated cutting planes in the current round. The cutting planes $r \in R$ of the form $a_r^t x \leq b_r$ are then selected in nonincreasing order of a score value which is a weighted sum of

- the *cut efficacy* $(a_r^t y_P^{\mathrm{LP}} - b_r)/\|a_r\|_2$,
- the *objective parallelism* $|a_r^t c|/(\|a_r\|_2 \cdot \|c\|_2)$, and
- the *orthogonality* $\min\{1 - |a_r^t a_{r'}|/(\|a_r\|_2 \cdot \|a_{r'}\|_2) : \quad r' \text{ already selected}\}$,

until a maximum number of cuts has been added or the storage becomes empty. In addition, the orthogonality to previously selected cuts must not go beneath a threshold. Both the maximum number of separated cuts, and the minimum orthogonality are subject to user parameters and are different for the root node P_0 and subsequent nodes.

In Section 4.3.3, we describe in more detail the cutting plane separation parameters with which we obtained an improved solver performance regarding the solving time and the number of branch-and-bound nodes when cutting planes are (re)-activated during the third phase. The computational results are presented in 5.1.3.

2.3.5 Node presolving

Node presolving denotes the process of inferring sequences of local domain reductions at the current node of the branch-and-bound search tree. The goal is to shrink the size of the current subproblem as much as possible at affordable computational cost. Inside SCIP, presolving algorithms are applied at two different stages of the MIP solving process. First, they are applied during presolving before the search is started, in which case the deductions hold globally for the problem. Second, they are used locally at nodes within the tree to infer reductions from the branching decisions.

Reductions on variable bounds from linear constraint activity first appeared in Brearly et al. [BMW75]. Savelsbergh [Sav94] proposed probing techniques on

binary variables and constraints, while Andersen and Andersen [AA95] exploited further presolving techniques for linear programming. For an overview of available node presolving algorithms used in SCIP we refer to [Ach07].

We mention node presolving only for the sake of completeness. In the remainder of this thesis, we do not alter the presolving algorithms. The next section shows that presolving algorithms have a very balanced impact on the progress of SCIP w.r.t. both the primal and dual gap.

2.4 Component impact on SCIP performance

The integration and execution of MIP solver components inside of a complete solver can influence the overall solver performance in a positive or negative way. Branching and node selection rules affect the number of branch-and-bound nodes that needs to be explored during the solving process. For primal heuristics, presolving methods, and cutting plane separation, a negative influence is mainly observed on MIPs for which these components incur a large computational overhead without contributing to the solving process. In this thesis, we sometimes decompose the performance of a solver into a primal and a dual part. For every feasible, bounded MIP P for which we know the optimal objective value, we measure the relative distance between the current incumbent and the optimal objective value in terms of a primal gap function. A primal gap of 0 % means that the incumbent is an optimal solution, although this might not be proven so far because the dual bound for P is less than the optimal objective. Similarly, we can use a dual gap function to measure the relative distance between the optimal objective value of P and the currently available dual bound for P. A formal definition of the primal and dual gap functions is given in Section 3.2.

In a first experiment, we categorize components of the MIP solver SCIP by the influence they exhibit on the primal and dual progress of SCIP. This progress is measured in terms of the integral of the average primal and dual gap functions over time. The primal integral was introduced in [ABH12]. A formal definition can be found, as for the primal and dual gap, in Section 3.2. As a test set for

Table 2.1: The shifted geometric means for the primal and dual integrals for every component setting. The means were calculated for the 161 instances of the test set for which an optimal solution is known.

	PrimalIntegral	%	DualIntegral	%
`default`	4243.0	100.0	4259.7	100.0
`random branch`	4533.2	106.8	7534.6	176.9
`no heuristics`	8362.7	197.1	4492.9	105.5
`dfs node sel`	5263.0	124.0	5671.8	133.1
`no presolving`	5340.4	125.9	5667.9	133.1
`no separation`	4361.0	102.8	6731.0	158.0

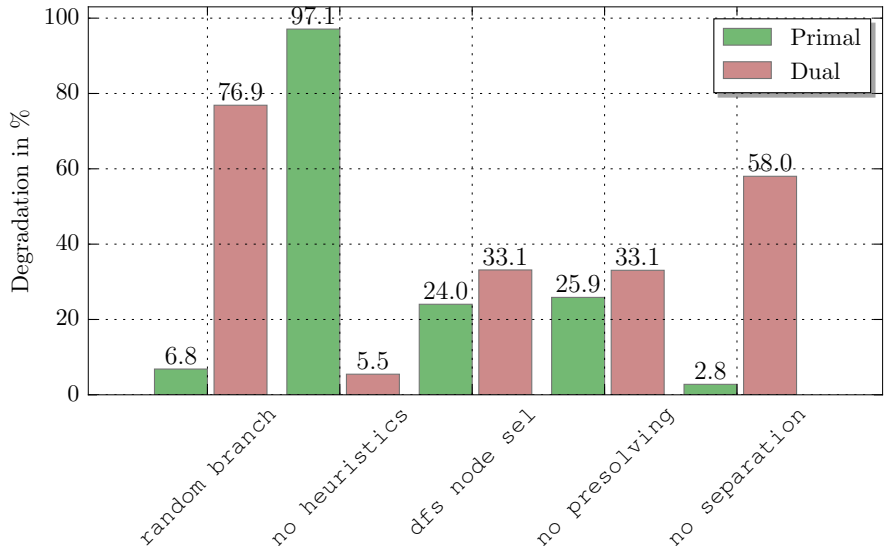


Figure 2.1: Percentage degradation compared to the SCIP performance with default settings regarding the primal and dual integral.

this and all following experiments, we use 168 MIP instances from the three publicly available libraries MIPLIB 3.0 [BCMS98], MIPLIB 2003 [AKM06], and MIPLIB 2010 [KAA+11]. We excluded seven instances which are infeasible or for which there is no optimal solution value known by the writing of this thesis, because the evaluation method depends on the knowledge of an optimal solution value. In contrast to the experiments in Chapter 5, we allowed the jobs to be processed in parallel here, because we are rather interested in a rough categorization.

In order to categorize the component influence, we use settings where we disable one set of components completely, e.g., turn off all of SCIP's primal heuristics. We then compare the obtained primal and dual integrals to the values obtained with SCIP with default settings. For primal heuristics, cutting planes, and preprocessing routines, we have one setting for each component where we disable all algorithms belonging to this component. For the branching rule, we use a randomized branching variable selection instead of the reliability pseudo-cost branching rule. Finally, the estimation-based node selection rule is replaced by a depth first search node selection.

The shifted geometric means of the primal and dual integrals are shown in Table 2.1 for all six obtained settings. A shift of 1000.0 was used, which corresponds to finding no solution for 10 sec. The table shows that the absence of each of the five component sets yields at least a slight degradation of the SCIP performance w.r.t. both the primal and dual integral. Instead of the absolute degradation, we present the relative degradations w.r.t the default setting of SCIP in Figure 2.1.

Replacing the default branching rule of SCIP by a random branching increases the value of the dual integral by about 76.9%. This is the most devastating dual effect among all tested components. The second largest degradation of 58.0% is

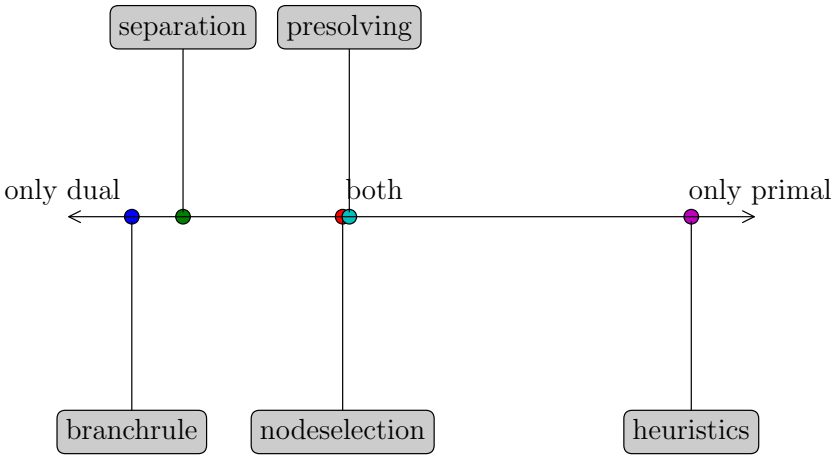


Figure 2.2: The components ordered along their primal and dual influence.

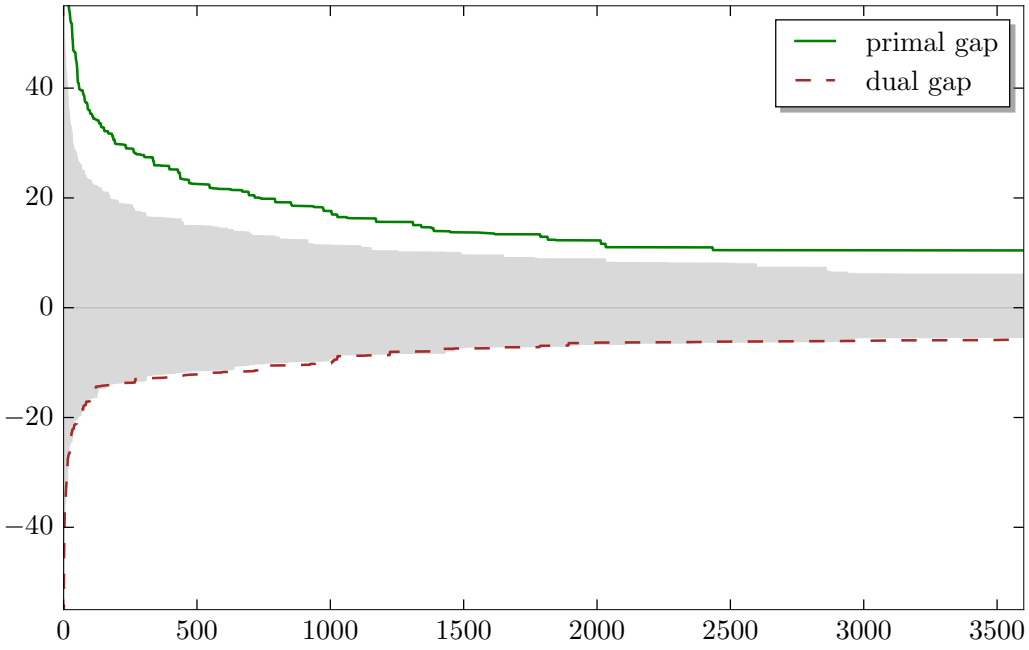
caused by turning off the separation components. The depth-first search node selection rule performs 33.1% worse than the default estimation-based node selection rule. A similar effect can be observed when turning off presolving components which raises the dual integral by 33.1%. A small degradation of the dual integral can even be observed when primal heuristics are turned off.

Primal heuristics, however, exhibit their main potential on the primal side; turning off primal heuristics deteriorates the primal performance by more than 97.1%. A strong influence on the primal integral can also be observed for presolving because the primal integral is increased by an average of 25.9% on unpresolved problems. The third most degrading setting is the absence of a node selection rule, whereas separation and branching can be observed to be less influential. These results indicate that all components except for presolving and node selection mainly influence one of the two aspects considered here. The absence of presolving, however, apparently weakens the remaining components, both primally and dually, in a similar way.

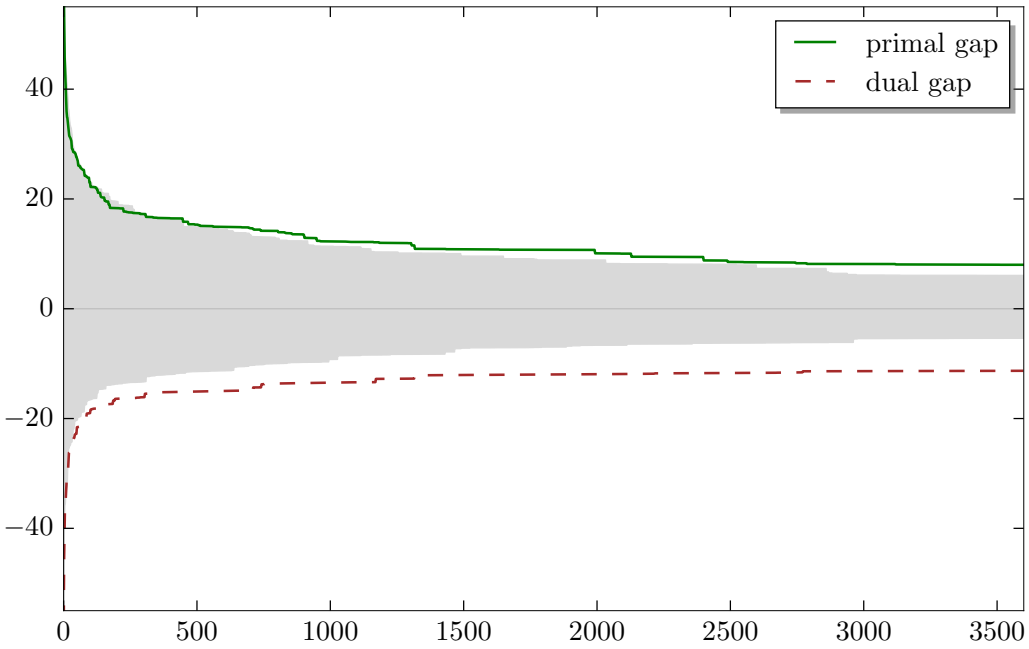
We combine the relative influence on the primal and dual integrals to characterize the strengths of each SCIP component. Figure 2.2 is presented as a conclusion of the described experiment. In this figure, the components are ordered from left to right by the difference of their percentaged primal and dual degradation. Primal heuristics are the rightmost component because their influence is almost exclusively of a primal nature. Presolving and node selection are closer to the center of the axis because they influence the primal and dual progress almost evenly, whereas the branching rule resides at the dual end of the line, followed by cutting plane separation.

In Figure 2.3, we show the progress of both the average primal and dual gaps

over time for two of the non-default settings. The shaded region depicts the average gap obtained by SCIP with default settings, whereas the positive line shows the average primal gap over time, and the negative line shows the average dual gap over time, multiplied by -1, for the specified setting. The primal progress line of the setting with disabled primal heuristics is clearly inferior to the progress obtained by default SCIP. However, as regards the dual gap, the default and the disabled primal heuristics almost align with each other. A different picture is obtained with a random branching rule, which mainly degrades the dual gap reduction. On the primal side, the curve of random branching is beneath the default curve for a short period at about 400 seconds before it is outperformed. The reason for this behavior is the use of *strong branching* as long as pseudo-costs are unreliable at the beginning of the search. The computationally expensive *strong branching* procedure lets SCIP spend more time solving the root node and subsequent "early" nodes, but pays off later in the solving process by reducing the size of the tree.



(a) no primal heuristics



(b) random branching rule

Figure 2.3: Average gaps over time for two of the non-default settings.

Chapter 3

Empirical mathematical programming

Regarding the variety of practical applications of MIP-related research, it is not surprising that almost every publication in this area contains a portion of computational results where the authors compare a newly developed algorithm against existing, well-established methods. A convincing experimental design for testing algorithmic performance involves a lot of choices: the selection of the competing algorithms, a representable collection of randomly generated or available test instances, and a comparison metric.

Since each of these steps has its caveats, the need for an empirical science of algorithm testing has been noted [Hoo94]. This section starts with a recap of general discussions on algorithm testing. Consecutive sections discuss the selection of test sets and metrics for comparison of mathematical programming software. The remainder of this chapter gives an overview of available statistical tests that are suitable for algorithm evaluation.

3.1 Comments on testing algorithms and test set selection

The advent of fast computer hardware enables an algorithm designer to try out and experiment with new ideas at relatively low cost regarding the human time spent on the conduction compared to other areas of science such as, e.g., biology or psychology. One of the basic requirements of a scientific method is its reproducibility. The variety of available hardware components, programming languages, compilers, and, last but not least, programming skills can make it impossible for other researchers to reproduce an interesting published result, see also [KMP13] for a detailed discussion about the reproducibility of a 15-year old result. However, there exist recommendations on reporting experiments in computer science.

Crowder et al. [CDM78] were among the first ones to ask for a thorough experimental design including reproducibility. The authors address the issue of changing technology and require that at least the authors themselves should be able to re-

produce a published experiment. Crowder et al. finish by presenting a check-list of how to report experimental results.

In her survey [McG96], McGeoch collects literature from statistical analysis, exploratory data analysis, and optimization. She presents a case-study for the bin-packing problem and describes how to apply both graphical representations and statistical reasoning to conduct a series of computational experiments. Some typical hazards of simulation programs such as the problematic coping with only limited problem size are also highlighted.

In [Hoo95], Hooker recommends a clear distinction between scientific testing and software benchmarks. He claims that the latter ones are best suited for software developers to measure their progress of accelerating their production code. Researchers, however, should be relieved from what Hooker calls "dual responsibility" of research and software development, and concentrate on effects which are less code-sensitive than the overall running time such as the number of explored branch-and-bound nodes for hypothesis testing.

All aforementioned authors address the problem of selecting a suitable test set for conducting experiments. There exist two main possibilities: on the one hand instances can be generated on a random basis by controlling a limited number of parameters which influence the problem size, density, etc. Such randomly generated instances, however, hardly ever resemble instances from real applications, in which a practitioner is really interested.

The second possibility is the use of an existing MIP library of real-world instances. It is hard to judge in how far a test set of real-world instances is representative of what can be called a general MIP instance. In this thesis, we use a set of instances from the three publicly available libraries MIPLIB 3.0 [BCMS98], MIPLIB 2003 [AKM06], and MIPLIB 2010 [KAA+11]. These libraries were originally compiled as heterogeneous MIP problem set of practical relevance. A public call for instances for the MIPLIB 2010 [KAA+11] led to a library of more than 300 instances, 87 of which are comprised in the official benchmark, which is also part of the test set we use for our computational experiments. For the MIPLIB 2010, special care was taken to filter homogeneous instances. Due to this selection, the resulting, publicly available library represents the current state-of-the-art set for general MIP instances of academic or commercial interest.

There also exists a combination of randomization and real-world data by randomly permuting the order of the variables and constraints of real-world library instances. Although a permuted instance is essentially identical to its unpermuted preimage, the running times of solvers can vary dramatically between the two. This phenomenon is known as *performance variability* [Dan08, KAA+11]. Reasons for performance variability are, e.g., the existence of different optimal LP-bases at a node, the selection of which affects the outcome of primal heuristics and cutting plane separators in an unpredictable way. A second reason lies in the tie-breaking used for selecting, e.g., branching variables. If tie-breaking is imperfect, the selection of a variable for branching might depend on the input order of the fractional candidates, such that the solver eventually creates a different tree.

In this thesis, we do not use permutations for increasing the size of our test

set. Instead, we make use of appropriate statistical tests for distinguishing between potential algorithmic improvements and noise that stems from performance variability.

3.2 Metrics for MIP algorithm evaluation

Experimenting with MIP solving software requires the selection of one or several performance metrics, depending on the aims of the study. Since a main objective of MIP solver development is the reduction of the solving time on a broad number of MIP problems arising in practical applications, the solving time is certainly the most common metric used for computational tests.

The *overall solving time* of a MIP instance is the time in seconds that the solver needs to find an optimal solution and prove its optimality (or prove that no such solution exists). Other time-related metrics include the time until a first or an optimal solution are found.

Despite its practical interest, the use of the solving times for computational experiments has a number of shortcomings which have to be considered: The solving time is very sensitive towards external factors such as the used hardware, the compiler, and the operating system in use. This restricts the reproducibility of computational studies which report the solving time because an identical configuration of hard- and software components might be unavailable. Furthermore, most experimenters impose a time limit to reduce the resource consumption of their studies. If the software is not able to finish within the time limit, its execution is stopped at an intermediate stage, and the time limit is reported as (a lower bound for) the solving time on this particular instance.

The *number of branch-and-bound nodes* accounts for the number of evaluated nodes before termination. Unlike the solving time, the number of nodes should be stable across platforms and system configurations if the algorithm works in a deterministic fashion. A small number of nodes is usually associated with a short solving time, which, in general, does not hold if the time spent on every node is high. Such behavior can be encountered, e.g., for node selection rules exploring the tree in a best-first manner: although such node selection rules outperform simple depth-first-search node selection regarding the number of branch-and-bound nodes, the time spent on each node is considerably higher, cf. Section 5.1.1. For solvers hitting the time limit, it is not clear how to use the node number for fair comparisons. If between two solvers A and B, only A hits the time limit, but evaluates less solving nodes than B, the smaller number of solving nodes is no criterion for preferring A over B. Thus, an evaluation of the solving nodes should be restricted to the subset of instances for which both A *and* B finished within the time limit.

A third possible measure is the *solution quality* at termination. The solution quality is usually reported as a relative distance of the objective value of the best found solution w.r.t. the optimal value or at least a best known value for this particular instance. Public benchmark libraries such as Miplib 2010 [KAA+11]

usually come with such data. The solution quality is not a criterion on infeasible MIPs or instances without an objective function. With a slight abuse of notation, we write $c(t)$ to denote the primal bound, i.e. the incumbent objective at a specific moment in time $t \geq 0$.

Definition 4 (Incumbent and gap function)**.** *Let S be a solver, P be a MIP with nonempty solution space $\mathcal{S}(P)$ and known optimal solution value c^{opt}. Let $\hat{y} : \mathbb{R}^+ \to \mathcal{S}(P) \cup \{\emptyset\}$ be the* incumbent function *of S, which maps every point in time t to the incumbent solution $\hat{y}(t)$ found by S until t, or to $\hat{y}(t) = \emptyset$ if no such solution is found until t. We will call $\gamma : \mathbb{R}^+ \to \mathbb{R}^+$ a* gap function *if γ is monotonously decreasing, and $\gamma(t) = 0$ if and only if $c(t) = c^{opt}$.*

Note that the definition of the gap function implicitly uses the optimal solution value c^{opt}. This general definition of a gap function is realized differently across different MIP solvers. In this thesis, we are interested in a gap function that is bounded by 1 (or 100 %) for normalization purposes. This requirement holds for the gap-function

$$\gamma(t) = \begin{cases} 100\,\%, & \text{if } \hat{y}(t) = \emptyset \text{ or } c(t) \cdot c^{\text{opt}} < 0, \\ 0\,\%, & \text{if } c(t) = c^{\text{opt}}, \\ 100\,\% \cdot \frac{\left|c(t) - c^{\text{opt}}\right|}{\max\{|c(t)|, |c^{\text{opt}}|\}}, & \text{else,} \end{cases} \tag{3.1}$$

which we use as *primal gap function* in this thesis. A dual gap function $\gamma^*(t)$ can be analogously defined using the dual bound $\delta(P, t)$ at time t instead of the incumbent objective (primal bound) function in Equation (3.1). By replacing the optimal solution value c^{opt} in Equation (3.1) by the dual bound $\delta(P, t)$ (here also as function of time), we obtain a *primal-dual gap function*. The MIP solver Cplex [cpl], e.g., reports this primal-dual gap function during search, with the slight difference that no gap is reported as long as $\hat{y}(t) = \emptyset$.

Based on this gap-definition, it is possible to express the entire evolution of the primal gap in the following measure:

Definition 5 (Primal and dual integral [ABH12])**.** *Let P be a MIP with optimal objective value $c^{opt} \in \mathbb{R}$, let S be a solver with primal gap function γ and dual gap function γ^*, and let $T \geq 0$. We call*

$$\Gamma(T) = \int_0^T \gamma(t)\, dt \tag{3.2}$$

the primal integral *of S for P. For $T > 0$, we define $\Gamma(T)/T$ to be the* average primal gap *of S for P. Analogously, a* dual integral *is defined as*

$$\Gamma^*(T) = \int_0^T \gamma^*(t)\, dt. \tag{3.3}$$

The primal integral was first used in [ABH12] to analyze the impact of different classes of primal heuristics on the primal gap. The use of the integral is a combination of two individual measures of the quality of a solution and the time during search when it was found. We already used Γ and Γ^* in Section 2.4 in order to categorize the SCIP components w.r.t. their influence on the primal and dual bound evolution during the search.

From an algorithmic improvement, we expect a reduction in one or several performance metrics in question on an instance set. All performance metrics, the solving time, the number of branch-and-bound nodes, and the primal and dual integral act on scales of several orders of magnitude. Observed solving times during our experiments 5, e.g., ranged from less than a second to 2 hours, the maximum time limit we used. Rather than in absolute differences between two solvers A and B, we are interested in the proportion $\frac{X^A}{X^B}$, where X^A and X^B denote the value of a performance metric X for A and B, respectively. If these factors are constantly smaller than 1 over a whole set of instances, we consider A to be better method than B w.r.t. X. This view makes a geometric mean the method of choice when A and B should be compared over an instance set. A geometric mean still has the disadvantage that it can be strongly influenced by instances which lie at the lower end of the scale; although an absolute difference of 0.2 seconds is only a negligible improvement, it is encountered as a 20 % improvement if $X^A = 0.8$ seconds and $X^B = 1.0$ seconds. Hence, when we compute an average performance metric over a set of instances, we use the *shifted geometric mean* throughout this thesis:

Definition 6 (Shifted geometric mean)**.** *Let $X = (X_i : 1 \leq i \leq q)$ be a series of observations of a performance metric on a set of q MIP instances $(P_i)_{1 \leq i \leq q}$, and let $\tau > 0$. We call*

$$\bar{X}^\tau = \left(\prod_{i=1}^{q} (X_i + \tau) \right)^{\frac{1}{q}} - \tau \tag{3.4}$$

the shifted geometric mean *over (P_i).*

The rationale of using shifted geometric means is a reduction of the influence of extreme instances w.r.t. the performance metric in question, e.g., instances with a very small or very large solving time. The shift value τ is responsible for outliers at the low end of the scale, which may exhibit undesired large relative variations between two solvers A and B on an instance. For the improvement of 0.2 seconds discussed above and using a shifted quotient with $\tau = 10$ seconds, we obtain with $\frac{0.8+10}{1.0+10} \approx 0.98$ a more moderate improvement of only 2 %. We use shifts of 10 seconds, 100 branch-and-bound nodes, and a shift of 1000 for the primal and dual integrals.

Although the influence of outliers regarding the scale of the performance metric is reduced, even shifted geometric means can be influenced by outliers regarding the shifted quotient. The made up series of observations A and B presented in Table 3.1 yield shifted geometric means of $\bar{A}^\tau = 156.36$ and $\bar{B}^\tau = 108.88$, which does not reflect the fact that $\frac{A_i+10}{B_i+10} \approx 0.5$ for 4 of the eight observations. The statistic is dominated by the extreme result at $i = 8$. Though made up, this example is

Table 3.1: An example of a series of eight observations with two solvers A and B, where the shifted geometric mean values with $\tau = 10$ are extremely influenced by the result for $i = 8$.

i	1	2	3	4	5	6	7	8
A_i	100	100	100	100	100	100	100	3000
B_i	200	201	199	202	99	101	102	5

symptomatic for reporting mean values alone. In the next section, we introduce statistical tests which consider rank-statistics of an entire set of observations that come in pairs, as in this example. They can be used to gain additional insights if we observed a real algorithmic improvement.

3.3 Statistical analysis of algorithmic tests

The previous section revised common metrics used in MIP software benchmarks. The aim of this section is to familiarize the reader with statistical hypothesis tests that are applicable for empirical analysis of mathematical programming algorithms. An excellent survey of the applicability of significance tests on a number of case studies from algorithm evaluation is given by Coffin & Saltzman [CS00]. A very illuminating book about applied mathematical statistic is [FBM03].

This section tries to give detailed descriptions of available statistical methods which can be applied to MIP software analysis. The methods are separated into different classes depending on whether they can be applied to continuous data or categorical data. We start with some basic definitions about random variables and probability theory.

3.3.1 Some stochastic preliminaries

This section covers the most basic definitions from probability theory. We only aim to provide the necessary definitions for random variables and distribution functions. A standard text book about probability theory is, e.g., [BT08].

A *measurable space* is a tuple (Ω, Σ) with $\Sigma \subseteq 2^\Omega$ a σ-algebra of a ground set Ω. A map $\mathbb{P} : \Sigma \to [0, 1]$ is called *probability measure* if $\mathbb{P}(\emptyset) = 0$, $\mathbb{P}(\Omega) = 1$, and $\mathbb{P}(\bigcup_{i \in I} E_i) = \sum_{i \in I} \mathbb{P}(E_i)$ for all countable, disjoint collections $(E_i)_{i \in I} \subseteq \Sigma$. A triple $(\Omega, \Sigma, \mathbb{P})$ is called *probability space* if (Ω, Σ) is a measurable space and $\mathbb{P}$ is a probability measure on (Ω, Σ). The elements of Σ are often referred to as *events*, and the *probability* of an event $S \in \Sigma$ is, of course, $\mathbb{P}(S)$.

Two events $S, T \in \Sigma$ are said to be *stochastically independent* if $\mathbb{P}(S \cap T) = \mathbb{P}(S) \cdot \mathbb{P}(T)$. A *random variable* is a map $X : (\Omega, \Sigma, \mathbb{P}) \to (\Omega', \Sigma')$ from a probability space to a measurable space such that $X^{-1}(S') \in \Sigma \ \forall S' \in \Sigma'$. Random variables

can be used to define probability measures on their image space. One can check that the map $\mathbb{P}'_X : \Sigma' \to [0,1]$ such that $\mathbb{P}'_X(S') := \mathbb{P}(X^{-1}(S'))$ is a well-defined probability measure on (Ω', Σ').

For $A \in \Sigma'$, we define the *indicator function* $\mathbb{1}_A(X)$ to yield 1, if $X \in A$, and 0 otherwise. If the image space $\Omega' = \mathbb{R}$ of a random variable are the real numbers, we define the *(cumulative) distribution function* $F_X(t) := \mathbb{P}(X \leq t)$ of X as probability that X takes a value not larger than t, for $t \in \Sigma'$. If there exists a nonnegative function $f_X : \mathbb{R} \to \mathbb{R}_0^+$ such that $F_X(x) = \int_{-\infty}^{x} f_X(t)dt$ for all $x \in \mathbb{R}$, we call f_X *probability density function* of X.

Finally, the *mean* of a real valued random variable X is defined as

$$\mathbb{E}(X) := \begin{cases} \sum\limits_{x \in X(\Omega)} x \cdot \mathbb{P}(X = x), \text{ if } X \text{ is discrete,} \\ \int\limits_{\mathbb{R}} f_X(t)t\,dt, \text{ if } X \text{ has a probability density function } f_X. \end{cases} \tag{3.5}$$

In (3.5), we neglect random variables that are neither discrete nor have a probability density function because they are irrelevant for this thesis. If the mean $\mathbb{E}(X)$ is finite, we define the *variance* of X as

$$\mathbb{V}(X) := \mathbb{E}((X - \mathbb{E}(X))^2). \tag{3.6}$$

3.3.2 Important distributions

In this section, we give a brief overview of the distributions that are relevant for the statistical tests in Sections 3.3.3 and 3.3.4.

The normal distribution

The *normal distribution* is characterized by its probability density function

$$f_{\mu,\sigma^2}(x) = \frac{1}{2\sqrt{\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where μ is the mean of the distribution and σ^2 is its variance. The normal distribution with parameters μ and σ^2 is denoted by $\mathcal{N}(\mu, \sigma^2)$. $\mathcal{N}(0,1)$ is called *standard normal distribution*. For the cumulative distribution function of the standard normal distribution, we write

$$\Phi(x) := F_{\mathcal{N}(0,1)}(x) = \int_{-\infty}^{x} f_{0,1}(t)dt. \tag{3.7}$$

The χ^2-distribution

The sum of squares of n independent, $\mathcal{N}(0,1)$-distributed variables is called χ^2-*distribution* with n degrees of freedom. Its density $f_n(x)$ reads

$$f_n(x) = \frac{1}{2^{\frac{n}{2}}\Gamma(\frac{n}{2})} x^{\frac{n}{2}-1} e^{-\frac{x}{2}},$$

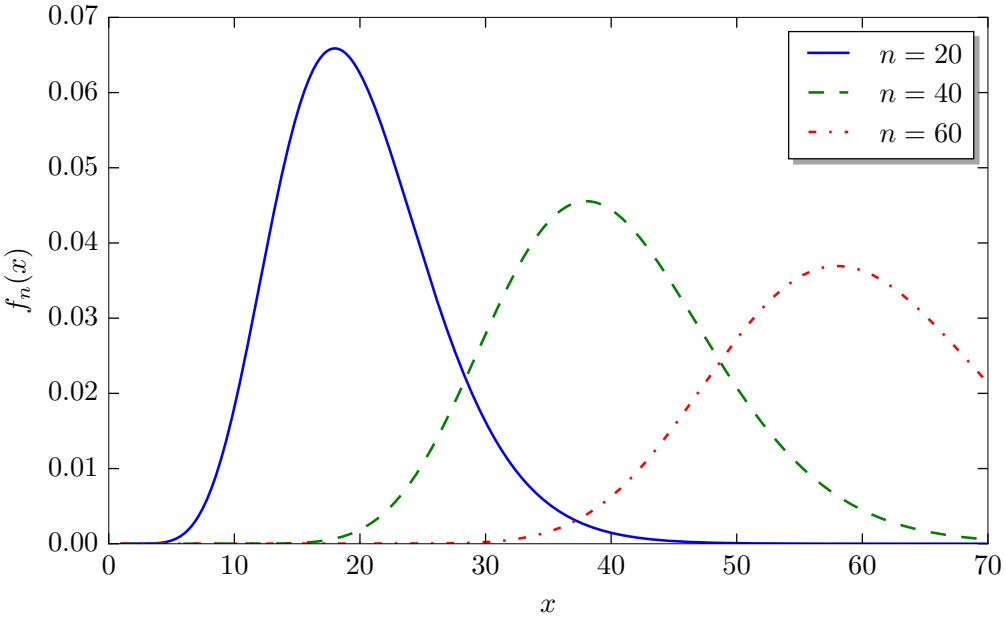


Figure 3.1: probability density of a χ^2-distribution with $n = 20, 40, 60$ degrees of freedom.

with the *gamma-function*

$$\Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt, \quad x > 0.$$

A χ^2-distributed variable X with n degrees of freedom has a mean $\mathbb{E}(X) = n$ and a variance $\mathbb{V}(X) = 2n$. Figure 3.1 depicts the density functions of three χ^2-distributions with different degrees of freedom.

The binomial distribution

The sum X of an independent series of n random trials $X_i \in \{0, 1\}$, $i = 1, \ldots, n$, each with *success probability* $p = \mathbb{P}(X_i = 1)$, takes the value k, $k = 0, \ldots, n$ with probability

$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}. \tag{3.8}$$

We call X *binomially distributed* with parameters n, p and write $X \sim \mathcal{B}(n, p)$.

If $p = \frac{q}{r}$ is rational, $\mathcal{B}(n, p)$ models the probability to draw k white balls with replacement from a bin containing q white balls out of a total of $r \geq q$ balls. If the experiment is conducted without replacement of previously drawn balls, p is not constant anymore (unless $q = r$ or $q = 0$), leading to the hypergeometric distribution.

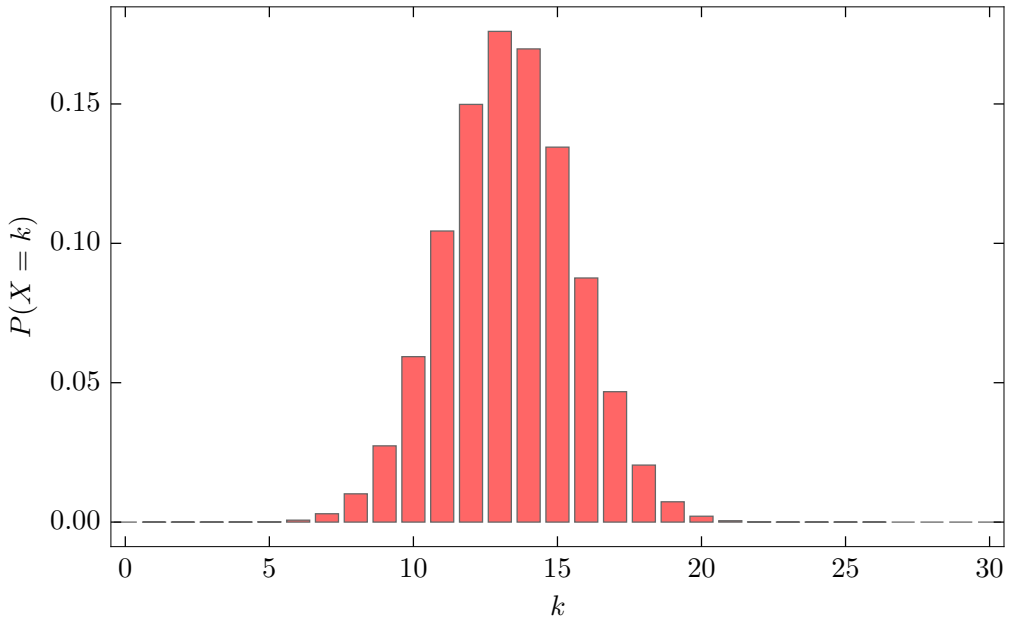


Figure 3.2: Probability mass function of a hypergeometrically distributed variable $X \sim H(90, 40, 30)$.

The hypergeometric distribution

Given a bin containing a total of r black and white balls, with $r - q$ and q the number of black balls and white balls, respectively, the *hypergeometric distribution* describes the probability that a selection of n balls without replacement from the bin contains exactly k white balls for $k = 1, \ldots, n$.

A random variable X is said to be hypergeometrically distributed, $X \sim \mathcal{H}(r, q, n)$, if $0 \leq n \leq q$, and

$$\mathbb{P}(X = k) = \frac{\binom{q}{k} \cdot \binom{r-q}{n-k}}{\binom{r}{n}}, \quad k = 1, \ldots, n.$$

3.3.3 Tests for categorical data

Categorical data arise when the realization of an observation is drawn from a finite set of values without order relation. Examples of such categorical data in MIP benchmarks are events of the form "Solver finished within time limit" and "Solver found a feasible/optimal solution" with binary realizations $X_i \in \{\text{yes}, \text{no}\}$, or comparing n different settings from a set $\{1, \ldots, n\}$.

When dealing with categorical data, one often has to base the evaluation on counting the number of occurrences of each value in the actual observation. Let $(X_1, Y_1), \ldots, (X_n, Y_n)$ be n independent observations, where $X_i \in A$ and $Y_i \in B$ for $i = 1, \ldots, n$, for finite sets A and B.

If both A and B contain only two elements $A = \{a_1, a_2\}$ and $B = \{b_1, b_2\}$, respectively, a series of observations (X_i, Y_i) can be summarized in a 2-by-2 table along with its row and column sums:

$\#\{X_i \backslash Y_i \in \dots\}$	b_1	b_2	
a_1	$n_{a_1 b_1}$	$n_{a_1 b_2}$	$\sum = n_{a_1 \cdot}$
a_2	$n_{a_2 b_1}$	$n_{a_2 b_2}$	$\sum = n_{a_2 \cdot}$
	$\sum = n_{\cdot b_1}$	$\sum = n_{\cdot b_2}$	$\sum = n$

A *contingency table* N is an $|A| \times |B|$-matrix with entries

$$n_{a,b} = \sum_{i=1}^{n} \mathbb{1}_{\{a\}}(X_i) \cdot \mathbb{1}_{\{b\}}(Y_i), \quad a \in A, b \in B.$$

The total number of observations $X_i = a$ for $a \in A$ is given as the sum of elements in row a of N and is denoted as $n_{a\cdot}$. The total number of observations $Y_i = b$ for $b \in B$ is the sum of column entries of column b and denoted by $n_{\cdot b}$.

A MIP-related example of a contingency table is given in Table 5.10 for the common frequency of heuristic phase transition criteria and the optimality of the incumbent when the solver hit the time limit.

The row and column sums $n_{a\cdot}$ and $n_{\cdot b}$ of the contingency table are also referred to as *margins* and play an important role for the following tests. For these tests, we consider a contingency table N as a realization of *random frequencies* ν_{ab} for all $a \in A$, $b \in B$. The exact Fisher test and the χ^2-test for independence measure the probability p to obtain a realization at least as extreme as N under the hypothesis that X and Y are independent. If p is small, we reject the hypothesis and assume instead a dependence between X and Y.

Fisher's test for independence

The Fisher test for independence can be used to test against the hypothesis that X and Y are independent. Under the hypothesis H_0 that X and Y are independent, the frequency ν_{ab} $a \in A$, $b \in B$ is hyper-geometrically distributed, $\nu_{ab} \sim \mathcal{H}(n, n_{a\cdot}, n_{\cdot b})$ (for a proof, see, e.g.,[FBM03]). The likelihood p to observe a value of n_{ab} or a more extreme value as frequency of a and b under the hypothesis H_0 is thus given by

$$p = \sum_{\substack{k=1 \\ \mathbb{P}(\nu_{ab}=k) \leq \mathbb{P}(\nu_{ab}=n_{ab})}}^{n} \mathbb{P}(\nu_{ab} = k), \tag{3.9}$$

and H_0 can be rejected for very small p.

The χ^2-test for independence of X and Y

Given a batch of categorical observations (X_i, Y_i), we can use the deviations of an observed contingency table entry n_{ab} from its expectation $\hat{n}_{a,b}$ under the assumption that X and Y are independent. More precise, if X and Y are independent,

the expected number $\hat{n}_{ab}$ is a function of the two corresponding marginal densities

$$\hat{n}_{ab} = n \cdot \mathbb{P}(X = a, Y = b) = n \cdot \mathbb{P}(X = a) \cdot \mathbb{P}(Y = b) \sim n \cdot \frac{n_{a\cdot}}{n} \cdot \frac{n_{\cdot b}}{n} = \frac{n_{a\cdot} \cdot n_{\cdot b}}{n} \tag{3.10}$$

The actual relative deviations of the table entries from their expected counterparts can be summed up to obtain a test statistic

$$S := \sum_{a \in A, b \in B} \frac{(n_{ab} - \hat{n}_{ab})^2}{\hat{n}_{ab}}. \tag{3.11}$$

Under the assumption of X and Y being independent, S is distributed along a χ^2-distribution with $(|A| - 1) \cdot (|B| - 1)$ degrees of freedom. Hence, a very large value of S speaks against the null hypothesis that X and Y were independent.

In the special case of $|A| = |B| = 2$, the calculation of S simplifies to

$$S = \frac{(n_{a_1 b_1} n_{a_2 b_2} - n_{a_2 b_1} n_{a_1 b_2})^2 \cdot n}{n_{a_1 \cdot} n_{a_2 \cdot} n_{\cdot b_1} n_{\cdot b_2}}, \tag{3.12}$$

and S is distributed along a χ^2-distribution with one degree of freedom [Coh95].

McNemar's test

We consider a 2-by-2 contingency table of $2n$ observations (X_i, Y_i), $i = 1, \ldots, n$, where each $X_i, Y_i \in \{0, 1\}$. In the context of MIP solving, we can think of two solvers X and Y, which are tested on a set of n instances, and a categorical observation is made for both X and Y on each instance i such as whether the time limit was hit or an optimal solution was found. For an empirical software analysis, we are interested in the question whether, e.g., the probability that X solves an instance which Y does not solve is different from the probability that the opposite holds.

The relevant information is found in the anti-diagonal of the contingency table N. Denote by $a := n_{1,0}$ the number of observations for which $X_i = 1$ and $Y_i = 0$, and let $b := n_{0,1}$ denote the other element of the anti-diagonal. Our hypothesis, under which we know a distribution of a and b, is

$$H_0 : \mathbb{P}(X = 0, Y = 1) = \mathbb{P}(X = 1, Y = 0). \tag{3.13}$$

Intuitively, if a and b are close to $(a + b)/2$, H_0 cannot be rejected. Restricting ourselves to the subset of $a + b$ observations that fell in either of the two groups, the entries $N_{0,1}$ and $N_{1,0}$ of the contingency table are binomially distributed with parameters $a + b$ and probability of success $p = 1/2$. A binomial test is therefore applicable, which reports as p-value

$$p = 2 \cdot \sum_{k=1}^{\min\{a,b\}} \binom{a+b}{k} \cdot 2^{-k} \cdot 2^{-(n-k)} = 2^{1-n} \cdot \sum_{k=1}^{\min\{a,b\}} \binom{a+b}{k}.$$

For larger n, the distribution of

$$Z := \frac{(a-(a+b)/2)^2}{(a+b)/2} + \frac{(b-(a+b)/2)^2}{(a+b)/2} = \frac{(a-b)^2}{a+b}$$

can be approximated by a χ^2-distribution with one degree of freedom. The reported p-value in this case is

$$p = 1 - F_{\chi^2}(Z),$$

where $F_{\chi^2}(Z)$ denotes the cumulative density function of a χ^2-distribution with one degree of freedom. This test is called *McNemar test* as a tribute to his author Quinn McNemar, who presented it in 1947 [McN47].

3.3.4 Tests for continuous data

This section gives an overview of available hypothesis tests for continuous data. The most frequent such data in testing mathematical programming software are the individual solving times observed on a set of instances.

Wilcoxon signed rank test for pairs of samples

Let $n \in \mathbb{N}$, and $(X_1, \ldots, X_n)$ and $(Y_1, \ldots, Y_n)$ be two independent and nonnegative observations that come in pairs (X_i, Y_i) for $1 \leq i \leq n$. An example of such paired observations are benchmark results of two MIP solvers X and Y on a set of n MIP instances, such that X_i and Y_i are the observed realizations of one of the discussed performance metrics such as the solving time on the i-th instance.

We noted before that MIP benchmark results such as the total solving time can vary between instances by orders of magnitude, which is why we are more interested in relative improvements or deteriorations X/Y than in absolute improvements.

Let $1 \leq i \leq n$, and let $\tau > 0$ be a shift value as used for the shifted geometric mean (cf. Section 3.2). We define the *logarithmic shifted quotient*

$$Q_i := \log\left(\frac{X_i + \tau}{Y_i + \tau}\right). \tag{3.14}$$

Note that Q_i is well-defined because $(X_i + \tau)/(Y_i + \tau) > 0$ since X_i and Y_i are nonnegative. A logarithmic shifted quotient $Q_i < 0$ is negative if and only if $X_i < Y_i$. The transformed samples remain independent because of the independence of the sample data. Furthermore, we can assume that all Q_i are identically distributed along an unknown distribution $F(x)$ for all $1 \leq i \leq n$. The use of the logarithm is an order-preserving transformation of the shifted quotients with the additional property that an improvement by $1+\epsilon$, $1 > \frac{X_i+\tau}{Y_i+\tau} = \frac{1}{1+\epsilon} > 0$ and a deterioration by ϵ, i.e. $\frac{X_j+\tau}{Y_j+\tau} = 1+\epsilon$, yields the same absolutes for Q_i and Q_j:

$$|Q_i| = \left|\log\left(\frac{X_i+\tau}{Y_i+\tau}\right)\right| = \left|\log\left(\frac{1}{1+\epsilon}\right)\right| = |\log 1 - \log(1+\epsilon)| = \log(1+\epsilon) = |Q_j|.$$

The Wilcoxon signed rank test is a nonparametric alternative to the paired t-test [FBM03] if the underlying distribution cannot be assumed as normal. We test against the hypothesis H_0 that the underlying distributions of X and Y are equal. Under H_0, the distribution of Q_i is centered about the origin. Without loss of generality, we assume that $|Q_1| < |Q_2| < \cdots < |Q_n|$, and that $|Q_1| \neq 0$. Each index i also represents the rank of the i-th sample. In practice, the samples are reduced by filtering all occurrences of $Q_i = 0$, and ties between ranks are solved by assigning the average rank to each of the samples in question.

We compute the *Wilcoxon sum statistics* W_+, W_- as

$$W_+ := \sum_{i=1}^{n} \mathbb{1}_{(0,\infty)}(Q_i) \cdot i \text{ and } W_- := \sum_{i=1}^{n} \mathbb{1}_{(\infty,0)}(Q_i) \cdot i. \tag{3.15}$$

Notably, W_+ and W_- always sum up to $n \cdot (n+1)/2$, the sum of all ranks. Under the hypothesis H_0, W_+ and W_- are identically distributed about a mean $\mu = n \cdot (n+1)/4$ with variance $\sigma^2 := n(n+1)(2n+1)/24$, and can be approximated by means of a normal distribution $z \sim \mathcal{N}(\mu, \sigma^2)$, if n is sufficiently large. Let $W_{\min} := \min\{W_-, W_+\}$ denote the minimum of W_- and W_+, and let $\alpha \in (0,1)$ be a fixed error rate which we use as threshold for rejecting H_0. Let $z_{(1-\alpha/2)}$ denote the $\alpha/2$-quantile of the standard normal distribution, i.e.

$$\mathbb{P}\left(\frac{z-\mu}{\sigma} \geq z_{(1-\alpha/2)}\right) = \frac{\alpha}{2}.$$

We reject H_0 if the condition $W_{\min} \leq \mu - z_{(1-\alpha/2)}\sigma$ is satisfied:

$$\begin{aligned}
&\mathbb{P}\left(W_{\min} \leq z \leq \frac{n(n+1)}{2} - W_{\min}\right) \\
=&\mathbb{P}\left(\frac{W_{\min}-\mu}{\sigma} \leq \frac{z-\mu}{\sigma} \leq \frac{\mu - W_{\min}}{\sigma}\right) \\
\geq&\mathbb{P}\left(-z_{(1-\alpha/2)} \leq \frac{z-\mu}{\sigma} \leq z_{(1-\alpha/2)}\right) = 1-\alpha.
\end{aligned}$$

The inequality holds because of the condition on $W_{\min}$, and because of the symmetry of the standard normal distribution.

We presented a short description of a two-sided Wilcoxon signed rank test. The hypothesis H_0 was that the Q_i are distributed about the origin, whereas the alternative hypothesis locates the median of the underlying distribution of the Q_i values somewhere different from the origin. There also exist two one-sided versions of the Wilcoxon-signed rank test, where one tests H_0: The median of the Q_i is nonnegative against H_1: the median is negative. The second version of the one-sided test is equivalent to the first one if we interchange the roles of X and Y. For the one-sided test, $W_{\min}$ needs to be replaced by W_+, and instead of $z_{1-\alpha/2}$, it already suffices to consider $z_{1-\alpha}$, because very large values for W_+ are in line with the one-sided hypothesis H_0.

Mann-Whitney U test

For $n, m \in \mathbb{N}$, let $(X_1, \ldots, X_n)$ be a batch of independent sample observations, and let independently $(Y_1, \ldots, Y_m)$ be a second batch of independent observations. In contrast to the Wilcoxon signed rank test, the two batches, called *groups*, do not come in pairs (X_i, Y_i), and may even differ in size. Let the X-group and the Y-group be sampled along (unknown) continuous distributions F and G, respectively. In order to test against the hypothesis H_0 that $F = G$, we consider the ranks of the Y-group in the combined sample.

Let all samples be distinct. Without loss of generality, we may assume that $Y_1 < Y_2 < \cdots < Y_m$ and $X_1 < X_2 < \cdots < X_n$. With this ordering, every index i denotes also the rank of X_i in the X-group, and every index j denotes the rank of Y_j in the Y-group. The rank R_{Y_j} of Y_j in the combined groups of X and Y can be written with the help of indicator functions as

$$R_{Y_j} = j + \sum_{i=1}^{n} \mathbb{1}_{(-\infty, Y_j]}(X_i), \tag{3.16}$$

where j is the rank of Y_j in the Y-group, and the sum is increased by 1 by every $X_i < Y_j$. With the help of the ranks (3.16), we define the *$U_{m,n}$-statistic*

$$U_{m,n} := \sum_{j=1}^{m} R_{Y_j} - \frac{m(m+1)}{2}. \tag{3.17}$$

The $U_{m,n}$-statistic is the rank-sum of the Y-group in the combined sample corrected by the contribution of the Y_j themselves. Note that it always holds that $U_{m,n} \in \{0, \ldots, mn\}$. Under the assumptions of H_0, it holds that

$$\mathbb{E}(U_{m,n}) = mn/2, \text{ and } \mathbb{V}(U_{m,n}) = mn(m+n+1)/12. \tag{3.18}$$

For a proof of (3.18), see, e.g., [FBM03]. For sufficiently large m, n, we can approximate the distribution of

$$Z := \frac{U_{m,n} - mn/2}{\sqrt{nm(m+n+1)/12}}$$

by means of a standard normal distribution with cumulative distribution function $\Phi(x)$, see, e.g., [Ser08]. The reported p-value for the Mann-Whitney U test is

$$p = 1 - \Phi(|Z|) + \Phi(-|Z|) \tag{3.19}$$

the probability to observe an even more extreme value of $|Z|$, approximated by a standard normal distribution.

The Mann-Whitney U test is sometimes referred to as Wilcoxon test or U test. It is only applicable for two treatment groups X and Y. The *Kruskal-Wallis test* [KW52] is a nonparametric test to check for significant differences of group medians at the presence of two or more groups.

Chapter 4

A 3-phase-approach for solving MIP

This chapter introduces a tripartion of the solving process into three phases, where the goal of each phase is different from the previous phase. After a formal definition of the phases, we dedicate Section 4.3 to discuss relevant parameters and techniques to achieve the goals of each phase. Furthermore, we introduce newly implemented components in SCIP together with the phase the component was tested for; *uct* node selection [SSR12] and *Distribution diving* [PC11] for the Feasibility phase, *Proximity search* search [FM14] for the Improvement phase, and two modifications to the *reliability pseudo/inference branching* rule for the Proof phase.

In Section 4.4, we introduce three heuristic criteria for a heuristic phase transition between the Improvement phase and the Proof phase; the *best-estimate criterion* compares the minimum best-estimate [BGG+71] over all open subproblems of the tree, which we call the active estimate, and the current incumbent. The *rank-1 criterion* introduces the notion of *rank-1* nodes, which are open subproblems with a best-estimate as least as good as the best previously solved node at the same depth. The third criterion uses a logarithmic model to approximate the progress of the primal bound.

In the remainder of the chapter, we present some of the related work from the literature in Section 4.5. Most of the prior work focused on estimating properties of the search tree, such as the number of branch-and-bound nodes after completion.

4.1 The parameter space of SCIP

The behaviour and performance of MIP solving algorithms drastically depend on the choice of user parameters. Throughout the development of the software, exhaustive computational experiments are dedicated to the search for parameter settings having a good performance on a heterogeneous set of MIP instances, cf. Section 3.1. SCIP currently features more than 1000 parameters falling into three different categories: *real*, *integer*, and *categorical* (cf. Table 4.1). The product space

Table 4.1: Different classes of MIP solver parameters

parameter class	description
real	Real parameters $\rho \in D_\rho$ where D_ρ is an interval subset of $\mathbb{R} \cup \{-\infty, \infty\}$. The maximum time until the solving is aborted (the time limit) is a real parameter with parameter domain $[0, \infty[\cup\{\infty\}$.
integer	Integer parameters ι have interval domains $D_\iota \subseteq \mathbb{Z}$. An LP iteration limit is such an integer parameter.
categorical	a categorical domain is a discrete domain without a reasonable order-relation. Binary parameters fall into this group. Other (crucial) categorical parameters are the choice of a branching or node selection strategy.

of all parameter domains is the *parameter space* of SCIP,

$$\mathcal{S} := \bigotimes_{p \in P} D_p$$

and every element of the parameter space is called *setting*. Naturally, we are interested in a setting that guarantee short running times on our instances. The size of the parameter space, however, makes an enumeration of all possible settings impractical. Parameter-tuning is the task of discovering settings with a good overall performance. Recent years have seen several automated approaches to find good parameter settings [ADL06, HHLBS09]. These methods have shown to be quite beneficial on sets of homogeneous instances, whereas the most common, very heterogeneous MIP benchmark sets such as [KAA+11] remain challenging even for sophisticated automated tuning tools.

4.2 MIP solving phases

The main idea addressed in this thesis is a partition of the solving process of a MIP into a set of phases. Our hypothesis is that an adaptive behavior of a solver w. r. t. the current phase can yield substantial performance improvements compared to a static, global parameter setting. Each of the phases emphasizes a different goal of the solving process, so that it seems natural to pursue these goals with different parameter settings, which are tailored to achieve the phase objective as fast as possible. We suggest a tripartition of the solving process as follows:

Definition 7. *Let P be a feasible MIP with optimal objective value $c^{opt} \in \mathbb{R}$, and let S be a solver for P with incumbent function $\hat{y}$ and dual bound function $\delta(P, \cdot)$.*

We define the three solving phases $\mathcal{P}_1$, $\mathcal{P}_2$, *and* $\mathcal{P}_3$ *of* S *for* P *as follows:*

$$\begin{aligned}
\mathcal{P}_1 &:= \{t \geq 0 \,:\, \hat{y}(t) = \emptyset\}, \\
\mathcal{P}_2 &:= \{t \geq 0 \,:\, \hat{y}(t) \neq \emptyset,\ c(t) > c^{opt}\},\ \textit{and} \\
\mathcal{P}_3 &:= \{t \geq 0 \,:\, c(t) = c^{opt},\ c^{opt} > \delta\,(P,t)\}.
\end{aligned}$$

Clearly, the phases are disjoint. Furthermore, if T is the total time spent by S for solving P to optimality, the phases are a tripartition of the interval $[0, T]$. The central objective during $\mathcal{P}_1$ is a first feasible solution, whose solution quality only plays a minor role. Therefore, we call $\mathcal{P}_1$ Feasibility phase. Feasible solutions are either provided by a node's LP-relaxation solution or by primal heuristics. The first feasible solution plays an important role for the solving process: First, it indicates the feasibility of the model to the user. Second, the bounding procedure of the branch-and-bound algorithm and some propagation routines depend on an incumbent solution. Furthermore, several primal heuristics, such as, e.g., *Proximity search* ([FM14], cf. Section 4.3.2), require a feasible solution as starting point to search for improvements.

After an initial feasible solution was found, the search for an optimal solution is conducted during the Improvement phase. During the Improvement phase, a sequence of IP-feasible solutions with decreasing objective value is constructed until the solver eventually finds an optimal solution.

The remaining time $\mathcal{P}_3$ of the solving process is spent on proving the optimality of the incumbent solution. Such a proof requires the full exploration of the remaining search tree until there are no more open nodes with a dual bound lower than the optimal primal objective value.

The only phase that is always nonempty is the Feasibility phase because we assume that a solving process always starts at time 0 without an incumbent. It is possible that the first feasible solution is also an optimal one, hence $\mathcal{P}_2 = \emptyset$. It can also occur that the best possible dual bound is found before an incumbent with this optimal objective value is constructed, so that no additional computational time is required for the Proof phase, i.e. $\mathcal{P}_3 = \emptyset$, or a combination of the latter two.

For the use of improved phase settings for each phase, we need to determine the point in time when the solving process enters the next phase. The following definition of phase transitions describes the desired moment in time when a phase-based solver should react to by a settings change. Since nonempty phases are right-open interval subsets of $\mathbb{R}_0^+$, we use supremum, whereas the union of all previous phases deals with a probably empty $\mathcal{P}_2$.

Definition 8 (Phase transition)**.** *Let* $\mathcal{P}_1, \mathcal{P}_2, \mathcal{P}_3$ *be the solving phases of a solver* S *for a MIP* P*. For* $i = 1$ *and* 2*, we call* t_i^* *with*

$$t_i^* := \sup \bigcup_{j=1}^{i} \mathcal{P}_j$$

the i-th phase transition.

Note that the solving phases and phase transitions are well-defined even if the solving process itself is not finite or if $\mathcal{P}_2 = \emptyset$. The phase transitions are always positive because $\mathcal{P}_1 \neq \emptyset$. Finally, it holds that $t_1^* \leq t_2^*$ because it it is the supremum of a subset of $\mathcal{P}_1 \cup \mathcal{P}_2$. The values are equal if and only if $\mathcal{P}_2 = \emptyset$, i.e. the first solution is already optimal for P. The recognition of the second phase transition t_2^* after the Improvement phase requires knowledge about the optimality of the current incumbent prior to the termination of the solving process. Note that it is in general not possible to detect this phase transition during the solving process except for trivial cases, where $\mathcal{P}_3 = \emptyset$. When $\mathcal{P}_3 \neq \emptyset$, the decision problem of proving that there exists no solution better than $\hat{y}(t_2^*)$ for our input MIP P remains to be solved. This decision problem is still $\mathcal{NP}$-complete in general.

A bipartion of the solving process has already been suggested in the literature, see [LS97] for an overview and further references. Those phases fold the Improvement phase and Proof phase into one phase, while the proposed strategies solely involve the node selection in use. The three-phase approach used in this thesis gives a more refined control of the solver behaviour. In practice, however, a guess has to be made at which point to assume the incumbent to be optimal. This assumption needs to be based on heuristic criteria that are not guaranteed to decide the optimality of an incumbent correctly. We present possible criteria for this heuristic phase transition in Section 4.4. Parameter considerations for the three solving phases are subject to the following section.

4.3 Computational aspects of the three solving phases

In this section, we discuss techniques to accelerate the achieving of the different phase objectives. The discussed components and parameters are the basis for the conducted computational experiments in Chapter 5. Apart from existing components, some of which were briefly introduced in Section 2.3.1 and 2.3.2, we introduce new components that were added to SCIP for this master thesis. While the *Distribution diving* primal heuristic [PC11] and the *uct* node selection [SSR12] are introduced in the context of the Feasibility phase, the *Proximity search* primal heuristic [FM14] is introduced for the Improvement phase. For the Proof phase, we discuss two modifications to the *reliability pseudo/inference branching* rule [AKM04]. Along with the new components, we discuss common parameters which influence the component performance for the individual phase objectives.

4.3.1 The Feasibility phase

The goal of the Feasibility phase is to find a first feasible solution for a given MIP P. The objective of the solution only plays a minor role during this phase. Feasible solutions are either found by solving the LP-relaxation at a node, or by means of primal heuristics.

Although it can happen that the LP-relaxation solution at the root node is already feasible (and hence optimal for P), feasible solutions are usually located at

deeper levels of the search tree. The search for feasible solutions through branch-and-bound should reflect this by a suitable node selection strategy. We introduced some basic node selection rules in Section 2.3.2. A *dfs* node selection rule is quickest for reaching deep levels of the search tree. In the context of LP-based branch-and-bound search, the *dfs* rule takes advantage of the warmstart capabilities of the dual simplex algorithm. The disadvantage of pure *dfs* is the absence of a backtracking mechanism; the *dfs* node selection might get trapped in infeasible parts of the search, especially if the infeasibility was introduced by early branching decisions. A way to circumvent this is to restart the *dfs* node selection periodically from a shallow open node in the tree. Inside SCIP, such a periodic restart is performed by the *restartdfs* node selection rule, which behaves like standard *dfs*, except for a backtrack to the dual-bound defining node after 100 explored leaf nodes. This can be seen as a variant of hybrid *bfs*/*dfs* node selection with an emphasis on diving. The cost of *restartdfs* is an increased number of LP iterations every time a restart is performed.

Another disadvantage of *dfs*-based node selection is the risk of evaluating superfluous nodes, i.e. nodes with a lower bound greater than the optimal solution value in case it exists. Pruning, however, can only be performed after a feasible solution was found. It is not possible during the Feasibility phase discussed here. Note that furthermore, during the Feasibility phase, it is not decided whether the MIP in question is infeasible. Proving infeasibility requires a full exploration of the search tree, which is performed quickest via *dfs*-based methods.

Another node selection strategy is the new *uct* rule that was presented quite recently [SSR12] in the context of MIP solving, see below for further information. In this thesis, we also experiment with *two-level node selection strategies* in order to overcome the weaknesses of previous approaches. With an initial node selection rule, we let the solver explore a limited number of branch-and-bound nodes. If this limit is reasonably small, one can even use *bfs*, *breadth-first*, or *uct* as initial node selection rule, and use their explorative strength. Since the subproblems at the top part of the tree are similar enough, the increased number of LP-iterations per node are affordable. After the limit has been reached, and the solving process has not been finished, we switch the node selection rule, typically to a rule that prioritizes the exploration of child nodes for making better use of warmstart capabilities such as, e.g., *restartdfs* or *hybrid best-estimate/dfs*.

The choice of the branching rule also influences the phase-1 performance. In our tests, we concentrate on two branching rules: a pure *inference branching* rule and a hybrid *reliability pseudo/inference branching* rule, cf. Section 2.3.1. While the former selects the branching candidate on its inference history observed so far, the latter uses a weighted sum of the pseudo-cost, inference, and cutoff histories of the candidate variables. Recall that *reliability pseudo/inference branching* is the default branching rule used by SCIP. The weights are parameters, and their default values emphasize the pseudo-cost score. The initial absence of reliable pseudo-cost information is overcome by strong-branching look-aheads. Although *reliability pseudo/inference branching* is still the state-of-the-art technique for solving MIP, its computational overhead compared to *inference branching* or a distribution-

based branching rule [PC11] is substantial at early stages of the search, in particular during the Feasibility phase. On feasibility problems with no objective function, e.g., *reliability pseudo/inference branching* decides similarly to *inference branching*, but with the additional LP iterations during strong branching. Using a different branching rule at the beginning of the search can have dramatic effects on the ability of the solver to complete the search because branching decisions at the top of the tree are crucial for keeping the overall tree size small. In practical situations it is often beneficial to explore alternatives to the branching rule. In SCIP, alternative branching rules are periodically applied inside diving heuristics, cf. Section 2.3.3.

For a comparison of the phase-1 performance of different node selection rules and branching strategies, see Section 5.1.1. In the remainder of this section, we introduce *uct* node selection and *Distribution diving*.

uct node selection

The letters *uct* name a node selection rule that has been recently proposed [SSR12] as node selection strategy for MIP branch-and-bound trees. The abbreviation stands for "Upper Confidence Bounds for Trees" and originates from the field of game tree search. The idea of *uct* is to balance exploration and exploitation of the search tree in a single score. Therefore, the notion of *visits* of a node is used; for a node Q of the branch-and-bound search tree, its visits v_Q is the number of explored nodes that have Q as an ancestor. The root node Q_0 has therefore $n-1$ visits for n the number of search tree nodes explored so far. The *uct*-score $U(Q)$ of a non-root node Q with parent Q' reads

$$U(Q) := \frac{\delta(Q_0) - \delta(Q)}{\max\{1, \min\{|\delta(Q_0)|, |\delta(Q)|\}\}} + \epsilon \cdot \frac{v_{Q'}}{v_Q + 1}.$$

The first summand of $U(Q)$ is the relative gap of the node's dual bound to the root dual bound. The normalization is necessary to make the left and right summand comparable. The right side compares the number of visits of the parent node to the visits of the node itself. The weight ϵ is used as a balancing parameter. Since *uct* originates from game tree search, after an open node was explored, a new path is traversed. At every intermediate node, starting at the root node, the child with higher *uct*-score $U(Q)$ is selected, until an open node R is reached. After R has been processed, the visit counters are increased by 1 along the path from the root to R. While *uct* is bound-driven at the beginning of the search and very much resembles *bfs* in this respect, the visit factor gradually gains importance. The authors suggest to use *uct* only at the very beginning of the search.

The implementation of *uct* in SCIP was done by the author of the thesis. As suggested by the description in [SSR12], the user needs to set a concrete limit on the number of nodes evaluated by *uct* before SCIP switches to the node selection method with second highest priority. Note that the tree of SCIP does not support the required forward path data structure to seek paths starting at the root node. Instead, the current implementation of *uct* in SCIP compares the open nodes Q

and R by following their paths until they intersect in a node S. The two children of S, S_- and S_+, are then evaluated as representatives of Q and R, and the open node with the higher representative score is kept as best node. The *uct* implementation in SCIP iterates over all open nodes of the tree and keeps track of the best open node found w.r.t the *uct*-score in every iteration. Since we use *uct* in our experiments only for a very limited number of tree nodes–31, to be precise– this implementation does not cause substantial overhead compared to a runtime optimized implementation.

Distribution diving and *Active constraint diving*

We present the *Distribution diving* heuristic, which was originally proposed as a branching rule in [PC11]. Since the authors aim at finding feasible solutions rather than proving optimality, the approach was implemented as a diving heuristic for SCIP by the author of this thesis.

The score for the heuristic tries to estimate the solution density of a subtree, i.e. the percentage of leaf nodes of the subtree that contain feasible solutions. Let all variable bounds be finite, i.e. let l, u at the current subproblem have finite norm. Now, we formally replace the variables x_j by random variables X_j following a uniform distribution over their domain, with mean value and variance

$$\mathbb{E}(X_j) = \frac{l_j + u_j}{2} \qquad \mathbb{V}(X_j) = \begin{cases} \frac{(u_j - l_j + 1)^2 - 1}{12}, & \text{if } j \in \mathcal{I}, \\ \frac{(u_j - l_j)^2}{12}, & \text{else.} \end{cases} \tag{4.1}$$

Constraint activities become random variables, as well. Assuming the variables to be independent, we obtain for $X = (X_1, \ldots, X_n)^t$ and for every constraint $a_i^t x \leq b_i$ of the subproblem a mean value μ_i and a variance σ_i^2 for the constraint activity by summation:

$$\mu_i = \mathbb{E}(a_i^t X) = \sum_{j=1}^{n} a_{ij} \mathbb{E}(X_j), \text{ and } \sigma_i^2 = \mathbb{V}(a_i^t X) = \sum_{j=1}^{n} a_{ij}^2 \mathbb{V}(X_j). \tag{4.2}$$

In order to estimate the solution density for constraint i, we approximate the probability $\mathbb{P}(a_i^t X \leq b_i)$ by means of a normally distributed variable $z_i \sim \mathcal{N}(\mu_i, \sigma_i^2)$, cf. Section 3.3.2. For a proof of convergence that uses the Lyapunov central limit theorem, we refer to [PQ08]. Figure 4.1 shows the relevant distributions for an example constraint $x + 0.5y + z \leq 10$ that involves three integer variables $x, z \in \{0, \ldots, 5\}$, and $y \in \{0, 10\}$. The activity of the constraint is modeled as a random variable Q with mean value $\mu = 7.5$ and variance $\sigma^2 \approx 8.3$, calculated via (4.2). The top picture shows the probabilities that Q takes any particular value $t \in \{0, 0.5, 1, \ldots, 15\}$. It also shows the probability density function $f_{\mu,\sigma^2}(t)$ of a normally distributed variable $R \sim \mathcal{N}(\mu, \sigma^2)$. The second diagram reveals the close relationship between the cumulative distribution functions of Q and R; the actual probability $\mathbb{P}(Q \leq 10) \approx 0.82$ is approximated well by $\Phi_{\mu,\sigma^2}(10) = \mathbb{P}(R \leq 10) \approx 0.81$.

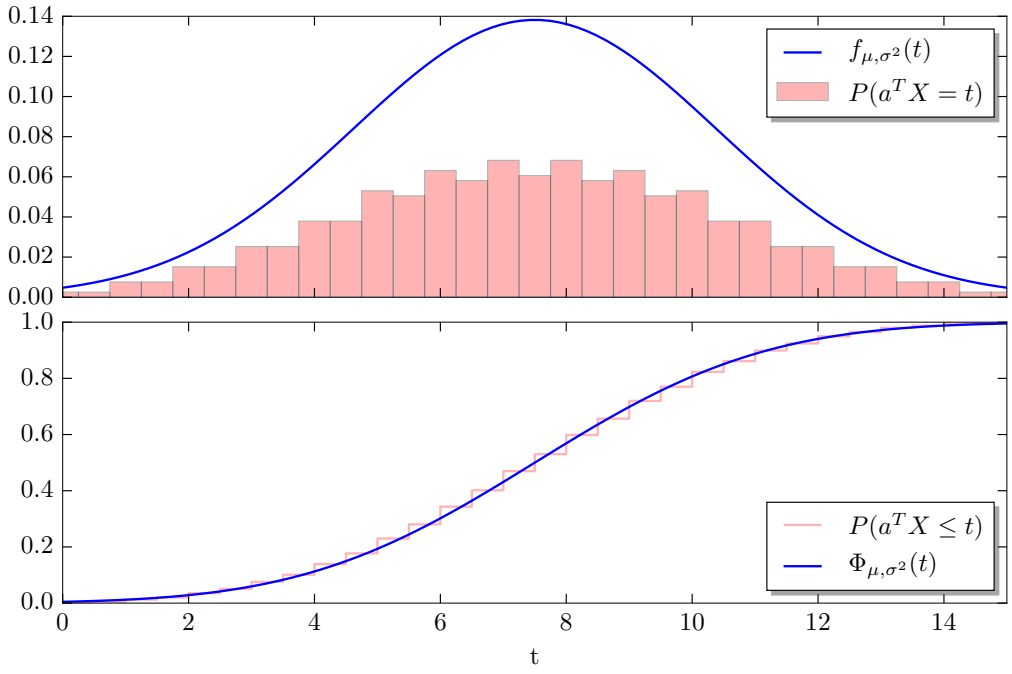


Figure 4.1: The actual probability density of $Q := X + 0.5Y + Z$, where $X, Z \sim \mathcal{U}(\{0, \ldots, 5\})$ and $Y \sim \mathcal{U}(\{0, \ldots, 10\})$, and its approximation by a normal distribution $\mathcal{N}(\mu, \sigma^2)$. The actual solution density is $\mathbb{P}(Q \leq 10) \approx 0.82$, whereas the normal approximation yields 0.81.

If there is a variable $j \in \mathcal{I} \cup \mathcal{C}$ with infinite domain and nonzero coefficient $a_{ij} \neq 0$ for constraint i, we cannot apply the mean and variance calculation (4.1). If j is unbounded in the satisfying direction of i, i.e. $a_{ij} > 0$ and $l_j = -\infty$ or $a_{ij} < 0$ and $u_j = \infty$, j can always be used to "repair" any partial assignment of the other variables of i. We therefore associate a probability of 1 to the satisfaction of i. If j is only unbounded in the violating direction of i, there either exists another variable $j' \neq j$ which is unbounded in the satisfying direction of i, in which case we assume a probability of 1 that we can satisfy i by repairing the activity of i using j'. If, however, there is no such j', we can instead infer a finite bound for j by using the minimum activity of i. In order to account for infinite bounds, we only need to keep counters for the number of infinite bounds in the satisfying direction of i. If the counter is not zero, we assign a probability of 1 as the approximated probability for i. If the counter becomes zero, we use the normal approximation for the calculation of the solution density for i.

Let $\mathbb{P}_i$ the approximated probability to satisfy constraint i at node P, and let $\mathbb{P}_i^+(j)$ and $\mathbb{P}_i^-(j)$ the altered probabilities after branching on variable $j \in \mathcal{F}_P$. Note that all probabilities lie in the unit interval $[0, 1]$. Thus, they provide a normalized comparison between constraints whose activities may otherwise act on different scales. Based on this notion, different scoring functions are possi-

ble. For each of the scoring functions below, we only consider the score for the branching direction upwards (s_j^+). The score s_j^- for branching downwards is defined analogously. A candidate variable $j^* \in \mathcal{F}$ is taken which has the maximum score $s_j = \max\{s_j^-, s_j^+\}$. The maximizing direction determines the child node that should be explored first. We use M_j to denote all constraints with nonzero coefficients for variable j. In total, our SCIP-implementation of *Distribution diving* features five different score functions which use the solution-density probabilities:

1. Branch on the variable with the *highest* probability for any constraint
$$s_j^+ = \max_{i \in M_j}\{\mathbb{P}_i^+(j)\}$$
2. Branch on the variable with the *lowest* probability
$$s_j^+ = \max_{i \in M_j}\{1 - \mathbb{P}_i^+(j)\}$$
3. Branch on the variable with *most violating votes* in one direction
$$s_j^+ = |\{i \in M_j \,:\, \{\mathbb{P}_i^+(j) < \mathbb{P}_i^-(j)\}|$$
4. Branch on the variable with *most satisfying votes* in one direction
$$s_j^+ = |\{i \in M_j \,:\, \{\mathbb{P}_i^+(j) > \mathbb{P}_i^-(j)\}|$$
5. Branch on the variable with the *largest difference* between the current and the child probability
$$s_j^+ = \max_{i \in M_j}\{\mathbb{P}_i - \mathbb{P}_i^+(j)\}$$

While the scores 1–4 are taken from [PC11], we added a difference score 5 as fifth scoring possibility. Chinneck and Pryor [PC11] report good results when using a score with decreasing constraint probabilities from the parent to the child node. The rationale behind this is that forcing variables into a direction that decreases the constraint probability leads to many additional fixings of other variables in the child node through propagation. For our SCIP-implementation of *Distribution diving*, we could not identify a score function inside *Distribution diving* which clearly outperformed the other scores. By default, *Distribution diving* in its current implementation revolves through the score functions, using one score function per dive.

A variant of the score function 3, also mentioned in [PC11], is to consider only the subset of active constraints at the current node. A constraint $i : a_i^t x \leq b_i$ is *active* at a node Q if the LP-solution y_Q^{LP} satisfies i with equality: $a_i^t y_Q^{\text{LP}} = b_i$. If $j \in \mathcal{F}_Q$ has a nonzero coefficient a_{ij} in an active constraint i, branching on j in the violating direction, i.e. upwards for $a_{ij} > 0$ or downwards for $a_{ij} < 0$, renders y_Q^{LP} infeasible for i in the child nodes and thus increases the chance to shrink other variable domains further through domain propagation. The *Active constraint*

diving heuristic inside SCIP makes use of this observation for its branching score. It prioritizes variables and branching directions with a large violating impact on the active constraints at the current node.

Other diving heuristics inside SCIP employ scores based on, e.g., the fractionality of the candidates, the number of up- and down-locks, or the variable pseudocosts, see [Ach07] for more details. All diving heuristics of SCIP employ a limit of the number of LP iterations, which is calculated in proportion to the total number κ_{SCIP} of LP iterations that SCIP needed so far. Let h be a diving heuristic, let κ_h be the number of LP iterations that h consumed so far, and let y_h (y_h^*) denote the number of (incumbent) solutions that h found in the past. Let c_h be the number of times the heuristic has been previously called. By using a quotient $\kappa_h^{\text{quot}} \geq 0$ and an offset $\kappa_h^{\text{off}} \geq 0$, The new iteration limit $\kappa_h^{\max}$ is then calculated as

$$\kappa_h^{\max} = \frac{(10y_h^* + y_h) + 1}{c_h + 1} \cdot \kappa_h^{\text{quot}} \cdot \kappa_{\text{SCIP}} + \kappa_h^{\text{off}} - \kappa_h. \tag{4.3}$$

The heuristic is not executed if $\kappa_h^{\max} \leq 0$, i.e. if it already consumed too many iterations in the past, compared to its success in finding solutions. Otherwise, it is allowed to perform $\max\{\kappa_h^{\max}, 10000\}$ iterations. SCIP with default settings uses $\kappa_h^{\text{quot}} = 0.05$ and an offset of $\kappa_h^{\text{off}} = 1000$ for all diving heuristics.

In an aggressive setting, we tested increased values for the parameters κ_h^{quot} and κ_h^{quot}, together with a more frequent execution schedule by using a frequency of 1 for each diving heuristic. Computational results in Section 5.1.1 indicate that the phase-1 running time performance of SCIP is slightly improved through this aggressive diving schedule.

4.3.2 The Improvement phase

The experiment in Section 2.4 has revealed that primal heuristics have the highest impact on the primal integral among the different components. Hence, we focus on adjustments of the execution strategy of primal heuristics within SCIP for improving the performance during the Improvement phase. Apart from diving heuristics, which we discussed in the previous section, Large Neighboorhood Search heuristics (cf. Section 2.3.3) are another subclass of primal heuristics which are particularly powerful for improving the incumbent solution. However, LNS-heuristics are the computationally most expensive primal heuristics inside SCIP because they solve a MIP themselves at every execution. Both the execution frequency of LNS heuristics during the search and the limitations to the solving process of the sub-MIP have to be carefully constrained. We discuss some important parameters which are common to all LNS-heuristics in the following introduction of the *Proximity search* heuristic, which was introduced in [FM14] and implemented in SCIP by the author of the thesis.

The *Proximity search* heuristic [FM14]

Proximity search is a Large Neighborhood Search heuristic which solves a sub-MIP with a different objective function. For the reformulated sub-MIP inside *Proximity*

search, the objective c of the original MIP P is replaced by a distance function between a solution x and the current incumbent $\hat{y}$ of P. An objective cutoff is given as a hard constraint instead. For the distance function $\|\cdot\|$, we only consider the binary variables $\mathcal{B}$ of the problem,

$$\mathcal{B} = \{j \in \mathcal{I} : \ l_j = 0, u_j = 1\}.$$

The distance function between x and $\hat{y}$ is the Hamming-distance between the two:

$$\|x - \hat{y}\| = \sum_{j \in \mathcal{B}, \hat{y}_j = 0} x_j + \sum_{j \in \mathcal{B}, \hat{y}_j = 1} 1 - x_j.$$

The distance between values on integer variables $\mathcal{I} \setminus \mathcal{B}$ is not used for practical reasons because the linear formulation would require auxiliary variables. If $\mathcal{B} = \emptyset$, the heuristic is not applied.

Let $\theta > 0$ be an objective cutoff. The sub-MIP is formulated as

$$\begin{array}{lrll} \min & \|x - \hat{y}\| & & \\ \text{s.t.} & Ax & \leq b & \\ & c^t x & \leq c^t \hat{y} - \theta & \qquad (\textit{Proximity search} \text{ sub-MIP}) \\ & l \leq x & \leq u & \\ & x_j & \in \mathbb{Z} \quad \forall j \in \mathcal{I} & \end{array}$$

Clearly, every feasible solution for (*Proximity search* sub-MIP) is feasible for P. Furthermore, the use of the objective cutoff makes every feasible solution a new incumbent for P. The *Local branching* heuristic [FL03], which is also available in SCIP, uses the Hamming-distance as a constraint; The solution space is reduced to solutions x with $\|x - \hat{y}\| \leq k$ for a suitable $k > 0$.

For determining the objective cutoff θ, our implementation uses a fixed fraction $p \in [0, 1]$ by which *Proximity search* should at least improve the current incumbent:

$$\theta = p \cdot (c^t \hat{y} - \delta\,(P))$$

The parameter p is called `minimprove` and also used inside other LNS heuristics. Its default value is $p = 0.01$, i.e. a 1 %-improvement. The solving process of the sub-MIP is restricted by delimiting the number of nodes and LP iterations allowed in the sub-MIP. Let n be the number of branch-and-bound nodes explored so far.

The maximum number of nodes n_{proxi} which *Proximity search* is allowed to consume is calculated as

$$n_{\text{proxi}} = \min\{q_{\text{proxi}} \cdot n + n^{\text{off}}_{\text{proxi}} - n^{\text{used}}_{\text{proxi}}, n^{\text{max}}_{\text{proxi}}\},$$

where each of q_{proxi}, $n^{\text{off}}_{\text{proxi}}$, and $n^{\text{max}}_{\text{proxi}}$ are subject to the user parameters `nodesquot`, `nodesoff`, and `maxnodes`, respectively. Similarly, the number of LP-iterations κ_{proxi} for the *Proximity search* sub-MIP is bounded by 20 % of the number of LP iterations spent on the original root node LP-relaxation of P. On the one hand, κ_{proxi} is used for the root node of the sub-MIP, and on the other hand, the evaluation of subsequent nodes is restricted further to take at most $\kappa_{\text{proxi}}/n_{\text{proxi}}$ LP

iterations. The rationale is to abort the search very early if the sub-MIP is not substantially easier to solve than the original one. Furthermore, all LNS-heuristics of SCIP come with a parameter to specify the minimum number of nodes between a new incumbent was found, and the execution of the heuristic with this new incumbent.

For our Improvement phase experiments, we compared the normal heuristic execution strategy to more aggressive strategies, which involved either all heuristics or only affected the execution of LNS-heuristics.

4.3.3 The Proof phase

The Proof phase begins when the solver has found an optimal solution during search. Optimality then needs to be proven by traversing the remaining search tree. The use of the simplex-algorithm with its warm-start capabilities makes a *dfs* node selection the method of choice in this scenario. Besides, primal heuristics cannot further contribute to the solution process during the Proof phase. It is therefore consequent to turn off primal heuristics completely in order to further reduce the node processing time.

Cutting plane separation has been shown to be effective for reducing the dual integral in Section 2.4. We test a setting where we reactivate cutting plane separation when we enter the Proof phase for further reducing the tree size.

The largest impact on the progress of the dual bound, however, was observed for the branching rule. In the following, we discuss two modifications to the *reliability pseudo/inference branching* branching rule which can be beneficial for tree size reduction.

The variance of pseudo-costs

Pseudo-costs ([BGG+71], see also Definition 3) become more reliable over time when branching information for the variables is observed. In order to grant good branching decisions at early stages of the search, the branching variable selection is performed by strong branching and pseudo-costs are updated based on the gathered cost information. Since strong branching is computationally expensive, one is interested in restricting strong branching calls only on variables for which not enough pseudo-cost information has been gathered so far. Achterberg et al. [AKM04] defined the current state-of-the-art scheme to decide whether to apply strong branching by introducing *reliability pseudo-cost branching*. For a *reliability parameter* $\eta_{\text{rel}} > 0$, strong branching is performed only on variables $j \in \mathcal{F}$ for which the number of branching observations η_j^+, η_j^- in at least one branching direction is smaller than η_{rel}. Empirical results [AKM04] indicate that reliability pseudo-cost branching is superior to pure strong branching, which is computationally too expensive, and pure pseudo-cost branching, which suffers from unreliable information at the beginning of the search.

The drawback of reliability pseudo-cost branching is that one fixed parameter η_{rel} is supposed to account for the reliability of all variables of the problem

equally well. Intuitively, it seems desirable to have a more individual look at the pseudo-cost information of every variable and to continue strong branching on those candidates whose pseudo-cost estimates fail to converge. Since the variable pseudo-costs are a sample mean, we might also want to use the sample variance for a refined reliability criterion, provided we have observed at least 2 gain samples.

Definition 9. *Let $X_1, \ldots, X_n$ be independent, identically distributed samples. The* corrected sample variance *about the sample mean $\bar{X}$ is given by*

$$s^2 = \frac{1}{n-1} \sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2 . \tag{4.4}$$

The corrected sample variance is an unbiased estimate of the variance of the underlying distribution of the X_i. Equation (4.4) can be rewritten to allow for constant-time updates of s^2 every time a new sample X is observed:

$$\begin{aligned}
(n-1) \cdot s^2 &= \sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2 \\
&= \sum_{i=1}^{n} \left(X_i^2 - 2X_i\bar{X} + \bar{X}^2\right) \\
&= \sum_{i=1}^{n} X_i^2 - 2\bar{X} \sum_{i=1}^{n} X_i + n\bar{X}^2 \\
&= \sum_{i=1}^{n} X_i^2 - n\bar{X}^2 \\
&= \sum_{i=1}^{n} X_i^2 - \frac{1}{n} \left(\sum_{i=1}^{n} X_i \right)^2
\end{aligned}$$

With increasing n, we can expect $\bar{X}$ to approach the mean of the distribution from the law of large numbers. Under the assumption that the samples $X_1, \ldots, X_n$ are drawn from a normal distribution with unknown mean μ and variance σ^2, the random variable

$$T = \frac{\bar{X} - \mu}{s/\sqrt{n}}$$

is distributed along a Student's t-distribution with $n-1$ degrees of freedom [FBM03]. This relation can be used to construct confidence intervals for the true value of μ for any fixed error-rate $0 < \alpha < 1$: with the help of the α-percentile $t_{\alpha,n-1} > 0$ such that

$$1 - \alpha = \mathbb{P}\left(-t_{\alpha,n-1} \leq T \leq t_{\alpha,n-1}\right), \tag{4.5}$$

the confidence interval C which contains the true value of μ with a probability of $1 - \alpha$ is

$$C = \left[\bar{X} - t_{\alpha,n-1}\frac{s}{\sqrt{n}}, \bar{X} + t_{\alpha,n-1}\frac{s}{\sqrt{n}}\right]. \tag{4.6}$$

Moreover, we can estimate the relative error of our estimation as

$$\epsilon^{\text{rel}} = t_{\alpha,n-1} \cdot \frac{s}{\sqrt{n}|\bar{X}|}. \tag{4.7}$$

whenever $|\bar{X}| > 0$.

Applied to pseudo-costs, we determine the relative error for the pseudo-costs associated with every variable. Whenever a new unit gain for for variable $j \in \mathcal{F}$ in the upwards branching direction at a node P was observed, we increase the counter η_j^+ by 1, update the sum of unit gains $\sigma_j^+ = \sigma_j^+ + \varsigma_j^+(P)$ and, in addition, keep track of the sum of squared unit gains $(\varsigma_j^+(P))^2$. This enables us to calculate the sample variance $\left(s_j^+\right)^2$ whenever $\eta^+ \geq 2$. At a node Q, we calculate the relative error ϵ_j^+ of the current pseudo-costs Ψ_j^+ as

$$\epsilon_j^+ = 1.96 \cdot \frac{s_j^+}{\sqrt{\eta_j^+}\Psi_j^+}. \tag{4.8}$$

In (4.8), we substitute $t_{\alpha,n-1}$ from (4.7) by the constant 1.96, which represents the limit α-percentile $\lim_{\eta_j^+ \to \infty} t_{\alpha,\eta_j^+-1}$ for $\alpha = 0.05$. Thus, we slightly underestimate the relative error of the pseudo-cost at a confidence level of 95 %. Recall that pseudo-costs are always non-negative. Hence, we can omit the absolute in the denominator of (4.7). Furthermore, if the upwards pseudo-costs of j are equal to 0, this also holds for the sample variance $\left(s_j^+\right)^2$. We therefore set the relative error to 0 in this case.

Similarly to the reliability threshold η_{rel}, we use a confidence-threshold η_{conf} which we set to 0.5. If the relative error in any of the directions $\max\{\epsilon_j^+, \epsilon_j^-\}$ exceeds η_{conf}, the variable is considered unreliable, and strong branching is performed again.

Adjusting score weights

In Section 2, we introduced the individual score weights of the *reliability pseudo/inference branching* branching rule in SCIP. By default, they prioritize the pseudo-costs of the variables for a branching decision. The other available scores for the variable conflicts as well as the inference and cutoff history of a variable have small weights, which makes them a tie-breaker if two candidate variables have the same pseudo-cost score, in particular on problems without an objective. Even on problems with an objective function and varying pseudo-costs between the candidates, it is not clear a priori that the default weight configuration is indeed the best available. In order to adapt the configuration to the problem at hand, we keep track of the number of different leaves of the branch-and-bound tree of the following kind:

- n_{obj} the number of leave nodes which could be pruned because their dual bound exceeded the current incumbent, and

- n_{cut} the number of leave nodes which were detected to be infeasible.

Based on the *leaf ratio* $q = (n_{\text{cut}}+1)/(n_{\text{obj}}+1)$, we adjust the weights for the cutoff- and conflict scores of *reliability pseudo/inference branching* by $\omega^{\text{cut}} \leftarrow \omega^{\text{cut}} \cdot q$, $\omega^{\text{conf}} \leftarrow \omega^{\text{conf}} \cdot q$. If $n_{\text{cut}} \gg n_{\text{obj}}$, more emphasis is put on selecting variables which have a high cutoff- and/or conflict-score, while the influence of pseudo-costs on the selection of the branching variable gets reduced. On the contrary, if the terminal states are dominated by nodes pruned because of their dual bound, we decrease the influence of the conflict and cutoff weights on the scoring function further. Note that because of $q > 0$, we do not decrease a weight to 0.

The counters do not account for nodes which have been explored during strong-branching. We only perform this adjustment once when entering the Proof phase. Here, we assume that by the time we reach the Proof phase, we have already explored a portion of the tree with sufficiently many leaves falling into the above categories. In practice, however, it might be beneficial to adjust the score weights independently of the phase, especially, if the second phase transition happens early during the search before a good leaf ratio has been determined.

4.4 Heuristic phase transition criteria

Because of the practical impossibility to detect t_2^* exactly before the solving process finishes, we present in this section criteria that we use to determine a *heuristic phase transition* for our phase-based solver. By the time a criterion is met, which we denote by t_2^{crit}, we assume that the current incumbent is optimal and let the solver react on this assumption by switching to settings for the Proof phase. We use the term "heuristic phase transition" in order to emphasize that there is no guarantee in general for the criterion-based assumption of optimality of the incumbent $\hat{y}(t_2^{\text{crit}})$ to hold. In practice, the criteria possibly under- or overestimate the true phase transition t_2^*. A phase-based solver that uses different settings after the heuristic phase transition remains exact; the use of different settings based on the heuristic phase transition might only influence the performance of the solver to finish the solving process.

We present three criteria, which establish properties of the tree and the solving process. This work has not been proposed elsewhere as to the author's knowledge.

4.4.1 The best-estimate criterion

Pseudo-costs (cf. Definition 3) are an estimate for the lower bound gain after branching on a variable. After sufficient pseudo-cost information is available, pseudo-cost estimates replace the computationally expensive procedure of *strong branching* in SCIP. Recall that the estimated gain for branching up on j is $\Psi_j^+ \cdot f_j^+(P)$, and the estimated gain after branching down on j is $\Psi_j^- \cdot f_j^-(P)$. Apart from their use in the selection of the best candidate for branching, pseudo-costs can also be applied to estimate the best solution attainable from a node P.

Whenever $\mathcal{F}_P \neq \emptyset$, we define

$$\Psi_j^*(P) := \min\{\Psi_j^- \cdot f_j^-(P), \Psi_j^+ \cdot f_j^+(P)\} \tag{4.9}$$

as the *estimated minimum cost* to make $j \in \mathcal{F}_P$ integer.

Definition 10 (Best-estimate [BGG+71])**.** *The* best-estimate *for a MIP P for which the LP-relaxation has been solved is given by the formula*

$$\hat{c}_P = c^t y_P^{LP} + \sum_{j \in \mathcal{F}_P} \Psi_j^*(P). \tag{4.10}$$

The best-estimate is an estimate of the optimal value of P. Note that the best-estimate (4.10) is exact for all nodes P for which $\mathcal{F}_P = \emptyset$. The rationale behind (4.10) is to independently consider the cost of making every single $j \in \mathcal{F}_P$ integer. The best-estimate does not account for a possible interplay between variables $j, k \in \mathcal{F}_P$ to influence the integrality of each other. This observation makes $\hat{c}_P$ likely to overestimate the actual objective value of the best attainable solution from the subtree rooted at P. Another important aspect concerns a possible degeneracy of the LP-relaxation: Whenever there exist different optima to the node LP-relaxation, they might lead to different estimates.

In order to determine an estimate of an open node Q, for which the LP-relaxation has not been solved, we infer an estimate from the parent of Q. Let Q be the child of another node P after branching upwards on $j \in \mathcal{F}_P$. An initial estimate of Q can be calculated via

$$\hat{c}_Q = \hat{c}_P - \Psi_j^*(P) + \Psi_j^+ \cdot f_j^+(P).$$

It is noteworthy that the node estimates in SCIP are not updated dynamically together with the pseudo-costs due to running-time considerations, i.e. Q keeps its initial estimate during the entire time it is in $\mathcal{Q}$, although more recent pseudo-cost information on the branching variable j is available.

We call the minimum best-estimate among the set of open nodes $\mathcal{Q}$,

$$\hat{c}_{\mathcal{Q}}^{\min} = \min\{\hat{c}_Q \,:\, Q \in \mathcal{Q}\} \tag{4.11}$$

the *active estimate*. As first heuristic phase transition, we compare the incumbent solution with the active estimate and assume the current incumbent to be optimal if the active estimate is not better than the incumbent objective anymore:

$$t_2^{\text{estim}} := \min\left\{t \geq t_1^* \,:\, c(t) \leq \hat{c}_{\mathcal{Q}}^{\min}(t)\right\} \tag{4.12}$$

Note that by requiring $t \geq t_1^*$, we make sure that there is indeed an incumbent $\hat{y}(t) \neq \emptyset$.

4.4.2 The rank-1 criterion

With an increasing number of explored branch-and-bound nodes, it becomes less and less likely to encounter a solution better than the current incumbent. On the other hand, every unprocessed node $Q \in \mathcal{Q}$ has the potential to contain a better solution in its subtree. The second criterion for heuristic phase transition is based on the following definition of node ranks. The rank rg_Q represents the minimum position of node Q in any list $\mathcal{P}^{d_Q}$ that contains all nodes at depth d_Q in nondecreasing order of their optimal solution.

Definition 11. *Let T be the search tree after termination, and define c_Q^{opt} be the optimal objective value for node $Q \in T$ (or ∞ if there is no feasible solution for Q). We define the* rank *rg_Q of Q as*

$$rg_Q := \left|\{Q' \in T : d_{Q'} = d_Q, c_{Q'}^{opt} < c_Q^{opt}\}\right| + 1. \tag{4.13}$$

The root node P_0 trivially has a rank of 1, because it is the only node at depth 0. Indeed, if T were known in advance, the rank is defined in such a way that an optimal solution can be found by following a path of nodes of rank 1, starting at the root node.

Although such information is practically unavailable, we still pursue the idea of node ranks. If the solving process has not uncovered an optimal solution yet, there exists a rank-1 node among the open nodes $\mathcal{Q}$. Note, however, that there may even be rank-1 nodes although the current incumbent is already optimal. As for the best-estimate criterion, we use the best-estimate (cf. Definition 10) to circumvent the absence of true knowledge about best solutions in the unexplored subtrees. For every depth d, we keep track of the minimum node estimate at this particular depth so far, including feasible nodes, i.e. subproblems with feasible LP-relaxation solutions. We impose a partial order relation on the nodes by

$$Q' < Q \quad \Leftrightarrow \quad Q' \text{ was processed before } Q, \quad \forall Q', Q \in T, Q' \neq Q.$$

With this partial order relation, we define the set of rank-1 nodes

$$\mathcal{Q}^{\text{rank-1}} := \{Q \in \mathcal{Q} : \ \hat{c}_Q \leq \inf\{\hat{c}_{Q'} : Q' < Q, d_{Q'} = d_Q\}\} \tag{4.14}$$

as the set of all active nodes with a best-estimate at least as good as the best evaluated node at the same depth. If there is an open node Q at a depth d_Q which was not yet explored by the solving process, it holds that $Q \in \mathcal{Q}^{\text{rank-1}}$ since

$$\hat{c}_Q \leq \inf\{\hat{c}_{Q'} : Q' < Q, d_{Q'} = d_Q\} = \inf \emptyset = \infty.$$

Every time a node is branched on, its two children are inserted in an array $\mathcal{Q}^d$ of open nodes at their depth d. $\mathcal{Q}^d$ is sorted in nondecreasing order of the best-estimates of the nodes. In order to keep the set $\mathcal{Q}^{\text{rank-1}}$ updated, we maintain the minimum best-estimate of every processed node for every depth of the branch-and-bound tree. After a node $Q \in \mathcal{Q}^{\text{rank-1}}$ was selected to be explored next, we delete all nodes with a larger best-estimate from $\mathcal{Q}^{d_Q}$.

Using the following *rank-1 criterion*, we assume that the current incumbent is optimal when $\mathcal{Q}^{\text{rank-1}}$ becomes empty:

$$t_2^{\text{rank-1}} := \min\{t \geq t_1^* : \mathcal{Q}^{\text{rank-1}}(t) = \emptyset\}. \tag{4.15}$$

Recall that the rank-1 criterion $\mathcal{Q}^{\text{rank-1}} = \emptyset$ is never satisfied as long as there exist open nodes which are deeper in the tree than any previously explored node. The main difference between the rank-1 and the best-estimate criteria in (4.12) is that no direct comparison between an incumbent solution objective and the node estimates is performed.

4.4.3 A logarithmic model of the solving progress

In this section, we discuss the use of a logarithmic model of the solving progress for heuristic phase transition. Let P be a feasible MIP with nonzero objective $c \neq 0$, and let S be a solver to be used for solving P. After S has been started to solve P, over time we observe a series of $k \geq 0$ incumbent solutions $(t_i, \hat{y}(t_i))$, $i = 1 \ldots, k$, at discrete points in time $t_i > t_{i-1} > 0$ for all $i > 1$, and with strictly decreasing objective value $c(i) := c(t_i)$. For $k \geq 1$, we model the progress of S as a logarithmic function:

Definition 12 (logarithmic primal progress)**.** *Let S be a solver to solve a MIP P. Let $k \geq 1$ denote the number of incumbent solutions $\hat{y}(t_i)$ for P found by S at time t_i for $1 \leq i \leq k$. We call a logarithmic function $q : \mathbb{R}_+ \to \mathbb{R}$,*

$$q(t) = \alpha \log t + \beta$$

the logarithmic primal progress *of S for P with* intercept β *and* slope α*, if q minimizes the least-square-sum error*

$$\sum_{i=1}^{k} (q(t_i) - c(i))^2 = \min_{a,b \in \mathbb{R}} \sum_{i=1}^{k} (a \log(t_i) + b - c(i))^2. \tag{4.16}$$

For $k = 1$ solution, the logarithmic primal progress is not unique. Every function $r = a_r \log(t) + b_r$ such that

$$a_r \log(t_1) + b_r = c(1)$$

is a logarithmic primal progress. For $k = 2$, α and β can be determined by solving the linear system

$$\begin{pmatrix} \log(t_1) & 1 \\ \log(t_2) & 1 \end{pmatrix} \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} c(1) \\ c(2) \end{pmatrix}$$

which always has a unique solution because the matrix has full rank due to the requirement $t_1 \neq t_2$. The interesting case is $k \geq 3$.

Lemma 1. *Let S be a solver, which found $k \geq 3$ incumbent solutions for a MIP P at points in time $t_k > t_{k-1} > \cdots > t_1 > 0$. Let $x_i := \log(t_i)$ for all $1 \leq i \leq k$.*

The logarithmic primal progress q of S is unique, and its coefficients α, and β can be written as

$$\alpha = \frac{\sum_{i=1}^{k} c(i)x_i - \frac{1}{k}\sum_{i=1}^{k} x_i \sum_{i=1}^{k} c(i)}{\sum_{i=1}^{k} x_i^2 - \frac{1}{k}\left(\sum_{i=1}^{k} x_i\right)^2} \tag{4.17}$$

and

$$\beta = \frac{\frac{1}{k}\sum_{i=1}^{k} c(i) \sum_{i=1}^{k} x_i^2 - \frac{1}{k}\sum_{i=1}^{k} c(i)x_i}{\sum_{i=1}^{k} x_i^2 - \frac{1}{k}\left(\sum_{i=1}^{k} x_i\right)^2}. \tag{4.18}$$

Proof. Lemma 1 can be proven by setting the partial derivatives of (4.16) w.r.t. a and b to zero. □

The interesting observation about the regression coefficients of the logarithmic primal progress is that they only require constant-time operations for an update, if a new incumbent is found. Algorithm 2 formalizes the update procedure for the regression coefficients. It is called every time a new incumbent is found. The procedure maintains sum variables S_* for the relevant pieces of the primal bound history. In line 6, the procedure makes sure that if more than solution is found at time t_i, we keep only the last one. Otherwise, k is increased by 1. Hence, the solution times t_i represented by k are indeed all distinct. If the number of incumbent solutions k at distinct points in time is at least 3, we update the coefficients of the logarithmic primal progress.

The logarithmic primal progress provides us with a continuous function that approximates the (discrete) evolution of the primal bound of a MIP P over time. Although the logarithmic primal progress $q(t)$ is divergent for $t \to \infty$, its derivative $\partial q/\partial t = \alpha/t$ tends to 0.

In order to use the logarithmic primal progress information for phase transition, we need to measure the convergence of its derivative. Note that α and β depend on the scale upon which the objective function of the MIP acts. Since the objective scales can differ by orders of magnitude between two MIPs, it would be problematic to impose a general threshold δ on the value of the derivative of q for triggering the phase transition if (the absolute of) the derivative falls below δ. The following definition uses the tangent of the logarithmic primal progress. At every point in time $t \geq t_k$, the tangent of the primal progress has the form

$$l_t(u) = \frac{\alpha}{t}(u - t) + \alpha \log(t) + \beta = \frac{\alpha}{t}u \underbrace{-\alpha + \alpha \log(t) + \beta}_{y\text{-intercept } d_y(t).} \tag{4.19}$$

As heuristic phase transition, we wait for the tangent y-intercept to go below $c(k)$. We use the relation between the first incumbent solution value $c(1)$, the last incumbent solution value $c(k)$, and the y-intercept of the tangent of q as a scale-neutral alternative which we call progress factor:

Algorithm 2: update-regression

Input : new incumbent objective $c(k+1)$ at time t_{k+1}
Output : updated regression coefficients α, β of the logarithmic primal progress

Set $x_{k+1} := \log(t_{k+1})$;
if $k = 0$ **then**
  Initialize $S_{c^2}, S_c, S_{cx}, S_x, S_{x^2} := 0$;
end
Update $S_{c^2} := S_{c^2} + c(k+1)^2$, $S_c := S_c + c(k+1)$,
$S_{cx} := S_{cx} + c(k+1)x_{k+1}$, $S_x := S_x + x_{k+1}$, and $S_{x^2} := S_{x^2} + x_{k+1}^2$;
if $k \geq 1$ *and* $x_{k+1} = x_k$ **then**
  $S_{c^2} := S_{c^2} - c(k)^2$, $S_c := S_c - c(k)$, $S_{cx} := S_{cx} - c(k)x_k$, $S_x := S_x - x_k$, and $S_{x^2} := S_{x^2} - x_k^2$;
else
  Set $k := k + 1$;
end
if $k \geq 3$ **then**
  $\alpha := \frac{S_{cx} - \frac{1}{k} S_c S_x}{S_{x^2} - \frac{1}{k} S_x^2}$;
  $\beta := \frac{\frac{1}{k} S_c S_{x^2} - \frac{1}{k} S_{cx}}{S_{x^2} - \frac{1}{k} S_x^2}$;
else
  $\alpha, \beta := \emptyset$;
end
return α, β;

Definition 13. *Let α, $\beta \in \mathbb{R}$ be the coefficients of a logarithmic primal progress q, $k \geq 3$ the number of incumbent solutions, and $t' > t_k$. Let $d_y(t')$ denote the y-intercept of the tangent of q as a function of time. We define the* progress factor *$\lambda(t')$ as*

$$\lambda(t') := \frac{c(k) - d_y(t')}{c(k) - c(1)}. \tag{4.20}$$

Requiring that the y-intercept $d_y(t')$ falls below $c(k)$ is equivalent to $\lambda(t') \leq 0$.

If there were at least 3 distinct incumbent solutions found, the third of which at time t_3, we assume the current incumbent to be optimal when

$$t_2^{\log} := \min\{t \geq t_3 \ : \ \lambda(t) \leq 0\}. \tag{4.21}$$

We call the condition $\lambda(t) \leq 0$ in (4.21) *logarithmic criterion*. This idea is best illustrated in Figure 4.2, which shows the evolution of the primal bound over time as points $(t_i, c(i))$ for the instance csched010 obtained with SCIP during a computational experiment for Section 5.2 Note that although an optimal solution was found after 689 seconds for this instance, the Proof phase was not finished after

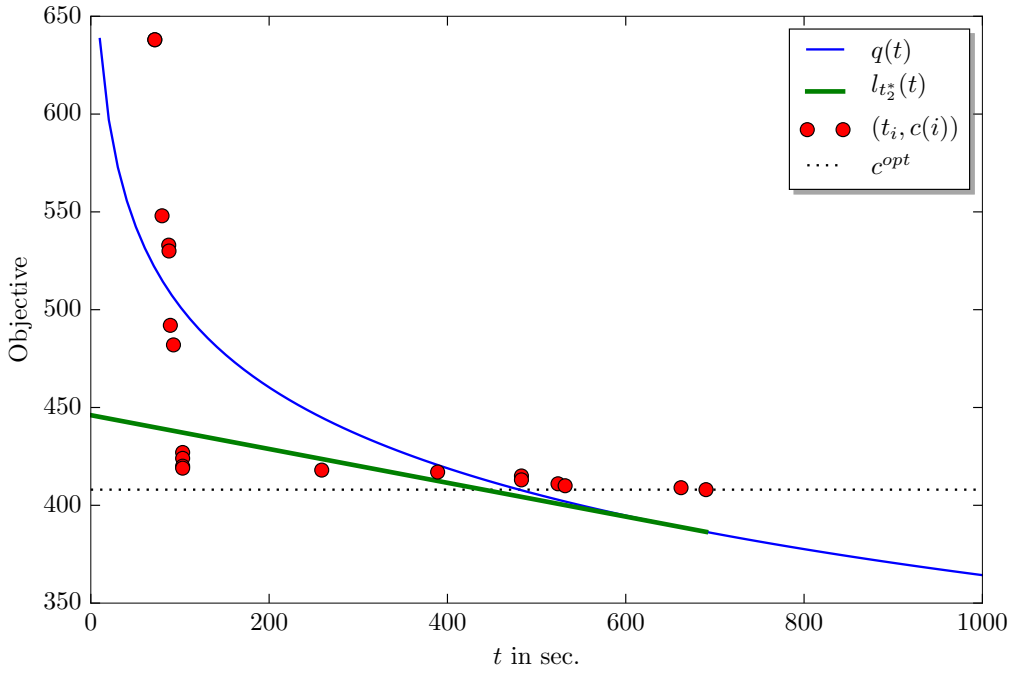


Figure 4.2: Example of a logarithmic primal progress $q(t)$ and its tangent $l_{t_2^*}(t)$.

a time limit of two hours. The rightmost point $(t_k, c(k))$ belongs to an optimal solution for this instance, as indicated by the dotted line. The logarithmic curve is the logarithmic primal progress $q(t)$ for all depicted points. Finally, the tangent $l_{t_2^*}$ of q at the second phase transition is shown. The y-intercept of the tangent is clearly above $c(k) = c^{\text{opt}}$, yielding a progress factor $\lambda(t_2^*) \approx 0.16$. Thus, the logarithmic criterion is not reached at time t_2^*. For $t' > t_2^*$ the progress factor is monotonously decreasing and reaches 0 at $t_2^{\log} \approx 1306$ seconds.

The use of the logarithmic primal progress has its limitations: We require at least three incumbent solutions. Furthermore, the use of time in all precedent equations can cause problems for the reproducibility of an experiment, when the time-measurement itself is non-deterministic, as in the case of SCIP.

For our experiments, we replace the time t_i, after which incumbent i for $1 \leq i \leq k$ was found, by the number of branch-and-bound nodes n_i and the number of LP-iterations κ_i. As a consequence, the number k of incumbent solutions might be different if we exchange the measure. If, e.g., we measure the number of nodes and a series of incumbent solutions is found during the root node, only the most recent is stored in the sums of Algorithm 2.

4.5 Estimates of search tree properties

In this section, we review some of the existing work on search tree size-/cost predictions for the branch-and-bound tree. Early work focused on predictions of

the number of explored tree nodes at termination. Let T denote the explored tree after termination of branch-and-bound. In general, we are interested in the value of a tree property

$$\phi(T) = \sum_{v \in T} \phi(v).$$

In this setting, different tree characteristics can be modeled by suitable choices for $\phi()$; the number of tree nodes can be counted with $\phi(v) = 1 \, \forall v \in T$. Other tree characteristics of interest are the solving time in sec., or the total amount of LP iterations. Indicator functions can be used to count the number of nodes that satisfy a certain property, as, e.g., leaf nodes. In practice, before the search has terminated, we have to rely on a suitable estimate $\hat{\phi}(T)$ based on incomplete information. Let $t \subseteq T$ be the subtree explored so far at some intermediate stage of the search. Let $d_t^{\max}$ denote the maximum depth of t.

Early work in this direction was conducted by Knuth [Knu74], who established an estimate $\hat{\phi}(T)$ of $\phi(T)$ for backtracking search. For this, repeated sample dives are performed, starting at the root node. For a sample $p \in \mathbb{N}$, the created subtree $t^{(p)}$ is a single path in which at every depth i, $i = 0, \ldots, d_{t^{(p)}}^{\max} - 1$, one of the f_i possible child nodes $1 \leq j \leq f_i$ is chosen at random following a uniform distribution, i.e. with probability $p_i(j) = \frac{1}{f_i}$, and expanded next. The dive terminates when $f_i = 0$. No backtracking is performed.

From the collected branching factors f_i, an estimate of the total number of nodes of T is constructed as

$$\hat{\phi}^{(p)}(T) = \sum_{i=0}^{d_{t^{(p)}}^{\max}} \prod_{j=0}^{i-1} f_i,$$

where v_i is the node expanded at depth i in $t^{(p)}$. Knuth verifies this estimate to be unbiased, i.e.

$$\mathbb{E}\left(\hat{\phi}^{(p)}(T)\right) = \phi(T) \quad \forall p \in \mathbb{N},$$

so that from the law of large numbers, we have

$$\lim_{p \to \infty} \frac{1}{p} \sum_{i=1}^{p} \hat{\phi}^{(i)}(T) = \phi(T) \quad \text{almost surely.}$$

Knuth notes, however, that such an estimator exhibits a large variance. In order to reduce the variance, he suggests to perform the child selection on a probability distribution other than uniform. It is shown that there exists an optimal choice for $p_i(j)$ making the estimator perfect, i.e. the estimator variance is zero. Such a sequence of probability distributions is, however, hard to find without knowledge of the entire tree.

Chen [Che92] describes a generalization of Knuth's sampling technique. Here, the nodes of the tree are partitioned into k different *strata* S_α, $1 \leq \alpha \leq k$, such that

$$T = \bigcup_{\alpha=1}^{k} S_\alpha, \qquad \text{and} \qquad S_\alpha \cap S_\beta = \emptyset, \beta \neq \alpha.$$

Furthermore, the strata are ordered top to bottom such that for every edge $(s,t) \in T$, $s \in s_\alpha$, $t \in S_\beta$, we have $\alpha > \beta$.

For a given stratification of the nodes into k strata, Chen's heuristic sampling expands a subtree of at most k nodes in a top to bottom manner. Each stratum S_α is represented by a single tree node s_α, together with an estimate of the size of this stratum ω_α. Strata which have not been processed so far are kept in a priority queue Q, sorted w.r.t the order relation on the strata. In every iteration, the maximum element $(s_\alpha, \omega_\alpha)$ is selected and removed from Q. For every child node t of s_α, the associated stratum $\alpha(t)$ is determined. If no representative for $\alpha(t)$ is present in Q, (t, ω_α) is added to Q. Otherwise, if $\alpha(t)$ already has some representative $(s_{\alpha(t)}, \omega_{\alpha(t)}) \in Q$, $\omega_{\alpha(t)}$ is increased by ω_α, and t replaces the current representative $s_{\alpha(t)}$ in Q with a positive probability. The sampling method terminates when Q is empty.

The estimator for $\phi(t)$ then reads

$$\hat{\phi}(T) = \sum_{\alpha=1}^{k} \omega_\alpha \phi(s_\alpha).$$

Choosing the (negative) depth as a stratifier, heuristic sampling becomes Knuth's sampling method.

In [CKL06], Cornuéjols et al. present a method to estimate the final size of the branch-and-bound tree early during the solving process. At every depth level i, $0 \le i \le d_t^{\max}$, their method counts the number of explored tree nodes $w_t(i)$ during the solving process so far. The quotients $\gamma_j = \frac{w_t(j+1)}{w_t(j)}$ for $0 \le j \le d_t^{\max}$ define the so-called γ-sequence of t. By its γ-sequence, the authors characterize the shape of t using the *last full level* $l_t := \min\{j \colon \gamma_j < 2\}$, the *waist* $b_t := \operatorname{argmax}_j\{w_t(j)\}$, and the maximum depth $d_t^{\max}$. Up to l_t, t is a complete binary tree. The authors observe that many branch-and-bound trees are complete binary trees at shallow depths, and that the *profile* of t, $(w_t(0), \ldots, w_t(d_t^{\max}))$, is monotonously increasing for $0 \le j \le b_t$, and monotonously decreasing afterwards.

Based on the assumption that the characteristics l_t, b_t, and $d_t^{\max}$ of the partial tree t resemble those of the final branch-and-bound tree T after termination, the authors construct an estimated γ-sequence $\hat{\gamma}$ for T as follows:

$$\hat{\gamma}_j := \begin{cases} 2, & 0 \le j \le l_t - 1, \\ 2 - \frac{j - l_t + 1}{b_t - l_t + 1}, & l_t \le j \le b_t - 1, \\ 1 - \frac{j - b_t + 1}{d_t^{\max} - b_t + 1}, & b_t \le j \le d_t^{\max}. \end{cases}$$

From these estimations, an estimate $\hat{n}_T$ of the number of nodes n_T is given by

$$\hat{n}_T := 1 + \sum_{i=1}^{d_t^{\max}} \hat{w}_T(i) = \sum_{i=1}^{d_t^{\max}} \prod_{j=0}^{i-1} \hat{\gamma}_j$$

The advantage of the latter model is that it can be updated during search tree exploration and is computationally cheaper than the methods of Knuth's and Chen's

which require a number of independent sample dives into the tree starting from the root node. Both Knuth and Chen noted that their method is applicable to branch-and-bound search if the optimal objective value is known beforehand. Therefore, such methods are best employed at the beginning of the Proof phase in order to estimate the remaining effort to complete the search.

Chapter 5

Computational results

This chapter comprises all computational experiments conducted for working on the MIP phases. First, a beneficial mixture of components is determined individually for each phase in Section 5.1. Results for the heuristic phase transition criteria from the Improvement phase to the Proof phase are subject to Section 5.2. Finally, we combine beneficial phase settings for each phase inside a phase-based solver, which uses the presented criteria for a heuristic phase transition. We compare our approach to SCIP with default settings and a phase-based solver that can exactly determine the phase-transitions.

Figure 5.1 serves as a motivation for the conducted experiments. It shows how the overall solving time is distributed over the three phases, where we restrict the diagram to instances for which the solving time in our setup lies between 10 seconds and 2 hours. On average, the SCIP default settings spend approximately 13 % during $\mathcal{P}_1$, 45.5 % on $\mathcal{P}_2$, and 41.5 % during $\mathcal{P}_3$. The figure shows that the solving process spends more than 50 % time for a proof of optimality on roughly a third of the instances.

5.1 Individual phase experiments

In Section 4.2, we partitioned the solving process into a Feasibility phase, an Improvement phase, and a Proof phase. For each of the phases, we conducted computational experiments with different settings of SCIP. The selection of settings was based on the discussion Section 4.3 of MIP-solver components and their use regarding the phase objectives.

For the experiments in this section, we used an oracle that could exactly determine the phase transition from the Improvement phase to the Proof phase, see also Section 5.1.2. All computations were performed on a cluster of 32 computers. Each computer runs with a 64bit Intel Xeon X5672 CPUs at 3.20 GHz with 12 MB cache and 48 GB main memory. The operating system was Ubuntu 14.4. A gcc compiler was used in version 4.8.2. Hyperthreading and Turboboost were disabled. We ran only one job per computer in order to minimize the random noise in the measured running time that might be caused by cache-misses if multiple processes

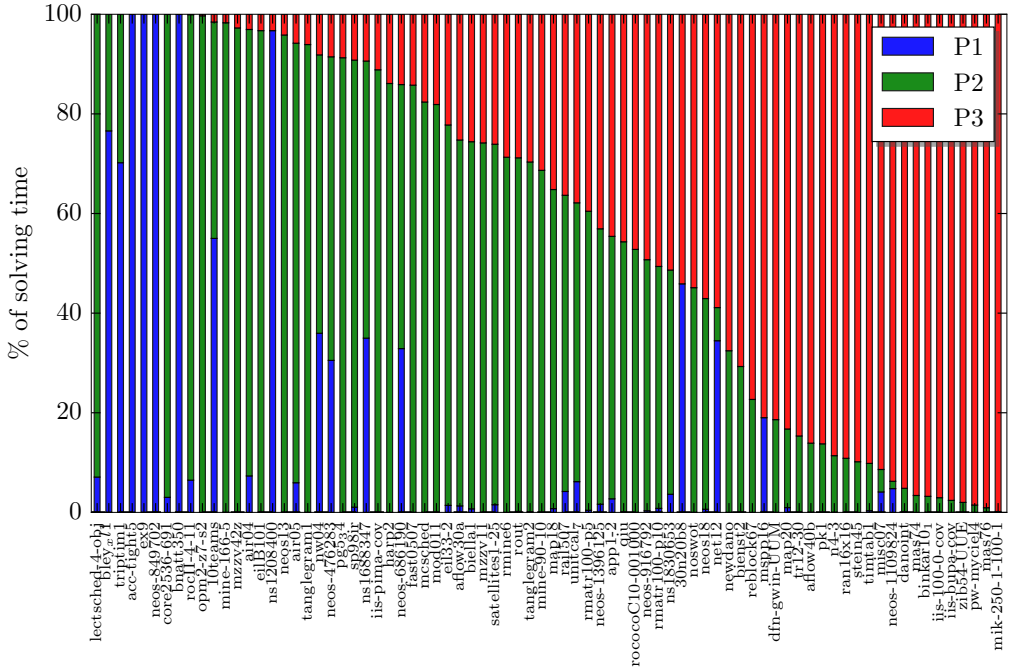


Figure 5.1: Percentaged distribution of overall solving time on individual phases.

share common resources. If not otherwise stated, the experiments were run with a time limit of 1h and a 24 GB memory limit. As SCIP version, we used SCIP 3.1.0, which we extended by the *Distribution diving* heuristic ([PC11], cf. Section 4.3.1) and several event handlers to store additional information and to trigger the phase transition. We use SOPLEX [sop] in version 2.0 as solver for the LP-relaxations at each node.

5.1.1 Feasibility phase

The goal of the Feasibility phase is a guidance of the solving process to a first feasible solution. We suggest settings which influence the branch-and-bound search alone. Thus, our experimental setup to compare different settings w.r.t. their performance to find a first feasible solution involves only instances from our library (cf. Section 3.1) for which SCIP with default settings is not able to provide a feasible solution until the root node is finished. Further, we excluded three instances which are infeasible, and the instance stp3d, for which SCIP did not finish solving the root node within the time limit of 1h with any of the suggested settings. These restrictions yield a remaining test set of 32 instances.

We tested nine different settings, which are summarized in Table 5.1. The setting `default` represents the default settings of SCIP. In the setting `act&dist`, we activated two additional diving heuristics: *Distribution diving* and *Active constraint diving* [PC11], each with a `frequency` of 3. For details about these heuristics, see 4.3.1. In a third setting `aggrdive`, we set the `frequency` parameter to 1

Table 5.1: SCIP settings tested during the Feasibility phase experiment.

Setting	Explanation
`default`	SCIP with default settings
`act&dist`	SCIP default settings together with two more diving heuristics
`aggrdive`	more aggressive diving heuristics
`inf`	use of *inference branching* rule
`dfs inf`	*dfs* node selection and *inference branching* rule
`rdfs inf`	*dfs* node selection with periodic restarts and *inference branching*
`uct inf`	SCIP default settings with *uct* node selection for the first 31 nodes.
`uct-rdfs inf`	*uct* node selection for 31 nodes, then like `rdfs inf`.
`br-rdfs inf`	*breadth-first* node selection for 31 nodes, then like `rdfs inf`

for all diving heuristics and increased the limits on LP iterations before the diving is terminated. The *inference branching* rule was tested together with various node selection rules: `inf` uses the default node selection of SCIP, while `dfs inf` and `rdfs inf` use the *dfs* and *restartdfs* node selection rules, respectively. The last group of tested settings uses two-level node selection strategies: At the very top levels of the search tree, exploratory node selection rules are used, whereas after a small number of nodes, the node selection is switched to a different rule. All of the settings in this group use an *inference branching* rule, and switch the node selection after 31 nodes, corresponding to a complete binary tree of depth 5. In `uct inf`, we apply the *uct* [SSR12] node selector for the first 31 nodes before switching back to the default estimate-based node selection of SCIP. The setting `uct-rdfs inf` uses the `rdfs inf` setting after an initial search tree exploration via *uct*. The last setting `br-rdfs inf` first uses a *breadth-first* search, and then switches to a *restartdfs* node selection. Finally, we set a solution limit of 1 for every setting.

The results of the experiment are summarized in the Tables 5.2 and 5.3. For an instance-wise outcome, we refer to the Tables H.1–H.5 in the appendix.

We use four different measures for evaluation: The first column $t_{>0}$ shows the time span in seconds after the solve of the root node was finished until the first solution was found, or the time limit of 1h was reached. The second column n shows the number of explored branch-and-bound nodes before termination because a feasible solution was found, or the time limit was hit. The average LP iterations per node that were performed after the root node solve was finished are presented in total (κ/n) and without strong branching iterations($\kappa^{-\mathrm{sb}}/n$), respectively. The shown results are shifted geometric means with a shift of 10 sec, 100 nodes, and 100

Table 5.2: Shifted geometric means of the time after root $t_{>0}$ and the number of nodes with a shift of 10 sec and 100 nodes.

	$t_{>0}$	%	p	n	%	p
`act&dist`	68.8	99.6	0.792	332.2	100.0	0.936
`aggrdive`	64.3	93.0	0.632	275.8	83.0	0.277
`br-rdfs inf`	34.4	49.7	0.001	269.7	81.2	0.583
`default`	69.1	100.0		332.3	100.0	
`dfs inf`	51.7	74.9	0.117	1304.5	392.6	0.003
`inf`	49.7	71.9	0.166	491.6	147.9	0.290
`rdfs inf`	29.4	42.5	0.000	572.9	172.4	0.045
`uct inf`	36.7	53.1	0.004	228.5	68.8	0.125
`uct-rdfs inf`	31.7	45.9	0.000	318.4	95.8	0.969

LP iterations per node. We also give a percentage for every measure compared to the `default` setting. Percentages lower than 100 % are an improvement, while percentages above 100 % denote a deterioration of the phase-1 performance. In a third column named p, we show the two-sided p-value obtained from a Wilcoxon-signed-rank test, see Section 3.3.4, between `default` and the setting in the corresponding row. Small p-values indicate that the changes in the corresponding mean value come from an improved (deteriorated) behavior over the test set, rather than only a few outliers.

All non-default settings outperform `default` w.r.t. the time measure, albeit the improvement observed for the `act&dist` setting is negligible. The setting `aggrdive` yields an improvement of 7 % over the `default` setting.

A more significant time reduction can be observed for the remaining settings using an *inference branching* rule. Replacing the default branching rule by an *inference branching* rule improved the time after root mean by 28 %. The best setting in this respect is `rdfs inf`, which reduces the time by almost 60 %. The `dfs inf` setting is the slowest amongst all settings which incorporate *inference branching*. Yet, it is 25 % faster in the shifted geometric mean than the `default` setting. It should also be noted that `dfs inf` could not find feasible solutions within the time limit on 3 instances. In these cases, the time after root is an actual lower bound for the true time this setting will take until it finds a feasible solution. There are two other settings that could not find solutions for every instance within the time limit: `inf` fails on 2 instances, `rdfs inf` on 1. The fastest setting which finds feasible solutions for all 32 instances is `uct-rdfs inf`, closely followed by `br-rdfs inf`.

Also the `uct inf` setting shows a substantial improvement over the `default` setting with a time reduction of about 47 %.

It is not surprising that especially those settings which do not use *strong branching* outperform the `default` setting w.r.t. time, because *strong branching* often consumes almost twice as many LP iterations as the tree search itself. The column κ/n shows the LP iterations per node after the root solve was finished, including

Table 5.3: Shifted geometric means of the LP iterations after root κ/n and $\kappa^{-\mathrm{sb}}/n$ with and without counting iterations used within *strong branching*. A shift of 100 iterations was used.

	κ/n	%	p	$\kappa^{-\mathrm{sb}}/n$	%	p
`act&dist`	1615.4	108.3	0.687	680.6	107.7	0.687
`aggrdive`	1795.5	120.4	0.084	766.1	121.2	0.030
`br-rdfs inf`	707.5	47.4	0.005	707.5	112.0	0.614
`default`	1491.7	100.0		631.9	100.0	
`dfs inf`	110.2	7.4	0.000	110.2	17.4	0.000
`inf`	462.0	31.0	0.000	462.0	73.1	0.166
`rdfs inf`	125.4	8.4	0.000	125.4	19.9	0.000
`uct inf`	833.6	55.9	0.030	833.6	131.9	0.155
`uct-rdfs inf`	572.0	38.3	0.001	572.0	90.5	0.681

strong branching iterations. The iteration columns include LP iterations performed during the execution of diving heuristics. Between the `default` setting and the `rdfs inf` setting, a factor of 12 times more LP iterations per node is encountered for the default setting. Even more LP iterations are performed by the settings `act&dist` and `aggrdive`. Since more than 50 % of the LP iterations are spent during *strong branching*, we subtract the LP iterations used for *strong branching* in the column $\kappa^{-\mathrm{sb}}/n$. The shifted geometric means are equal for all settings using an *inference branching* rule. Even if *strong branching* iterations are not counted, the `inf` setting needs 27 % less iterations per node on the average than SCIP with default settings. Both the `dfs inf` and `rdfs inf` settings show a reduction of more than 80 % iterations per node. Combining *restartdfs* with an exploratory search at the beginning, the reduction of `rdfs inf` and the higher iteration numbers caused by an initial *uct* or *breadth-first* node selection are within 12 % of the iterations per node consumed by the `default` setting, `uct-rdfs inf` taking less iterations than `br-rdfs inf`. Recall that backtracking from a non-terminal node of the tree causes both the LP relaxation to take more iterations and triggers the execution of a diving heuristic. The small number encountered for `rdfs inf` compared to `uct inf` also stems from the fact that SCIP does not apply diving heuristics at all when using *dfs* or *restartdfs* search. The use of an aggressive diving schedule or `uct inf` in addition to the default node selection rule increases the iterations count per node by 21 % and 31 % w.r.t. the default settings, respectively.

The n column shows `uct inf` as best setting with a node reduction of 31 % compared to the `default` setting. A very aggressive diving strategy applied by the setting `aggrdive` also yields a substantial node reduction of 17 %. The setting `rdfs inf` requires 72 % more nodes than the `default` setting. Note that n does not account for auxiliary search nodes explored by `default` within diving heuristics. The use of pure *dfs* or *restartdfs* node selection rules increases the number of phase-1 nodes, by a factor of almost 4 for the `dfs inf` setting. A comparison of the *inference branching* and `uct inf` columns reveals an interesting observation:

While the use of an *inference branching* rule together with the SCIP default node selection rule increases the phase-1 number of nodes by 47 % in the mean, the use of *uct* for an initial exploration almost reverts this degradation into a node reduction of 31 %.

We conclude that adjustments to the default settings of SCIP can substantially improve the solver performance whenever no feasible solution was found during the processing of the root node. The results show in particular that the use of an *inference branching* rule outperforms a *reliability pseudo/inference branching* rule, which is the default in SCIP, if in addition the node selection strategy is altered. Note that giving more weight to the inference score in the hybrid rule *reliability pseudo/inference branching* is an alternative to the use of *inference branching* alone. For a discussion of the weights, we refer to Sections 2.3.1 and 4.3.3.

As we expected, *dfs* without any backtracking strategy gets sometimes trapped in infeasible portions of the search tree. This behavior is partially repaired by a simple backtrack strategy such as periodic restarts after 100 leaf nodes. The time improvement is highest for a *restartdfs* node selection and an *inference branching* rule. This setting outperforms a pure *dfs* node selection in all respects. One should keep in mind, however, that diving heuristics, which diversify the search algorithm by providing alternatives to the main branching rule, are not applied in this setting. Still, `rdfs inf` could not find a solution for the instance rd-rplusc-21. Two-level strategies `uct inf`, `uct-rdfs inf` and `br-rdfs inf` were almost as fast as `rdfs inf`, but find feasible solutions for all instances. We conclude that these strategies are best suited for the Feasibility phase amongst all tested settings because their trade-off between exploration, exploitation, and the application of alternative branching strategies through diving heuristics outperforms the default settings significantly without missing a solution.

Furthermore, the results indicate that the use of a two-level node selection rule such as `uct-rdfs inf`, `uct-rdfs inf`, or `br-rdfs inf` can be useful inside LNS-heuristics, where every feasible solution is a new incumbent to the original MIP. Another, yet less substantial improvement of the Feasibility phase performance was observed for a more aggressive schedule of all diving heuristics of SCIP, whereas the use of two additional, feasibility-driven diving heuristics alone, *Active constraint diving* and *Distribution diving*, does not substantially alter the Feasibility phase performance when used inside SCIP.

Although we observed significant speed-ups compared to the default Feasibility phase performance of SCIP, altering the branching rule to *inference branching* during that phase may lead to an undesired growth of the overall search tree and make it harder or even impossible to achieve the goals of the Improvement phase and the Proof phase.

5.1.2 Improvement phase

The goal of the Improvement phase is to guide the branch-and-bound search towards an optimal solution after the feasibility of the model has been proven during the Feasibility phase. In order to compare settings for the Improvement phase, we

selected all feasible models from our test set for which SCIP with default settings does not find the first solution and an optimal solution at the same node of the branch-and-bound tree. In particular, this excludes instances for which the first found solution is already optimal (thereby omitting the Improvement phase). We excluded another four instances for which the optimal objective value is unknown by the time of this writing. Finally, we excluded the instances markshare1 and harp2 for which numerical troubles led to a slight infeasibility of the incumbent when reaching the time limit in two cases. The remaining test set consists of 120 instances.

The settings we tested in this section put different emphasis on how primal heuristics are applied during the Improvement phase, in particular LNS-heuristics. Apart from the default settings of SCIP, we tested aggressive heuristic settings `agg`. In addition to a more frequent execution strategy for the SCIP primal heuristics, the limits for diving and LNS-heuristics concerning LP iterations and sub-MIP node limits, respectively, are less strict than in the `default` setting. For a detailed list of changed parameters for the `agg` setting, we refer to Section H.1.1. The `l-agg` setting uses aggressive settings only for the LNS-heuristics, while other heuristics, in particular diving heuristics, are applied as by the default settings of SCIP. Another setting, $\texttt{agg}_{05}$, is based on the aggressive setting `agg` for heuristic execution and, additionally, decreases the `minimprove` parameter for all LNS-heuristics from 0.01 to 0.005. Three more settings alter the node selection rule used within the sub-MIPs of the LNS-heuristics; `l-uct`, `agg l-uct`, and `l-agg l-uct`, which adapt the settings `default`, `agg`, and `l-agg`, respectively, and additionally use a *uct* node selection for the exploration of the first 31 nodes of the sub-MIPs, yielding a two-level node selection rule inside LNS-heuristics. We test the two-level strategy for solving sub-MIPs as a consequence of the Feasibility phase experiment, where the two-level strategies were the quickest to find solutions on all tested instances. Recall that every feasible solution to a sub-MIP is a new incumbent for the original MIP.

We altered the code of SCIP to read in the optimal solution values from a file and interrupt the solving process as soon as an incumbent with optimal solution value is found. Some of the known optimal solutions are only precise up to a certain number of decimal digits. We allow slight relative deviations from the reference values by using a tolerance of 10^{-9}. Values a, b are considered equal if

$$|a - b| \leq 10^{-9} \cdot \max(|a|, |b|, 1). \tag{5.1}$$

Alternative possibilities such as passing the optimal value either as a constraint or by setting an objective limit would have an effect on the solving process itself.

SCIP was run with default settings until a first feasible solution was found. After the first feasible solution was found, the settings were altered as described above. We only present aggregated results in this section, for an instancewise outcome, we refer to Table H.6. We present Table 5.4 to compare the tested settings w.r.t. the number of instances for which they provided optimal solutions within the time limit. From the 120 instances, the `default` setting finishes the Improvement phase within the time limit on 89. The best settings in this respect are $\texttt{agg}_{05}$, `l-agg`,

Table 5.4: The number of instances for which an optimal solution was found.

	default	agg	agg l-uct	agg_{05}	l-agg	l-agg l-uct	l-uct
optimal	89	91	90	92	92	92	91

Table 5.5: Shifted geometric means for the Improvement phase regarding the solving time, and nodes on 88 instances for which all settings could finish the phase. The shifts used are 10 sec., and 100 nodes

	$t_{\mathcal{P}_2}$	%	p	$n_{\mathcal{P}_2}$	%	p
default	62.5	100.0		1202.4	100.0	
agg	62.0	99.2	0.473	818.0	68.0	0.000
agg l-uct	69.0	110.5	0.029	900.5	74.9	0.001
agg_{05}	62.1	99.4	0.237	837.8	69.7	0.000
l-agg	61.2	97.9	0.448	825.3	68.6	0.000
l-agg l-uct	67.0	107.2	0.157	959.9	79.8	0.001
l-uct	60.4	96.7	0.090	1178.6	98.0	0.152
virtual	42.9	68.6	0.000	464.8	38.7	0.000

and l-agg l-uct, each of which finds optimal solutions for 92 instances. There are 94 instances in total for which at least one setting finished the Improvement phase. All seven settings succeed on 88 instances. These instances constitute the basis for the time and node performance comparison.

On the subset of 88 instances for which all settings could find optimal solutions, we present the shifted geometric mean time and number of nodes during the Improvement phase only, i.e., between the first and second phase transition. The results are shown in Table 5.5. The p-column shows the two-sided p-value obtained from a Wilcoxon-signed rank test between the corresponding setting and default. The table also shows the results for a virtual setting, which we would obtain if we were able to choose the best setting instance-wise. Note that the virtual setting can be different depending on the choice of only the phase-2 time or the phase-2 nodes as a criterion. The time column shows slight improvements for four of the actual settings, the best setting being l-uct which reduced the phase-2 time by 3.3 % in the shifted geometric mean. Two settings, agg l-uct and l-agg l-uct, performed worse than default by 10.5 % and 7.2 %, respectively. The virtual best setting requires only 68.6 % of the phase-2 time compared to the default settings.

The number of phase-2 nodes, which is shown in column $n_{\mathcal{P}_2}$, shows significant reductions with all five aggressive heuristic settings by 20 %–32 %. The p-column indicates these reductions to be significant on a 1 %-level. Even the setting l-uct yields a slight improvement of 2 %. A virtual best setting could outperform the default settings in this respect by more than 60 %. The performance profile in Figure 5.2 depicts the node reductions graphically. A performance profile shows for every factor k the percentage of instances that a setting could finish within

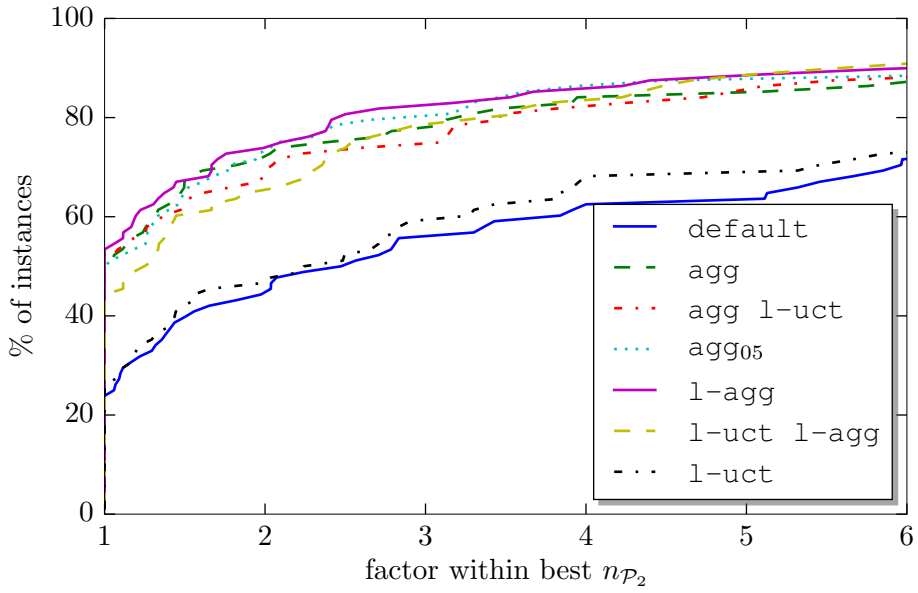


Figure 5.2: phase-2 node performance on 88 instances for which all settings finish the Improvement phase within the time limit of 1 hour.

k times the result of the virtual best solver. The five aggressive settings clearly lie above the default settings in this respect, which is also slightly inferior to the `l-uct` setting. On the other hand, there is no setting which comes closer than a factor of 2 on 80 % of the instances.

We restrict this evaluation to the instances where all settings finished the second phase within the time limit because we generally suggest settings which increase the computation times per node by applying more heuristics per node, or by increasing the running time of LNS-heuristics with a more exploratory node selection rule (at the beginning), thereby consuming more simplex iterations per node on the average. We now compare the different settings regarding their primal integral during the Improvement phase. The advantage of this measure is that it accounts for both the objective value of an incumbent and the solving time until it was found, which allows us to include the instances which hit the time limit.

In Table 5.6, we show the shifted geometric mean integrals of the Improvement phase only. We use a shift of 1000 which corresponds to a gap of 100 % over 10 seconds. The settings `agg` and `agg`$_{05}$ improve the integral obtained with `default` by 5.4 % and 3.7 %, respectively, whereas `l-agg` only shows a minor improvement of 2.4 %. A Wilcoxon signed rank test does not reveal a significant improvement for any of the non-default settings. We also show a performance profile of the primal-optimal gap at termination in Figure 5.3. We compare primal-optimal gaps on 32 instances for which at least one setting did not finish the Improvement phase. We see that the curve for `default` lies centered along all factors k we present here. None of the other settings can dominate the default settings of SCIP for all factors

Table 5.6: Shifted geometric means for the primal integral during the Feasibility phase with a shift of 1000.

	$\Gamma_{\mathcal{P}_2}(T)$	%	p
`default`	2796.7	100.0	
`agg`	2644.6	94.6	0.166
`agg l-uct`	2652.3	94.8	0.331
$\texttt{agg}_{05}$	2694.0	96.3	0.336
`l-agg`	2729.2	97.6	0.434
`l-agg l-uct`	2706.7	96.8	0.451
`l-uct`	2762.1	98.8	0.350
`virtual`	2222.3	79.5	0.000

Table 5.7: Shifted geometric mean time in seconds for the Improvement phase over all 120 instances with a virtual setting that uses the better of the two settings. Values on the diagonal show the shifted geometric mean time for this setting.

	`default`	`agg`	`agg l-uct`	$\texttt{agg}_{05}$	`l-agg`	`l-agg l-uct`	`l-uct`
`default`	208.8	176.8	185.7	179.5	171.0	180.0	194.1
`agg`		203.0	191.8	190.5	173.7	182.0	174.3
`agg l-uct`			217.7	191.2	180.2	190.6	182.9
$\texttt{agg}_{05}$				204.3	172.4	182.8	178.9
`l-agg`					196.2	184.2	169.4
`l-agg l-uct`						210.0	179.4
`l-uct`							200.4

$k \in [1, 5]$, although the settings `l-agg l-uct` and `agg` show a better performance in this respect for factors of 2 and above.

Table 5.7 shows the shifted geometric mean phase-2 time in seconds for each individual setting on the diagonal. In contrast to the previous Table 5.5, we include all tested instances into the calculation of the mean. In every entry other than the diagonal, we present shifted geometric mean phase-2 times if we were to choose the better of the settings in the corresponding row and column on each individual instance. The table shows that an optimal combination of `l-agg l-uct` and `l-agg` with a combined mean of 169.4 sec. comes closest to the virtual best solver over all settings, which takes 138.9 sec. Table 5.7 also reveals `l-agg` to be the most beneficial complement for every setting except for `l-agg l-uct` and `l-agg` itself.

The results show that the primal heuristics of SCIP, which are our main focus in this section, are already well-tuned for the Improvement phase. Although the tested settings could increase the total number of instances for which the Improvement phase is finished within the time limit, none of them could significantly outperform the $\mathcal{P}_2$ performance of the SCIP default settings regarding time or the primal integral. From the significant node reduction using an aggressive primal heuristic policy, we assume that there exists a trade-off setting that can outperform the

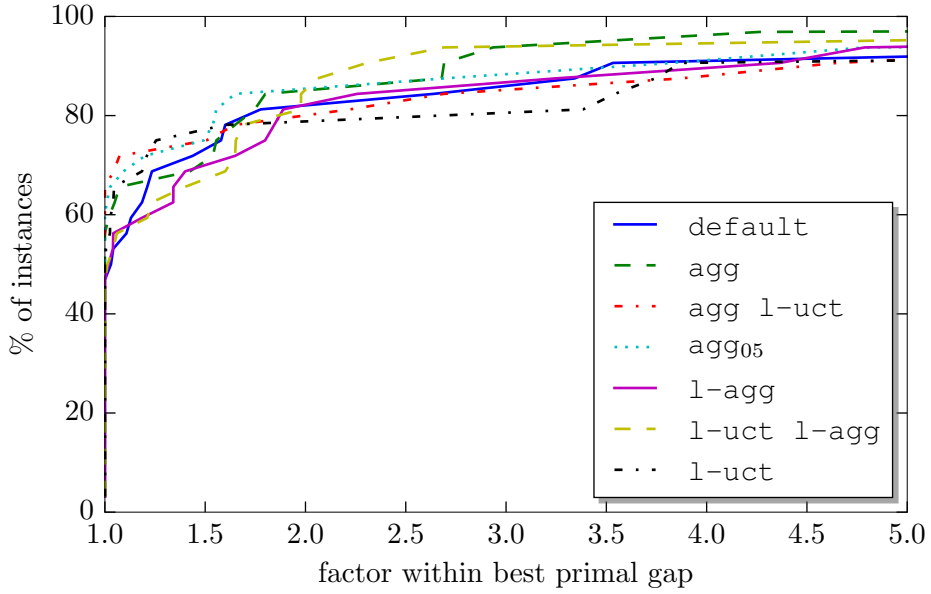


Figure 5.3: Profile of the primal gap after termination on 32 instances for which at least one setting did not finish the Improvement phase within the time limit of 1 hour.

default settings regarding both time and nodes. The results obtained by a virtual best setting indicate the potential of the tested settings together inside a parallel MIP solver such as [SAB+12].

The small differences between the `l-agg` and the `agg` setting in this experiment indicate that the positive effects for the Improvement phase can be mainly attributed to the LNS-heuristics. Another observation concerns the use of a two-level node selection strategy inside LNS-heuristics. While we could improve the default settings in all respects by using a *uct* node selection rule for the sub-MIP solving process inside LNS-heuristics, the opposite is true if we apply a two-level node selection in combination with a more aggressive execution strategy of the LNS-heuristics. This result motivates further experiments for finding a beneficial search strategy to be used by the LNS-heuristics inside SCIP.

5.1.3 Proof phase

In this section, we concentrate on the solver performance after an optimal solution has been found. We call this final phase the Proof phase of the search, cf. Section 4.2. In practice, the Proof phase can only be detected at its very end, when the dual bound and the primal bound of the problem are equal.

For the experiments in this section, we use a time limit of 2h. From our test set, we collected the subset of 103 instances for which SCIP finds an optimal solution within 2h, but the proof of optimality requires the solving of at least 1 additional node. SCIP was run with default settings until an optimal incumbent

was found. We consider six different settings for the Proof phase: Apart from SCIP with default settings, we test a setting with all primal heuristics disabled and a pure *dfs* node selection, `dfsHeuroff`. Furthermore, we compare two different modifications to the *reliability pseudo/inference branching* branching rule. The setting `weights` rebalances the individual weights for the cutoff, conflict, and pseudo-cost score weights of the candidate variables. The rebalancing compares the number of pruned infeasible nodes to the number of nodes that could be bounded, and adjusts the weights accordingly to better reflect this ratio which we introduced as leaf ratio in Section 4.3.3.

In the `error based` setting, we use a relative error criterion for the reliability of candidate pseudo-costs; If the relative error, which we introduced in Section 4.3.3, of the observed unit gains of a candidate variable is above $\eta_{\text{conf}} = 50\,\%$, we consider the pseudo-costs unreliable and perform *strong branching* for this candidate.

The `sepa` setting intensifies the use of cutting plane algorithms. Especially, cutting plane algorithms are also applied at other nodes than only the root node. For an overview of the individual parameters of the setting, we refer to Section H.1.2 a last setting `combined`, we apply the three settings `weights`, `dfsHeuroff`, and `sepa` together.

We focus on 97 instances of the test set that could be solved within the time limit by all settings. We further subdivide these instances into an *easy* set of instances which could be solved within less than 200 seconds by all of the tested settings, and a group of *hard* instances for which at least one setting needed more than 200 sec. The easy and hard sets contain 48 and 49 instances, respectively. We measure the individual time and nodes, $t_{\mathcal{P}_3}$ and $n_{\mathcal{P}_3}$, respectively, after an optimal solution was found (and the settings were changed) until the Proof phase was finished. The shifted geometric means are presented in Table 5.8 for the easy and the hard instances individually, together with the overall solving time t (sec). We used a shift of 10 seconds and 100 nodes, respectively. The third column p shows the two-sided p-values obtained from a Wilcoxon signed-rank test between the corresponding rows compared with the `default` setting.

First we compare `default` and `dfsHeuroff`. On the set of easy instances, `dfsHeuroff` has a better performance than `default` w.r.t. both the phase-3 time and the overall solving time. It takes `dfsHeuroff` 11 % less time to finish the Proof phase on average. The improvement regarding the overall solving time, which includes the time for the Feasibility phase and the Improvement phase, is still 4.8 % on average, while the required number of nodes changes by less than 0.1 %. On the hard instances, however, the use of a `dfsHeuroff` setting increases $t_{\mathcal{P}_3}$ by 1.0 % on average, and the overall solving time by 4.0 %. Furthermore, we observe an increase of the required solving nodes in the third phase by 9.9 % on the hard instances. Both time columns show p-values of less than 1 %, although the percentage difference compared to the default settings is rather small, especially for the phase-3 time. The reason for this is a single outlier, enlight13, for which the use of `dfsHeuroff` increases the phase-3 time by a factor of 36, and the number of nodes by a factor of 143.

Figure 5.4 shows a histogram of the distribution of the logarithmic shifted

Table 5.8: Results for the Proof phase for 48 easy and 49 hard instances separately, and for all instances.

(a) easy instances: solved to optimality by every setting in no more than 200 sec.

	t (sec)	%	p	$t_{\mathcal{P}_3}$	%	p	$n_{\mathcal{P}_3}$	%	p
weights	14.5	100.4	0.167	6.5	100.7	0.156	658.0	100.0	0.457
combined	14.4	100.0	0.450	6.4	99.8	0.275	637.7	96.9	0.003
default	14.4	100.0		6.4	100.0		658.2	100.0	
dfsHeuroff	13.7	95.2	0.050	5.7	89.0	0.000	658.3	100.0	0.426
error based	14.7	102.3	0.014	6.9	108.0	0.010	557.1	84.6	0.001
sepa	15.3	106.4	0.000	7.2	112.2	0.007	637.8	96.9	0.001

(b) hard instances: solved to optimality in more than 200 seconds by at least 1 setting.

	t (sec)	%	p	$t_{\mathcal{P}_3}$	%	p	$n_{\mathcal{P}_3}$	%	p
weights	693.4	100.0	0.089	233.4	98.1	0.206	11583.4	97.0	0.392
combined	671.0	96.8	0.000	216.4	90.9	0.000	10656.3	89.2	0.000
default	693.3	100.0		238.0	100.0		11939.9	100.0	
dfsHeuroff	721.0	104.0	0.001	240.3	101.0	0.001	13072.4	109.5	0.316
error based	753.8	108.7	0.023	259.9	109.2	0.099	11703.3	98.0	0.002
sepa	684.6	98.7	0.340	230.1	96.7	0.206	9984.1	83.6	0.000

(c) all instances in 5.8a or 5.8b.

	t (sec)	%	p	$t_{\mathcal{P}_3}$	%	p	$n_{\mathcal{P}_3}$	%	p
weights	123.5	100.1	0.073	54.2	99.1	0.101	2918.2	98.4	0.458
combined	121.2	98.2	0.001	51.8	94.6	0.000	2756.2	93.0	0.000
default	123.3	100.0		54.7	100.0		2964.7	100.0	
dfsHeuroff	124.0	100.6	0.000	53.6	98.0	0.000	3107.2	104.8	0.353
error based	129.9	105.4	0.002	58.5	107.1	0.010	2726.7	92.0	0.000
sepa	125.0	101.3	0.037	55.1	100.8	0.235	2664.6	89.9	0.000

phase-3 time quotients that are responsible for the p-values. Instances with a shifted logarithmic quotient below 0 are the instances where `dfsHeuroff` outperforms `default`. The figure shows an asymmetric distribution in favor of `dfsHeuroff` except for a single outlier enlight13 to the right. Over the entire set of instances, we observe a reduction of the phase-3 time by 2.0 % compared to the default set. In addition, we could solve one more instance to optimality within the time limit using `dfsHeuroff`.

In order to keep the comparison fair, we remove the most extreme outliers on both sides. We present in Table 5.9 the results for all 97 instances if we dropped the single leftmost and rightmost outliers w.r.t. the shifted quotients. Each row of the table shows the changed shifted geometric means for the corresponding setting and for the `default` settings after the removal of the two outliers. The column left out shows the instance for which `default` was outperformed the most regarding the shifted quotients, the right out-column contains the instance with the largest

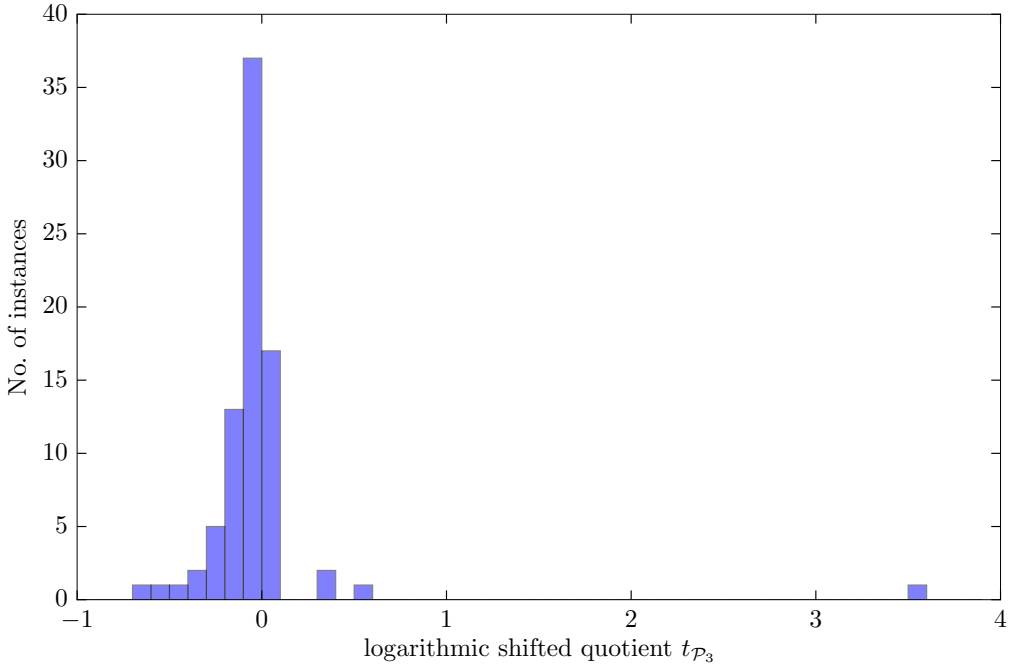


Figure 5.4: Distribution of logarithmic shifted geometric quotients comparing `dfsHeuroff` and `default` settings. The bin width is 0.1.

shifted quotient, i.e. the most extreme case in favor of `default`. Note the almost exclusive presence of the two instances satellites1-25 and enlight13 as leftmost and rightmost outliers, respectively, for both phase-3 time and nodes.

After the removal of outliers, we observe a 5.5 % phase-3 time improvement for `dfsHeuroff` and no significant change of the number of phase-3 nodes compared with `default`. The `error based`-setting shows the highest percentaged node reduction of 11.2 %, but comes with a running time deterioration of 4.4 %. By adjusting branching weights, we do not obtain a significant node reduction. More aggressive cutting plane separation yields a significant improvement regarding the number of phase-3 nodes. The combination of more aggressive separation, adjusting branching weights, and `dfsHeuroff` yields a percentaged improvement over `default` of 6.1 % time and 8 % nodes.

We show two profiles for the $n_{\mathcal{P}_3}$- and $t_{\mathcal{P}_3}$-performance for the hard instances only. The first profile in Figure 5.5a shows the dominance of the `combined` setting regarding the phase-3 time. While the lines for the `weights` setting and the `default` are close to each other, we note a slight advantage of the former setting, which never performs much worse than twice a virtual best setting. The graph of the `error based` setting is perceived the slowest in this respect. The second performance profile in Figure 5.5b indicates a superior phase-3 node performance of the `sepa` and `combined` setting, which even outperform `error based`. The remaining two graphs of `dfsHeuroff` and `weights` almost align with the graph of

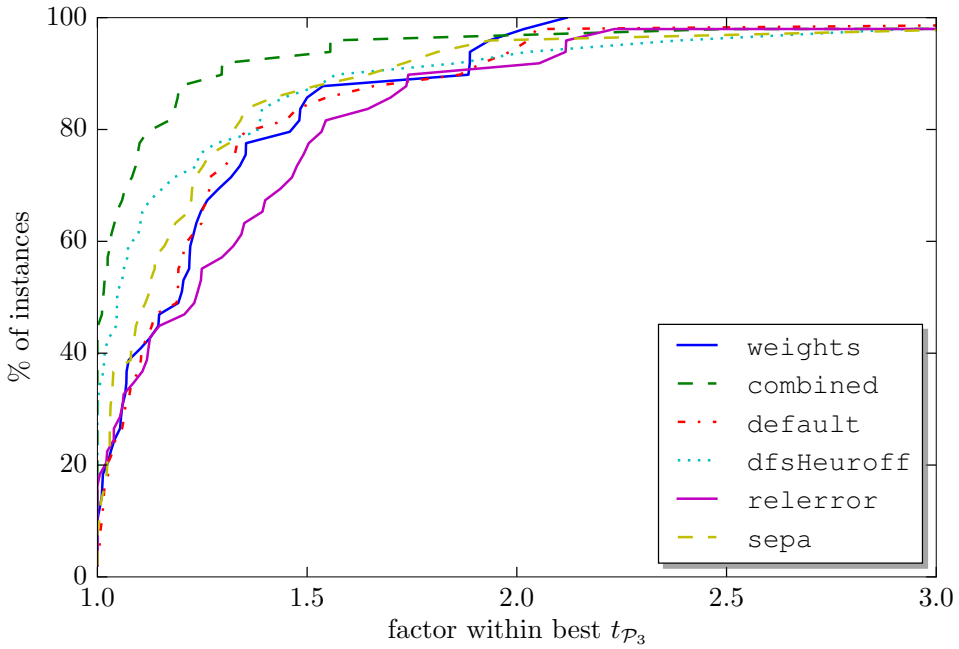
Table 5.9: Revised results on all instances if most extreme outliers regarding shifted quotients are dropped.

(a) Revised results w.r.t. $t_{\mathcal{P}_3}$.

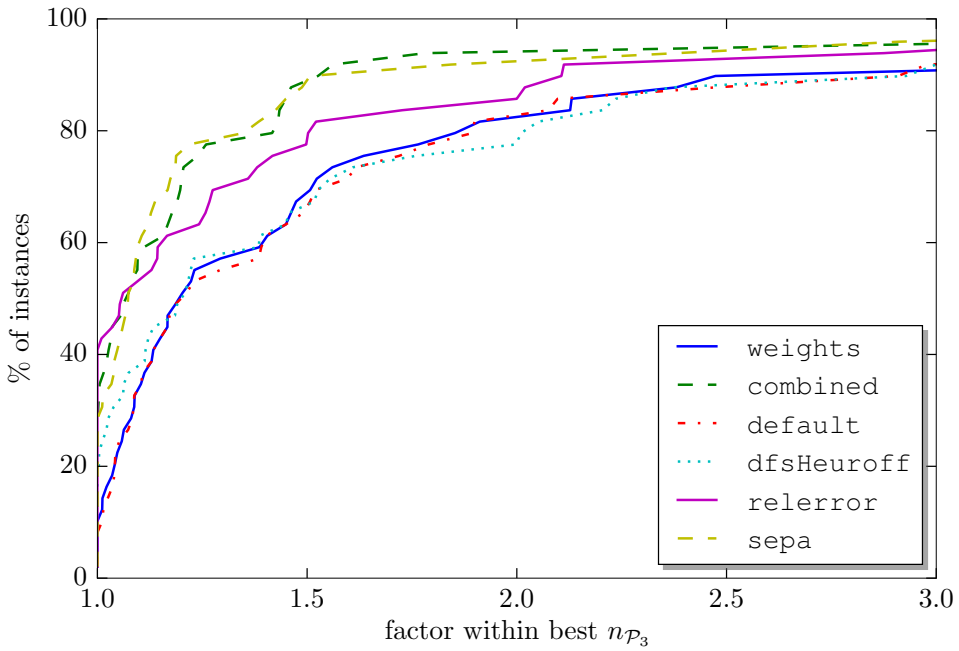
	$t_{\mathcal{P}_3}$	default	%	p	left out	right out
weights	53.2	53.2	100.0	0.1	satellites1-25	reblock67
combined	51.5	54.8	93.9	0.0	satellites1-25	enlight13
dfsHeuroff	51.8	54.8	94.5	0.0	satellites1-25	enlight13
error based	57.2	54.8	104.4	0.0	satellites1-25	enlight13
sepa	52.1	51.7	100.9	0.2	satellites1-25	mspp16

(b) Revised results w.r.t. $n_{\mathcal{P}_3}$.

	$n_{\mathcal{P}_3}$	default	%	p	left out	right out
weights	3019.7	3042.0	99.3	0.5	satellites1-25	eil33-2
combined	2694.5	2932.9	91.9	0.0	satellites1-25	enlight13
dfsHeuroff	2929.1	2932.9	99.9	0.4	satellites1-25	enlight13
error based	2631.0	2962.4	88.8	0.0	modglob	enlight13
sepa	2777.9	3080.2	90.2	0.0	satellites1-25	mspp16



(a) Performance profile for the phase-3 time on hard instances ($\max t > 200$ seconds).



(b) Performance profile for the phase-3 nodes on hard instances ($\max t > 200$ seconds).

Figure 5.5: Performance profile for the phase-3 time and nodes on hard instances ($\max t > 200$ seconds).

`default`.

In this section, we have tested several modifications to SCIP for improving the remaining search performance after an optimal incumbent was detected. The improvement was highest for a setting that combined more aggressive separation, adjusting branching weights, did not use primal heuristics, and explored the tree with depth first search.

The `error based` setting significantly decreases the number of solving nodes, but increases the processing time of each node because it requires more *strong branching* iterations than the default branching rule of SCIP with fixed reliabilty thresholds. By adjusting branching weights alone, we could not observe a significant node reduction.

Note that in this experiment, we focused on the phase transition between the Improvement phase and the Proof phase and modified branching weights only once during the solving process. One may assume that modified weights could even be beneficial throughout the search, independently from the discussed solving phases. A dynamic setting could periodically adjust the weights for the hybrid branching rule depending on the observed frequencies of infeasible and bounded leaves.

5.2 Phase transition

The experiments from the previous section were based on an oracle that could exactly determine the transition between the Improvement phase and the Proof phase. In this section, we investigate heuristic alternatives for this oracle. Three approaches were introduced in Section 4.4. While two of the approaches, the rank-1 criterion and the best-estimate criterion, maintain different global views on the set of all open nodes of the search tree, the last approach summarizes the history of incumbent solutions by means of a logarithmic primal progress.

We ran SCIP with default settings and a time limit of 2h on a test set of 161 instances of our library after excluding the three infeasible instances because they are irrelevant for the phase transition between the Improvement phase and the Proof phase. Besides, we excluded the four instances for which, by the time of this writing, the optimal objective value is unknown, because it is not possible to determine the actual phase transition t_2^*. We tested four criteria: best-estimate, rank-1, and the logarithmic progress factor of the logarithmic primal progress as a function of the number of branch-and-bound nodes and the number of simplex iterations. After every solved branch-and-bound node, we tested if a criterion had been reached, but we waited for the Feasibility phase to be finished before the testing began. When a criterion was met, we recorded the solving time in seconds as heuristic phase transition indicated by this criterion. For the rank-1 and the best-estimate criterion, we also required that at least 50 branch-and-bound nodes were explored. For the progress factor, we required at least three solutions at distinct measurement points, i.e., solutions at three different nodes or solutions that were found after three distinct invocations of the LP-solver. This means in particular that the logarithmic criterion using LP iterations could have

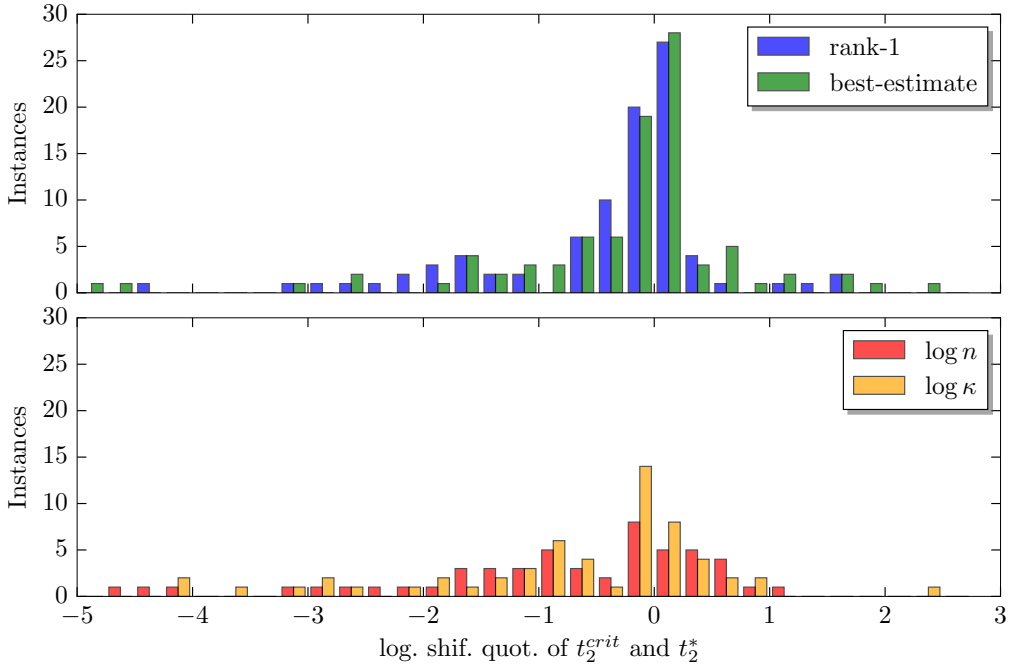


Figure 5.6: The distribution of logarithmic shifted quotients of t_2^{crit} and t_2^* on instances where both values are smaller than 7200 seconds in our test.

already been satisfied at the root node of the branch-and-bound search, whereas the nodes criterion required the search to explore at least three nodes. All required information was collected by means of several event handlers. The implementation was done by the author of the thesis. Instance-wise results are given in Table H.8.

We present Figure 5.6 to compare the true second phase transition t_2^* and the phase transition t_2^{crit} that we recorded for each of the four criteria. The histogram uses a bin width of 0.25. We measure the distance between the two points in time by means of their logarithmic shifted quotient $\log\left((t_2^{\text{crit}}+\tau)/(t_2^*+\tau)\right)$ using a shift of $\tau = 10$ seconds. A nonnegative logarithmic shifted quotient for an instance means that a heuristic phase transition would have been triggered by the criterion after the incumbent solution was truly optimal, i.e., during the Proof phase. A negative logarithmic quotient, however, is encountered for instances where the criterion was met during the Improvement phase. Note that the rank-1 and the best-estimate criterion are trivially met whenever there is no open node left in the tree, i.e. after the search was completed. We do not show such instances in the figure. We see in the figure that the bars for all four criteria are centered about zero, with a tendency to take negative values, i.e. to underestimate the second phase transition. The rank-1 and the best-estimate criteria are met more often, especially in the most important bin right to zero, which accounts for instances where the criterion was met shortly after the optimal solution was found. Note that even if a heuristic phase transition were triggered before t_2^*, the use of settings for the Proof phase

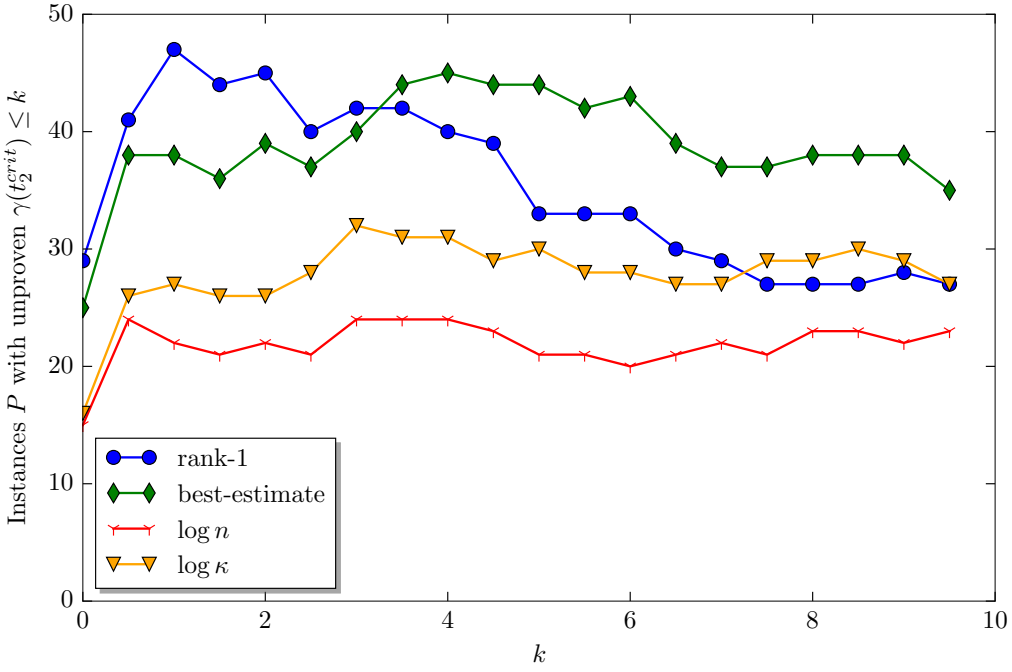


Figure 5.7: The number of instances for which the primal gap $\gamma(t_2^{\text{crit}}) \leq k$, but the primal-dual gap is larger than k, as a function of k.

might still be beneficial if the incumbent objective is close enough to that of an optimal solution. We measure the incumbent quality by means of the primal gap function (cf. (3.1)).

Figure 5.7 shows for all criteria and for $k \in [0, 10]$ in steps of 0.5 the number of instances with a primal gap $\gamma(t_2^{\text{crit}}) \leq k$ by the time the criterion was met, but the primal-dual gap is still larger than k. Recall that the opposite can never be true because the primal-dual gap is an upper bound of the primal gap. In contrast to the last figure, we do not require an optimal solution found within the time limit. The rank-1 curve is the highest for $k \leq 3$, with a peak at $k = 1$, where it is higher than the best-estimate curve by almost 10 instances. The best-estimate curve reaches its peak for $k = 4$ and remains the highest curve for all $3.5 \leq k \leq 10$. The logarithmic criteria have lower curves, which is also due to the fact that they are less often met during the search. The curve for the logarithmic criterion as a function of LP iterations is above the line for the logarithmic nodes criterion for all $0 \leq k \leq 10$ by an almost constant margin of 6–8 instances.

For the instance stp-3d, no incumbent solution is found within two hours, so that none of the criteria is met by definition. Among the remaining 36 instances that hit the time limit with an incumbent, there are 6 for which the incumbent is already optimal. We present in Table 5.10 a contingency table for each criterion and two categories: Whether the criterion was reached within the time limit or not and whether the incumbent solution at termination was already optimal. Note

Table 5.10: Contingency tables that group the 36 time limit instances into four categories whether the criterion was reached, and whether the incumbent at termination is optimal.

(a) Criterion:`estim`

	no	yes	All
reached	9	5	14
not reached	21	1	22
All	30	6	36

(b) Criterion:`rank-1`

	no	yes	All
reached	15	4	19
not reached	15	2	17
All	30	6	36

(c) Criterion:`log-n`

	no	yes	All
reached	23	6	29
not reached	7	0	7
All	30	6	36

(d) Criterion:`log-it`

	no	yes	All
reached	24	6	30
not reached	6	0	6
All	30	6	36

that we measure only if a criterion was met at some point during the search, not if it was met precisely at termination. For the `estim`-criterion, we see that for 5 out of 14 instances for which the criterion was reached, SCIP finds an optimal solution. However, when the criterion is never reached, this is in line with a suboptimal incumbent at termination in 21 out of 22 cases. An exact Fisher test (cf. Section 3.3.3) for this table yields a p-value of 0.02415 under the null hypothesis of independence between the criterion and the optimality of the incumbent. The table for the `rank-1`-criterion does not show a similar result. Here, instances with suboptimal incumbent and with optimal incumbent are spread almost evenly across the two groups for this criterion. The tables for the two logarithmic criteria are very similar. Notably, either of the two criteria was not reached only on instances from our test set for which no optimal solution could be found within the time limit, either. This relation, however, is not indicated to be significant by a Fisher test, which yields p-values of 0.31715 and 0.56102 for the `log-n` and the `log-it` criterion, respectively. The reason is that both criteria were reached too often.

Figure 5.8 shows the primal gap at termination for the 36 instances of the test set that could not be solved to optimality within the time limit. For each criterion, the instances are grouped into two groups depending on whether the criterion was reached until $T = 7200$ seconds or not. The number of instances in each group is shown at the top of the box plot. A sample of gaps is presented by means of a box plot, where the median of each sample is indicated by a black line. The filled boxes range from the 25 % to the 75 %-quartile of the distribution, above and below which a whisker usually indicates the minimum and maximum element of the sample. Outliers beyond the whiskers are indicated by crosses. Samples are considered outliers if they are not contained in the interval $[x_1 - 1.5 \cdot (x_3 - x_1), x_3 + 1.5 \cdot (x_3 - x_1)]$ with x_1 and x_3 being the 25 % and 75%-quartiles, respectively. For each of the

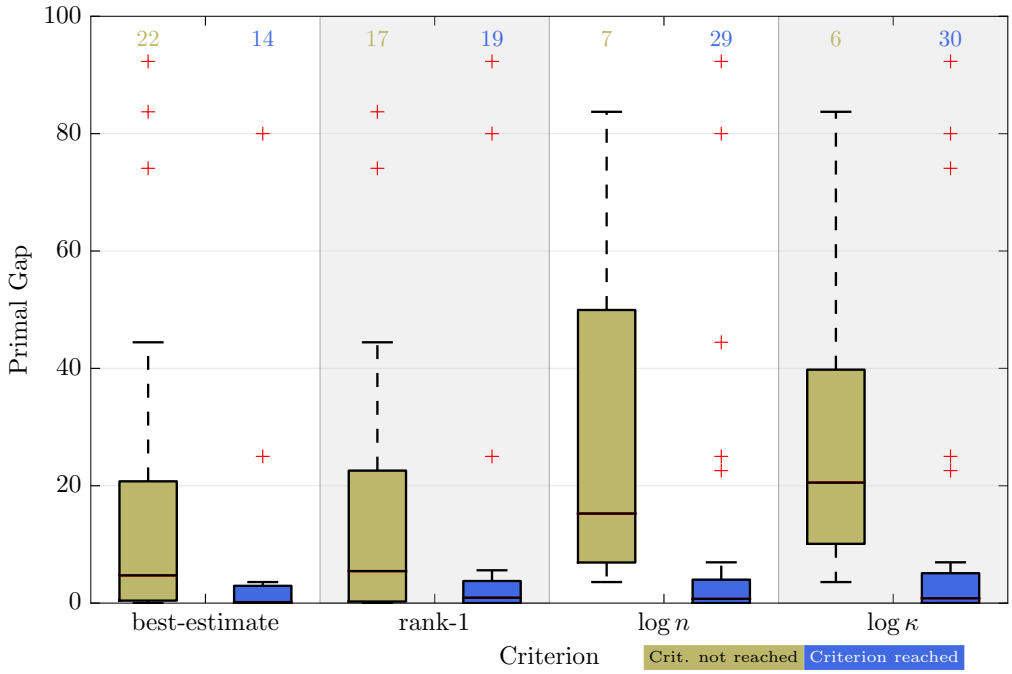


Figure 5.8: Box plots of the primal gap at termination for 36 instances which could not be solved within the time limit.

four criteria, we present two box plots for the groups of instances for which the criterion was not reached (left box) and for which it was reached (right box). For all four criteria, there is a clear tendency of the right group towards zero. At the top of each plot, we show the size of the corresponding group. Despite some of the groups being quite small, we apply a Mann-Whitney-U test. We present the obtained values of the U-statistic and the obtained p-values in Table 5.11. Instances that hit the time limit were grouped into those for which the criterion was met, and those for which the criterion was not met within the time limit. In the table we use abbreviations for the settings: `estim` denotes the best-estimate criterion, `rank-1` the rank-1 criterion, while `log-n` and `log-it` denote the nodes-based and iterations-based logarithmic criteria. We see that the obtained p-values for the criteria `log-it` and `log-n` are below 1 %, and the p-value for `estim` is 0.0148.

We present Figure 5.9 where we grouped the 36 time limit instances into two

Table 5.11: The results of a Mann-Whitney-U test for each criterion.

Setting	`estim`	`rank-1`	`log-n`	`log-it`
$U_{m,n}$	86.5	113.0	31.0	26.0
p	0.0148	0.0640	0.0026	0.0035

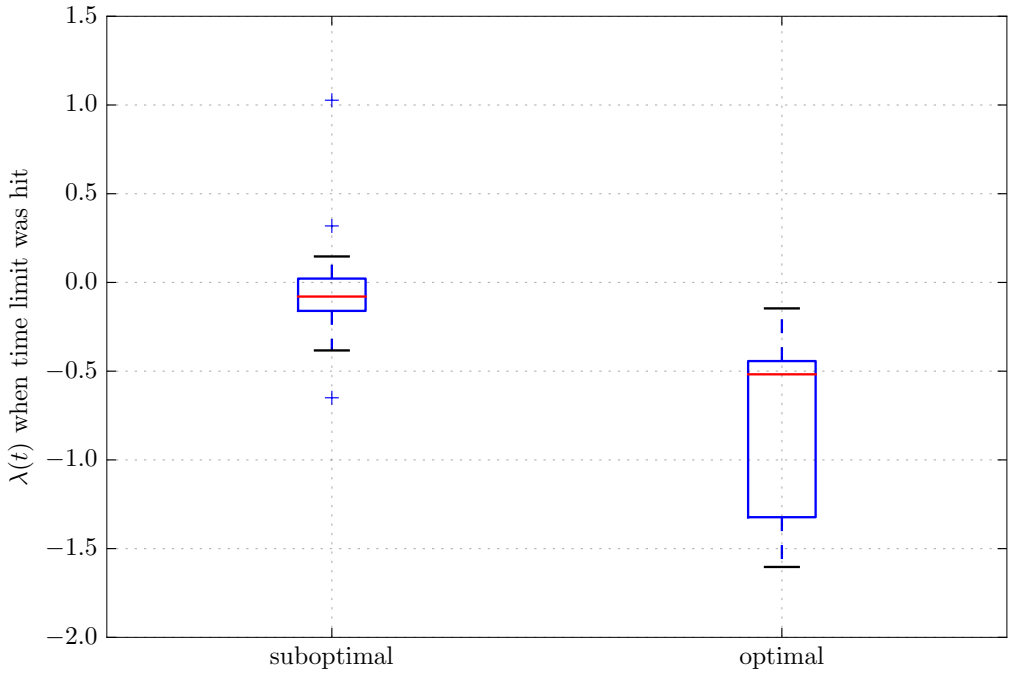


Figure 5.9: Box plots for the logarithmic progress factor at termination for instances with an optimal and suboptimal incumbent at termination.

groups depending on whether an optimal solution for them was found within the time limit or not. For each group, we show a box plot of the progress factor, see Section 4.4.3, at termination. The progress factors are calculated for the logarithmic primal progress regarding time, but similar pictures are obtained for the logarithmic primal progress regarding nodes or iterations. This picture is different from the previous results in that it refers to a specific progress factor at termination. We observe that the progress factor acts on a very small scale between -1.6 and 1.5 across the different MIP instances. This is important for the use of the criterion, which uses a constant threshold of 0.0 for the progress factor for a heuristic phase transition. Besides, the optimal group has a clear tendency towards smaller progress factors, which are all negative. The figure suggests that we could improve the logarithmic criteria by lowering the threshold for the progress factor to, e.g., -0.4.

5.3 Combining the results

In this section, we tie up the loose ends by combining promising settings for each phase and the discussed phase transition strategies. For the Feasibility phase, we use a `uct-rdfs inf` setting, which was the fastest setting in our Feasibility phase experiment that also found feasible solutions for all instances within the time limit of 1h in Section 5.1.1. During the Improvement phase, we employ a setting `l-uct`,

Table 5.12: The number of instances (out of 164) that could be solved to optimality by each setting within two hours.

	default	estim	log-n	log-it	oracle	rank-1
solved	127	129	130	129	130	128

see Section 5.1.2. After a heuristic phase transition, which depends on the choice of the transition criterion, we apply the `combined` setting from Section 5.1.3 for the remainder of the search. As before, by `default`, we denote the default settings of SCIP used throughout all three phases. The setting `oracle` detects the second phase transition exactly. The other settings make use of the proposed heuristic alternatives to the `oracle` as follows: `estim` uses the best-estimate criterion (4.12) for heuristic phase transition, `rank-1` the rank-1 criterion (4.15), whereas the two remaining settings `log-n` and `log-it` keep track of the logarithmic primal history as a function of the number of branch-and-bound nodes and the number of LP-iterations, respectively. The latter settings use the sign of the corresponding progress factor (cf. Definition 12) for phase transition. For all runs, we set a time limit of two hours.

We present the number of instances that were solved to optimality in Table 5.12. For a tabular presentation of the instance-wise outcome, we refer to Table H.9. The `default` setting could solve 127 instances within the time limit. All other settings solved between 1 and 3 instances more in total. The best settings in this respect are the settings `oracle` and `log-n`, which solve 130 instances each. 125 instances were solved by all settings.

The shifted geometric mean solving time is shown in Table 5.13, where we use a shift of $\tau = 10$ seconds. The table shows the results obtained over all 164 instances as well as the obtained results for two groups of 71 easy and 93 hard instances. Apart from the shifted geometric mean solving time, we present the percentage change compared to `default` as well as p-values obtained from a two-sided Wilcoxon signed rank test as described in Section 3.3.4. An instance

Table 5.13: Shifted geometric mean results for t (sec) for all 164 instances and grouped into 93 hard and 71 easy instances.

	all instances			easy ($\max t \leq 200$)			hard ($\max t > 200$)		
	t (sec)	%	p	t (sec)	%	p	t (sec)	%	p
`default`	257.0	100.0		10.4	100.0		1888.7	100.0	
`estim`	245.0	95.3	0.905	10.5	100.9	0.454	1734.4	91.8	0.479
`log-n`	248.8	96.8	0.399	10.4	99.5	0.877	1790.6	94.8	0.349
`log-it`	258.9	100.7	0.919	10.7	102.8	0.439	1891.5	100.1	0.702
`oracle`	242.7	94.4	0.013	11.1	105.9	0.292	1674.7	88.7	0.000
`rank-1`	243.1	94.6	0.008	10.2	97.3	0.357	1736.4	91.9	0.017
`virtual`	221.7	86.3	0.000	9.5	90.6	0.000	1525.3	80.8	0.000

is considered easy if all settings could solve it within 200 seconds, and hard if at least one solver needed more than 200 sec. We also present the results for a virtual solver that always picks the best setting for an instance. Over the entire test set, we observe improvements in the shifted geometric mean solving time for every setting except `log-it` compared to the default settings of SCIP. The best setting is indeed the `oracle`-setting, which is about 14 seconds or 5.6 % faster than `default` in the shifted geometric mean. With the `rank-1` setting, we obtain a similar speed-up of 5.4 %. These two improvements are accompanied by small p-values of 0.013 and 0.008, whereas the improvements shown for `estim` and `log-n` are not significant according to the Wilcoxon test.

On the easy instances, we surprisingly observe the largest increase in the shifted geometric mean solving time for the `oracle` setting, by almost 6 %. While `rank-1` is the fastest amongst the tested settings, the p-values do not reveal any of the settings to be significantly different from `default`. On the hard instances, however, we observe improvements of up to 11.3 % with the `oracle` setting. The setting `estim` improves the shifted geometric mean time by 8.2 % but the corresponding p-value of 0.479 does not identify this improvement as significant. The improvement of the shifted geometric mean for this setting is mainly due to the speed-up on a single instance, acc-tight5, by a factor of 23. This instance is a pure feasibility instance in the sense that the dual bound is already provided by the initial LP relaxation and a feasible solution of this objective needs to be found. Thus, the performance on this instance is greatly affected by our modifications to the settings of SCIP during the Feasibility phase. Recall that we deactivate *strong branching* completely until a first feasible solution is found. The other phase-transition based settings yield the same speed-up for acc-tight5. Yet, an improvement of 8 % in the shifted geometric mean with the `rank-1` setting is accompanied by a p-value of less than 2 %, which indicates that the rank-1 criterion is the better criterion w.r.t. the solving time. The instance is responsible for approximately 3 % improvement in the group of hard instances.

Table 5.14 shows the shifted geometric mean results for all settings regarding the number of branch-and-bound nodes until optimality was proven. As before, we restrict the instances for the node comparison to the subset which could be solved to optimality within the time limit. The table also shows the results for the 71 easy instances and 54 hard instances of this set of instances. For the calculation of the mean, we use a shift of 100 nodes. The `oracle` setting improves the overall shifted geometric mean of the `default` setting by 10 %, which can be split into a 4 %-deterioration on the easy instances and a 20 %-improvement on the hard instances. While the p-value on the easy instances does not indicate a significant effect, it is less than 0.1 % for the set of hard instances and for the overall test set. For the criteria `estim` and `rank-1`, we also observe a reduction of the overall shifted geometric mean of nodes by 6.5 % and 4.6 %, respectively. In this case, however, the p-column does not indicate the latter improvements as significant. The split into easy and hard instances attributes the observed reductions mainly to the hard instances, where the `estim`-setting shows an improvement of more than 12 % compared to `default`.

Table 5.14: Shifted geometric mean results for the number of branch-and-bound nodes n. Results are restricted to 125 instances for which all settings could finish the solve within two hours. The easy and hard groups contain 71 and 54 instances, respectively.

	all instances			easy ($\max t \leq 200$)			hard ($\max t > 200$)		
	n	%	p	n	%	p	n	%	p
`default`	1353.8	100.0		445.6	100.0		19720.2	100.0	
`estim`	1265.2	93.5	0.199	458.1	102.8	0.706	17348.8	88.0	0.031
`log-n`	1408.9	104.1	0.831	462.5	103.8	0.604	21796.6	110.5	0.907
`log-it`	1358.4	100.3	0.689	466.0	104.6	0.942	20213.1	102.5	0.498
`oracle`	1216.9	89.9	0.000	466.3	104.6	0.506	15855.4	80.4	0.000
`rank-1`	1291.7	95.4	0.175	447.2	100.4	0.750	18549.3	94.1	0.148
`virtual`	1058.5	78.2	0.000	401.2	90.0	0.000	14072.8	71.4	0.000

In Table 5.15, we present the shifted geometric mean values of the primal and dual integrals over the set of 161 feasible instances. The `oracle` setting outperforms the `default` settings of SCIP by 6.4 % regarding the primal integral. An even better primal integral is observed for the `estim` setting, which shows an improvement of 7. % over the `default` settings. Both settings using the logarithmic primal progress for phase-transition have higher primal integrals than `default` in the shifted geometric mean, in particular `log-it`, which increases the shifted geometric mean of the primal integral by 8.6 %. Regarding the dual integral $\Gamma^*(T)$, we note an increase in the shifted geometric mean for every phase-transition based setting by up to 19 % with the `log-it` setting. Even the setting `oracle` yields an increase of the shifted geometric mean dual integral by 6 %. The least percentage increase of 2.8 % was observed for the setting `estim`. Furthermore, the p-column shows p-values of less than 0.1 % for every except `estim`.

The results in this section indicate that a phase-based solver indeed outperforms the default settings of SCIP. A phase-based solver that combines beneficial phase-specific settings decreases the shifted geometric mean time and nodes by more than

Table 5.15: Shifted geometric mean results for the primal integral $\Gamma(T)$ and dual integral $\Gamma^*(T)$ over 161 feasible instances. A shift of 1000 was used.

	$\Gamma(T)$	%	p	$\Gamma^*(T)$	%	p
`default`	3692.1	100.0		3891.4	100.0	
`estim`	3430.2	92.9	0.870	3998.7	102.8	0.062
`log-n`	3741.6	101.3	0.435	4376.3	112.5	0.000
`log-it`	4010.4	108.6	0.029	4645.3	119.4	0.000
`oracle`	3456.2	93.6	0.769	4134.1	106.2	0.000
`rank-1`	3689.7	99.9	0.194	4340.4	111.5	0.000
`virtual`	3101.4	84.0	0.000	3736.3	96.0	0.000

5 % and 10 %, respectively. The improvements are mainly observed for instances which we categorized as hard. With this setting, we also solved three additional instances to optimality that could not be solved by the default settings within the time limit.

Using the rank-1 criterion for phase transition, we obtain a solving time improvement that is similar to the improvement obtained with the oracle setting. By employing an `estim`-based phase transition, we observed the highest improvement regarding the number of solving nodes, second to `oracle`. Finally, by using the logarithmic primal progress measured as function of the nodes for phase transition, we could also solve 130 instances to optimality. Comparing the results for an oracle-based phase transition and the heuristic phase-transition criteria that we introduced, we conclude that the `rank-1`-criterion is sufficient in practice to achieve a solving-time performance similar to what can be obtained in principle if we could determine the phase transition exactly. A critical result is the increase of the shifted geometric mean dual integrals, which is an indication that we sacrifice progress on the dual side, most likely by the use of an *inference branching* rule at the beginning of the search.

Chapter 6

IPET–an interactive evaluation tool

Dealing with solver benchmark data is sometimes quite involved. The execution of the software is usually performed via a command line interface. The visual output during the branch-and-bound solve is reduced to sporadic text table information streamed to the console. This compact representation has the advantage of an improved solving performance, compared to the computational overhead from a graphical interface. After the optimization process was terminated, some more information is usually displayed to summarize the solving process.

Existing scripts that come with SCIP already help with parsing and summarizing benchmark log files. However, their customization takes time, especially for self-tailored data other than solving time or nodes. Further, they do not provide an interactive mode. The output generated by these scripts is completely textual. In contrast, the Paver 2.0 [BDV14] tool comes with some benchmark data visualization, such as bar charts for performance differences between settings. Paver also features a solver-independent solution checker. However, Paver requires the data to be prepared in a special file format, which can be generated by the above mentioned SCIP scripts, or by user-written scripts in between.

In order to combine such features and further facilitate benchmark evaluations also for specific data, we developed the IPET *(Interactive Python evaluation tool)*. It is designed to aggregate raw log files in a more interactive fashion. The main features of IPET comprise:

1. a predefined set of readers that already read more than 1000 distinct pieces of data per instance from a SCIP log file. The reader set can quickly be extended to match custom data not covered.

2. Tables for user-specified sub sets of the data, both instance-wise and aggregated.

3. Tools for filtering instances or customized aggregations.

4. Graphical visualizations of the data to look for trends or outliers.

5. A well-tested Python library, which can be easily imported inside command line scripts.

We use the term "ipet" to denote both the Python package and the user interface therein. Whenever we refer to the package, i.e., everything that can be imported into the Python name space through

```
import ipet
```

we write `ipet`. By IPET, we mean the tools provided by the graphical user interface – which are part of `ipet`.

The code is entirely written in Python 2.7, and makes use of the libraries `pandas` [McK12] for data operations and storage, `Tkinter` for the development of the user interface, and `matplotlib` [Hun07] for visualization. Python has been the language of choice for this project for several reasons. First of all, it features a variety of libraries such as the aforementioned `pandas` library or the `matplotlib`, which come with an excellent documentation and can be used free of charge inside projects. Furthermore, Python combines the qualities of a scripting language, such as, e.g., concise string-processing methods, with those of an object-oriented programming language such as, e.g., Java or C++, which facilitates the work on a user interface.

Although the focus of this section is on presenting an overview of the graphical user interface, we will also give command line recipes for important tasks. Such recipes can be used inside user-written scripts that import the `ipet` library. For the data acquisition and evaluation for the computational experiments in this thesis, we used customized scripts based on the `ipet`-functionalities and the cited libraries.

6.1 Overview of the library

The `ipet` library consists of two main parts: a back end, which is responsible for data acquisition, data storage, and file IO, and a front end, which consists of the necessary components to interact with the back end data through a user interface. The central back end controller class is called Comparator, and resides in a module of the same name. In order to acquire SCIP benchmark data, we have to feed our program with SCIP output in the form of *log files*. See Section 6.4 for the specifications of an expected log file that can be read by the Comparator. It is safe to rely on the format of ".out"-files that are generated by SCIP when running the `make test` command.

An object of the Comparator class comprises an arbitrary number of TestRun instances, which are associated to a (list of) SCIP log files. Data are read by means of StatisticReader instances. A newly constructed Comparator instance comes with a large variety of available readers for parsing the solving time and number of branch-and-bound nodes, but also more advanced readers that read in complete tables, or so-called *histories*; the latter combine column output from the periodic SCIP status table, such as the development of the primal bound as a function of the solving time. It is also possible to add additional readers and reevaluate the log file information.

After the collection of the data has been finished, the acquired data for each TestRun is stored as *DataFrame* object of the ***pandas*** *library* [McK12]. Data frames allow for fast data manipulations such as grouping data, transformations, aggregations, and combinations of them. The data can also be exported to several formats such as, e.g., LaTeX, Microsoft Excel ".xls"-format, or ".csv"-files.

The Comparator further provides filters and aggregations to reduce the data to an interesting subset or aggregate numeric data into dense results tables in order to prove the success of their tested method(s).

6.2 Installation and prerequisites

This section covers the installation of Ipet. We assume that you have a working Python 2.7 installation. On top of that, we require

- `matplotlib` version 1.3.1 or higher
- `Tkinter` with Tcl/Tk version 8.5 or higher
- `pandas` version 0.12.0 or higher
- for some distributions, the *Pillow* library as a replacement for the *PIL* library.

A zipped folder of the `ipet`-package will be made available upon request from the author, but we are planning to soon make it publicly available. After extracting the file into a directory "some/dir/name" of your choice, open a terminal or command line, and change to the directory "some/dir/name". In order to install the `ipet`-module and its contents to the python path and thus make it importable from every working directory you are in, run the command:

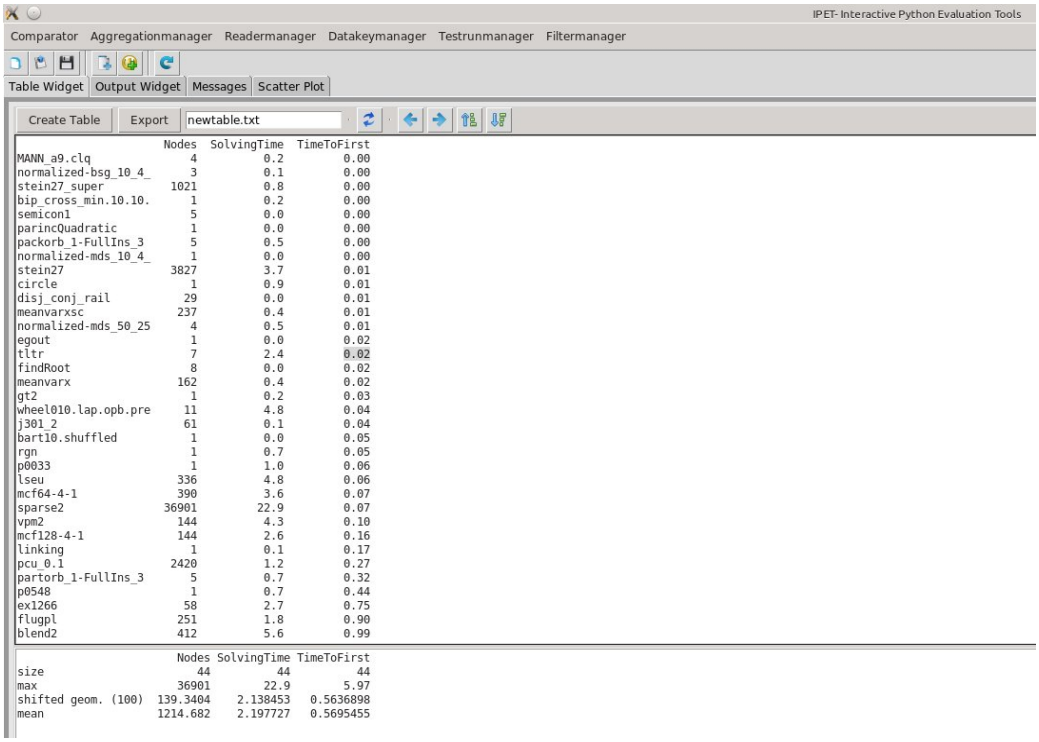
```
python setup.py install [--user]
```

This will install the `ipet`-package to your Python distribution, and make it importable from every working directory in the file system. If you do not have administrator rights on your system, append `--user` in order to install the package to your local Python libraries. If you now start an interactive python interpreter from any working directory, you can import `ipet` as a module into your Python scripts and use its methods. The user interface uses the *Python Image Library (PIL)* to display button images via PNG files. In case you have an error that says PIL could not be found, install *Pillow* via

```
pip install Pillow [--user]
```

or

```
python easy_install Pillow [--user]
```



	Nodes	SolvingTime	TimeToFirst
MANN_a9.clq	4	0.2	0.00
normalized-bsg_10_4_	3	0.1	0.00
stein27_super	1021	0.8	0.00
bip_cross_min.10.10.	1	0.2	0.00
semicon1	5	0.0	0.00
parincQuadratic	1	0.0	0.00
packorb_1-FullIns_3	5	0.5	0.00
normalized-mds_10_4_	1	0.0	0.00
stein27	3827	3.7	0.01
circle	1	0.9	0.01
disj_conj_rail	29	0.0	0.01
meanvarxsc	237	0.4	0.01
normalized-mds_50_25	4	0.5	0.01
egout	1	0.0	0.02
tltr	7	2.4	0.02
findRoot	8	0.0	0.02
meanvarx	162	0.4	0.02
gt2	1	0.2	0.03
wheel010.lap.opb.pre	11	4.8	0.04
j301_2	61	0.1	0.04
bart10.shuffled	1	0.0	0.05
rgn	1	0.7	0.05
p0033	1	1.0	0.06
lseu	336	4.8	0.06
mcf64-4-1	390	3.6	0.07
sparse2	36901	22.9	0.07
vpm2	144	4.3	0.10
mcf128-4-1	144	2.6	0.16
linking	1	0.1	0.17
pcu_0.1	2420	1.2	0.27
partorb_1-FullIns_3	5	0.7	0.32
p0548	1	0.7	0.44
ex1266	58	2.7	0.75
flugpl	251	1.8	0.90
blend2	412	5.6	0.99

	Nodes	SolvingTime	TimeToFirst
size	44	44	44
max	36901	22.9	5.97
shifted geom. (100)	139.3404	2.138453	0.5636898
mean	1214.682	2.197727	0.5695455

Figure 6.1: A screenshot of the Table widget.

6.3 Starting the IPET user interface

It is now possible to run the start-up script `startipet.py` for the user interface, which is located in the `scripts` subdirectory of the `ipet`-directory, from every working directory. Under Windows, double-click on the icon "startipet" in the `scripts` subdirectory of the `ipet` home directory.

After the application has started, IPET starts with focus on the (initially empty) table widget. The main area of the IPET lets the user switch between four sub-windows, so-called *widgets*:

- a *Table widget* which is open after starting the IPET to present an arbitrary subset of the acquired data instancewise, and aggregated.
- an *Output widget*, which lets the user browse the log file output of a particular instance and define additional readers.
- a *Scatter plot* widget to scatter two arbitrary data columns and search for trends or outliers.
- a *Message widget* to display the past messages from the IPET.

The IPET lets the user browse back and forth between the widgets by clicking on their register tab. The IPET also features a menu and a navigation panel at the top, as well as a status bar at the bottom. Some of the widgets feature an

```
@01 instances/CP/linking.cip ===========
...
<Additional SCIP output>
...
=ready=
```

Figure 6.2: Minimum sample log file for correct instance recognition by `ipet` readers. The first line indicates the beginning of the log output for an instance The last line indicates that the output for this instance is finished. Every parsed input in between is saved for the index `linking`, where the directory path and the file extension are dropped for more readable indices.

additional navigation panel, which allows for widget-specific interaction with the data.

6.4 Reading log files

By the term log file, we denote an ASCII-encoded file that contains the command line output by SCIP when it is invoked for a (set of) MIP instances. If multiple MIP instances are present in the log file, we expect every output for a single instance as a consecutive portion of the log-file, i.e., outputs for different instances are not mixed. The output for each instance should be preceded a unique starting expression, `@01`, followed by a whitespace and a unique identifier for this instance. If the identifier is the file name of this instance, both a preceding path name to the file and the file extension will be dropped to yield a shorter identifier. In order to guarantee the correct match of instances to their data, we need make sure that every instance is represented by a unique *file name*. The expression `=ready=` should be contained after every instance to indicate that all information for this instance is now available. A sample log file format is shown in Figure 6.2.

In the most common use case, a single log file contains the output of SCIP with a single setting on a set of test instances. Different log files then differ in the setting they apply, but the instance set remains the same for all tested settings. Although IPET allows for different settings on each instance within a single log file, it will be identified by the setting used for the first instance.

The easiest way to generate log-files that match all specifications is to use the test scripts that are already provided by SCIP. Specify a test set as a file "Instances.test" with the extension ".test" that contains the paths to the instances so that SCIP can find them, and run `make test TEST=Instances` from the SCIP home directory. This will create a file in the sub-directory "check/results" of the SCIP home directory with the extension ".out".

While it is possible to link a TestRun to arbitrarily many log files through the library methods, see also Recipe 6.1, the user interface currently supports only one log file per TestRun. In order to do so, click on the `Add log files`-button in the navigation panel, or choose "Add log file(s)" from the `Comparator`-menu. It is

```
# use '#' for comments
#<modifier> <instance> [<optimal or best known primal bound>]
# e.g.
=opt=    linking         -15.00
=inf=    infeas_1
#...
```

Figure 6.3: File format for `.solu`-file to contain additional solution information.

possible to select multiple log files from the file browser. Each selected file will be associated with a newly created TestRun instance.

```
from ipet.Comparator import Comparator
from ipet.TestRun import TestRun

comp = Comparator()
tr = TestRun()

comp.addLogFile("path/to/logfile1_1.out", tr)
comp.addLogFile("path/to/logfile1_2.out", tr)
# tr now has two associated log files which
# will be opened in the order they were added
```

Recipe 6.1: Passing a TestRun instance as optional argument to `addLogFile` allows to add multiple log files to this test run.

It is also possible to associate log file information to all test runs. This is particularly helpful for passing, e.g., information about optimal solution values for the tested instances, if such information is available. Such information can then be used by the Comparator to verify the outcome of the tests and warn the user if a particular outcome contradicts the instance information. The easiest way to achieve this is to pass such information as *solution file* to the Comparator. Figure 6.3 shows a sample file, which could be directly passed to a Comparator. Every line should contain a modifier indicating the current knowledge about the optimum objective of this instance: Possible modifiers are `=inf=` for infeasible instances, `=opt=` for instances whose optimal objective value is known, `=best=` for instances for which an optimal solution value is not known, or the optimality of the given primal bound was not proven. The modifier is followed by the instance identifier (cf. sample log file 6.2). If the modifier is not `=inf=`, a primal bound needs to be given as reference value. Solution files have `.solu` as preferred file extension. Solution files passed to the Comparator will be appended to the file list of each TestRun instance when the data collection is invoked. Users can pass one or several solution files via the `Add Solu File(s)`-button or the corresponding entry in the `Comparator`-menu. The command line method is `addSoluFile()`.

After all desired log files have been added to the Comparator, data acquisition can be triggered via the `Collect data`-button in the top navigation panel, or via the corresponding entry in the `Comparator`-menu. In a script, call the

Table 6.1: The most important data keys.

Key	Description
`SolvingTime`	The total solving time in seconds for an instance as reported by SCIP.
`Nodes`	The total number of explored branch-and-bound nodes.
`TimeToFirst`	The time in seconds until a feasible solution was found.
`Status`	`ok` if solver correctly solved the instance, `TimeLimit`, `MemoryLimit`, or `NodeLimit`, if the solving process reached one of those limits, `abort`, if solving process prematurely exited, maybe due to a bug in the solver, or `fail`, if the solver terminated with a result that is not in line with the solution file information.
`GitHash`	The git hash that was used to solve this instance.
`Settings`	The settings file that was used to solve this instance.
`DateTime_Start`	The date and time when a solving process was started.
`DateTime_End`	The date and time when the solving process was finished.

`collectData()`-method of a Comparator object. The log files for each TestRun are traversed in the order they were appended to it. If an instance identifier appears more than once, only the *last* log file output will be taken into account for the statistics by overriding previous information. An exception is solution information, which is naturally passed separately from the actual output. If data acquisition is invoked through the user interface, its progress is indicated as a progress bar at the bottom of the IPET.

During the data acquisition, each reader reads the specified log files line by line. Every line is checked for a possible match with a (set of) regular expression pattern(s) of this reader. From a matching line, it parses one or several pieces of data from that line and adds them to the internal data storage for the current TestRun under a specific data key for the current instance identifier. Some of these data keys are presented in Table 6.3. Similar data keys come in groups such as, e.g., the execution times of primal heuristics. All execution times of primal heuristics that are displayed in the log file statistics are parsed and stored under a unique data key of the form `HeurTime_<heuristic name>`. The underscore in the data key indicates to the IPET a data key group. Groups can be nested as, e.g., the number of separated cutting planes through the linear constraint handler of SCIP is stored as `Constraints_cuts_linear`.

After the data collection has been finished, it is possible to display instance-wise tables of the desired data pieces. For more on displaying tabular data, see Section 6.5.1. If an experiment requires the acquisition of custom data that are not provided by the default readers, additional readers can be specified through the user interface via the Output widget. For details, see Section 6.5.2.

6.5 Widgets of the Ipet

In the `Tkinter` toolkit of Python, *Widget* is the general term for all graphical components of a graphical user interface. We adapt this term for the IPET, where *Widgets* are windows inside the interface, which provide the necessary controls to perform a distinct task of data visualization. In particular, widgets work independently from the current preference settings other widgets. Currently, the IPET features four widgets for data visualization. The most important widget is certainly the *Table widget*, which can be used for creating custom table representations of the data.

6.5.1 Table widget

The Table widget is the default focus widget after starting IPET. In order to make use of this widget, it is necessary to collect some log file data first. The acquisition of data is described in detail in Section 6.4. A new table can be created through a configuration window, which is opened via the `Create Table`-button in the navigation panel at the top of this widget. The configuration window presents all acquired data keys in a tree-like hierarchy. The hierarchy groups similar data keys together. Groups need to be expanded to access the underlying data keys, and can be collapsed afterwards.

In order to populate a table with data, select one or multiple data keys as well as the TestRun instances which should be added to the table. The selection of multiple entries is possible through holding the Shift or Ctrl-key while clicking on an entry. Use the `Add current selection`-button to add the current selection of test runs and data keys to the table. The Table widget is instantaneously populated by a tabular representation of the requested data in two windows. The top window shows an instance-wise outcome of the results, where one column is devoted to each possible combination of requested data keys and test runs. Each row represents the results for a single instance. As row index, the instance identifier, which is used to store the instance results in the internal data storage, is shown. It is possible to truncate very long identifiers by modifying the `index width`-parameter, which is accessible through the `Options`-menu of the Table widget. Entries for which no data are stored for the combination (instance, data key) are represented by `NaN`-values (`NaN` stands for "**N**ot **a** **N**umber").

The second table shows aggregated results for each column of the table. The currently displayed aggregations are the arithmetic mean and the shifted geometric mean (cf. Section 3.2) of a column as well as its minimum entry, maximum entry, and size. The size denotes the number of entries in this column including `NaN`-entries. At the presence of `NaN`-entries in a column, certain aggregations yield `NaN` as well. This also holds if a column is nonnumeric such as, e.g., the columns for the data keys DateTime_Start and DateTime_End.

A column can be selected by a double click into the column area. A triple click on an entry highlights the corresponding row and column of that entry. For a selected column, a context menu is opened with the right mouse button. The

Table 6.2: File extensions for data export currently supported by the Table widget.

Name	Extension	Description
Text table	`.txt`	Saves a text representation of the current table.
Latex table	`.tex`	Converts and saves the table as a LaTeX-table.
CSV table	`.csv`	Saves the current table in CSV-format.
Excel table[1]	`.xls`, `.xlsx`	Exports the table directly into a spreadsheet, which is readable by, e.g., Microsoft Excel and Open Office.

context menu shows a list of column operations that can be applied to the table. The menu allows to sort the table w.r.t. the specified column or w.r.t. the index. Furthermore, the sub-menu `Data Plots` allows for graphical visualizations of the selected column. By selecting one of the entries from the `Data Plots`-menu, a quick wizard opens if additional information is required for a plot. A scatter plot, e.g., asks the user to select a second column from a drop down menu, which will be used as y-coordinate for the scatter points. For a histogram, the width of the bins needs to be given.

The table shows the acquired data for each instance that was not filtered by one of the active filters in the Filtermanager of the Comparator instance. Filters can be modified via the `Filtermanager`-menu in the menu bar of the IPET. For more about filters, we refer to the corresponding paragraph in Section 6.6.

The data can be exported to several file formats. Use the entry field in the navigation panel of the Table widget to specify a file name and extension, and use the `Export`-button to convert and save the table to a specified format. The format is specified by the file extension. All supported file extensions are presented in Table 6.2. If the file extension is omitted or an unknown file extension is used, the table will be exported in textual format to this file.

6.5.2 Output widget

The *Output widget* displays the log file output for a single instance. The output serves two purposes. First, it is the basis for many evaluations. If a newly tested setting worked very well (poorly) for a particular instance, looking at the SCIP periodic status line and statistics can give insight which component of SCIP may have caused this impact. The Output widget recognizes the periodic status line of SCIP and the statistics after the solving process was finished. It allows to toggle either of them to concentrate on the remaining part and reduce the necessary amount of scrolling.

The second intention behind the Output widget is to facilitate the creation of customized readers for data acquisition. If a log-file contains numeric information

[1] This option requires some additional Python packages and should be considered experimental. If it does not work well on a particular Python distribution, it is safe to use the CSV-export functionality instead. CSV files can be imported to both Microsoft Excel and Open Office.

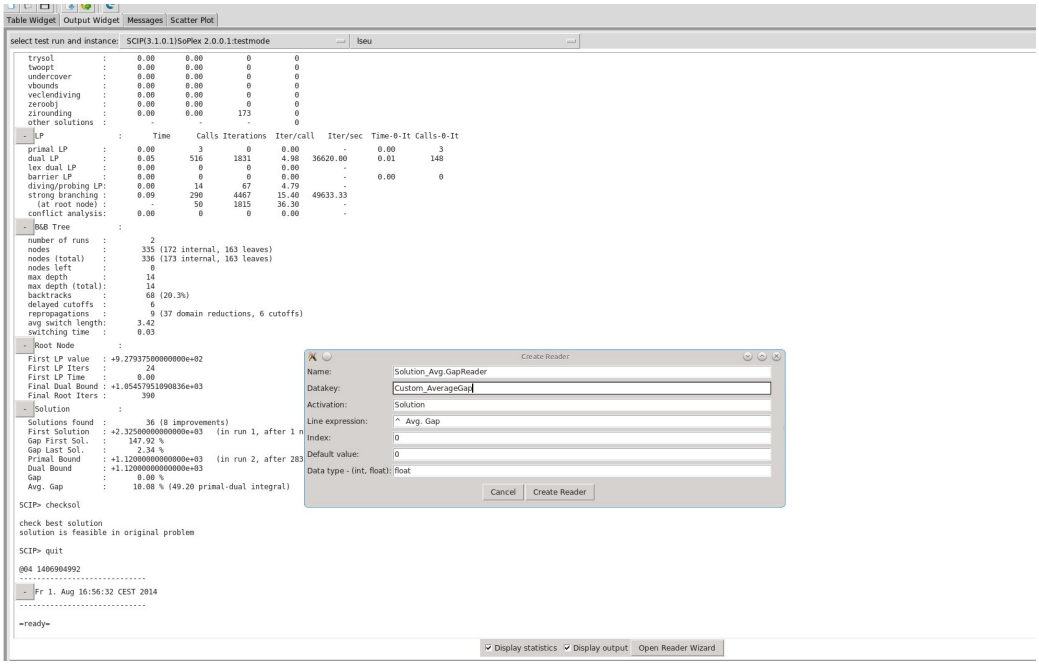


Figure 6.4: The Reader wizard called for the creation of a customized reader.

about the instances that is not parsed by any of the available readers, approach a number with the mouse and double click on it. This will highlight the selected number and a form sheet for creating a new reader for the selected data, the *Reader Wizard*, can be opened via the `Open Reader Wizard`-button. In many cases, it is sufficient to click on the `Create Reader`-button to create a CustomReader instance that is applied in subsequent data collections to parse the selected piece of data. Sometimes, it might be desirable to rename the data key suggestion, or the reader name before creating the reader.

The data key suggestion of the wizard contains an underscore. Recall that underscores indicate that a data key belongs to a group, and are later accessible inside this group in the tree-hierarchy for data key selection. It is perfectly admissible to use `Custom_` or another suitable group prefix instead of the suggestion to fold data that were read by CustomReaders under a single item of the tree-hierarchy. The number of interest is characterized through its *index*, which specifies the position of this particular number among all numbers that are found in a line which matches the line pattern. The index of the first number in the line is `0`. Numbers are integer or floating point values contained in the line. Both floating point and scientific notation (by means of an exponent) is admissible.

The recognition of the data element is subject to the matching of two patterns: a *line pattern* and an *activation pattern*. While a simple word or the empty string is sufficient to define a pattern, regular expression syntax is also supported to allow for more complex patterns. The line pattern is used for the recognition of the line in the log file that contains the specified piece of data. The line pattern

```
Original Problem   :
  Problem name     : ...
  Variables        : 5 (3 binary, 2 integer, 0 implicit integer, ...
  ...
Presolved Problem  :
  Problem name     : ...
  Variables        : 4 (3 binary, 1 integer, 0 implicit integer, ...
  ...
```

Figure 6.5: A typical situation where an activation pattern is necessary for a clear distinction.

```
time | node  | left  |LP iter|...|cuts |confs|strbr|  dual...
 0.1s|     1 |     0 |    96 |...|   0 |   0 |   0 | 7.163...
  ...
```

Figure 6.6: A sample of the periodic status line of SCIP.

should therefore be precise enough to rule out as many conflicting lines as possible. However, it should not contain instance-specific information, which is not matched on other instances.

Sometimes, it is not possible to rule out all conflicting lines of a SCIP log file by a single line pattern alone, see, e.g., Figure 6.5. The output sample contains two different lines that start with `Variables`. Since their nonnumeric content is identical, an activation pattern such as, e.g., `Original Problem`, should be used to identify the correct line from which the number should be parsed.

Thus, the reader is inactive at the beginning of the data acquisition, and after an instance output was finished. While a reader is inactive, it scans the lines for its activation pattern. If a line contains the activation pattern, the reader becomes active. Only active readers search for possible matches of their line pattern to parse the desired data. After the line pattern was matched, the reader becomes inactive until another line matches the activation pattern. Whenever the line pattern suffices for a unique line recognition, the activation pattern can be left empty.

With this rule in mind, it is clear how to treat information that occurs repeatedly in the output for a single instance. If an activation pattern is used which precedes the first occurrence, only the first occurrence of the line pattern will be used for data acquisition if the activation pattern is unique. A unique activation pattern is `@01`, in which case the reader is active from the beginning. If the activation pattern is instead left empty, the reader stores the specified number from the latest line which was matched by the line pattern.

Besides the CustomReader-class the `ipet`-package features another customizable reader class, namely a CustomHistoryReader. By the term *history*, we denote the common evolution of two or more quantities during the course of the solving process, as shown in the periodic status line of SCIP. An example of a history is

the evolution of the primal bound over time. This information is already provided as data key `PrimalBoundHistory`. A sample output of the periodic status line of SCIP is shown in Figure 6.6. One usually combines a monotonous measure such as the time, number of nodes, or number of LP iterations, and one or more pieces of additional data provided by the status line such as, e.g., the number of open subproblems in the tree, or the number of conflict clauses generated. Recipe 6.2 shows how a CustomHistoryReader can be added to a Comparator. In this recipe, we are interested in the primal bound as a function of the number of nodes. All it takes to create the reader is a specification of the relevant header tags of the status line, in this case `node` and `primalbound`, which are passed to the constructor as a list.

```
from ipet.Comparator import Comparator
from ipet.StatisticReader_CustomHistoryReader import \
     CustomHistoryReader

comparator = Comparator()
reader = CustomHistoryReader(
                              listofheaders=['node', 'primalbound']
                             )
comparator.addReader(reader)
```

Recipe 6.2: Recipe to add a CustomHistoryReader to a Comparator

6.5.3 Scatter widget

The *Scatter widget* provides a visualization of the acquired data as scatter plot. We need to select the TestRun instances and data keys for the x- and y-axis of the plot. When both TestRuns and data keys are selected, a scatter plot appears in the window. Besides the scattered points, which represent instances and have as x- and y-coordinate the values of the selected TestRuns and data keys, respectively. Instances for which one or both values are missing, are not shown in the plot. Besides the scatter plot, a diagonal is drawn to facilitate the search for trends in the plot. If an outlier or otherwise interesting point is localized, a click with the left mouse button shows the instance to which the selected point corresponds. If several points are close to each other, the information is displayed for all of them.

The *Tool bar* at the bottom of the widget allows for zooming or panning the visible area of the plot. The different views obtained through zooming or panning can be undone and redone. Note that the points of the scatter plot are only clickable if the selection-mode is active. It is also possible to resize the plot area, edit axis and curve properties, or save the plot to a file. The list of available file formats is long and includes JPEG, PDF, SVG etc. For a complete list, see the `File type`-option in the Save dialog. For a list of available options in the Tool bar, see also Table 6.3.

Table 6.3: Available tools from the *Tool bar* at the bottom of each plot window.

Position	Name	Function
1	`Reset`	undo all previous actions in pan or zoom-mode.
2	`Back`	undo last change in pan or zoom-mode.
3	`Forward`	redo last change in pan or zoom-mode.
4	`Pan`	toggle pan-mode. In pan-mode, you can move the center of the visible plot area or zoom.
5	`Zoom`	toggle zoom-mode. In zoom-mode, you can specify a rectangle to zoom into. Draw the rectangle with the left mouse button to zoom in, and with the right mouse button to zoom out.
6	`Configure`	resizing options for the plot area w.r.t. the surrounding frame
7	`Save`	Save the figure to a file on disk.
8	`Edit`	Open a dialog to edit general properties of the plot area as well as properties of plotted elements such as curves or scattered points.

6.5.4 Further plot widgets

Graphical visualizations of the data are either provided by the stand-alone Scatter widget, see Section 6.5.3, or accessible through the context menu of the Table widget as `Data plots`. The latter take the selected column of the displayed table as input. If additional input is needed before the plot can be drawn, a dialog is shown to set the required arguments. If a scatter plot should be drawn, the selected column data defines the x-values of the scatter points. The column for the y-values needs to be selected (or confirmed) in the dialog.

The dialog only offers columns which are in the table. The `New Plot` check box can be used to make the resulting plot appear in a new window, or if the desired plot should be added to the last plot window. The latter option is only possible, if the last created plot window was not closed yet.

A histogram of the selected data is also available. As before, selected columns can either be appended to the last created plot or drawn in a fresh window. When columns are added to a histogram, the available horizontal space for every bin is equally distributed among the columns for which a histogram is shown. The bin width can be specified either as a fixed number of bins or via the bin width. If the bin number is positive, it gets precedence over a bin width specification. Furthermore, the range for the bins needs to be specified.

The Ipet remembers the bin configuration so that every further column can be added without the need to adjust the bin parameters.

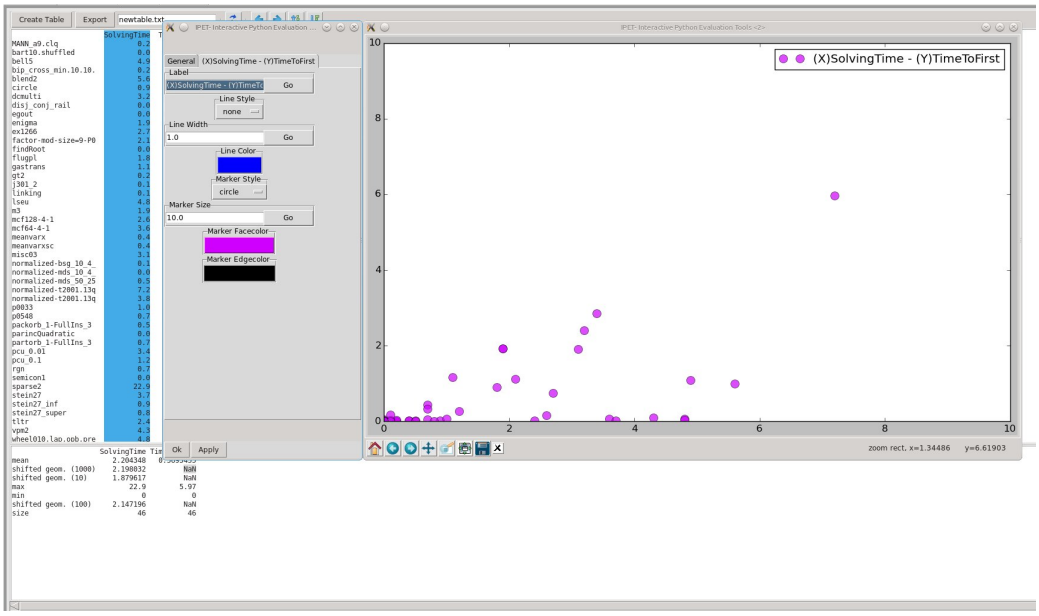


Figure 6.7: Example of a scatter plot for two columns of the Table widget.

6.5.5 Message widget

The *Message widget* displays the past message stream. Normal messages are displayed in black, while error messages are displayed in red. This feature is mainly used for debugging purposes.

6.6 Filters and aggregations

Filters and aggregations have already been mentioned before. Filters can be used to exclude instances which do not satisfy a certain criterion. One may wish to view a table only for instances that hit the time limit, instances that were solved to optimality within at most 10 seconds, instances that required one of the present TestRuns at least 50 branch-and-bound nodes to solve, or the intersection of the latter two subsets of instances. The Comparator provides a set of filters, which can be edited through the *Filter browser* of the IPET. The Filter browser shows all filters in a list. One or multiple filters can be selected. The current selection can be (de)-activated. Deactivated filters are highlighted by a red background color. Every filter is displayed through a name that already explains its use. An active filter of the name "any SolvingTime $\geq$ 100" only accepts instances for which at least one of the present TestRuns needed at least 100 seconds to solve. All other instances are dropped and will therefore not be displayed in created tables and aggregated statistics or data visualizations until the filter is deactivated.

A filter can also be edited through the Filter browser. A double click on a filter name opens a form sheet to edit the filter attributes. (Un)-check the left box to indicate whether filter condition should hold for at least one TestRun, or

all of them. The filter condition itself consists of three fields: Two expressions and a binary operator in between. If you are done editing an entry, click on the Go-button to refresh the reader. You can see that its name changes if your entry was successful. As expression, data keys as well as integer, floating point, or string expressions are accepted. The filter examples above could be realized as follows:

- instances for which all TestRuns hit the time limit:

```
(any/all) (Expr 1)   (Op)  (Expr 2)
   all     Status     ==   TimeLimit
```

- instances that were solved to optimality within at most 10 seconds:

```
all     SolvingTime   <=   10
```

- instances for which one TestRun needed at least 50 nodes:

```
any     Nodes   >=   50
```

Aggregations follow the same convention. They can be browsed by their name like filters. Only active aggregations are displayed in the aggregated table results. Their names can be edited through the browser. Only for shifted geometric means, it is currently possible to change the behaviour of the aggregation by editing the shift value.

6.7 Outlook

The IPET in its current version provides basic functionalities to work with MIP benchmark data that come in the form of raw log files. The intention behind both the user interface and the underlying `ipet`-package was an easy-to-use interface to common MIP benchmark evaluation tasks. For special tasks/evaluations that it is not capable of, the IPET facilitates the export of the benchmark data to formats readable by many available general data evaluation software products. We benefit from the use of the Python language and its highly sophisticated libraries such as the `pandas` or `matplotlib` libraries, which provide most of the required functionalities for data storage, export, and visualization. The `ipet`-package separates data acquisition from the both data evaluation and data visualization and export. It is therefore suitable for the inclusion inside custom evaluation scripts, which in turn benefit from an improved readability and maintainability.

Besides an obvious inclusion of more options, visualizations, filters, etc., there exist many interesting possibilities for a future development of the package and user interface such as, e.g., the following:

- an extension of the parsing of log files of solvers other than SCIP, which can be directly influenced through the user interface. Currently, the detection of a solver used for a log file is limited to the following solvers: SCIP, CPLEX, GUROBI, XPRESS, CBC. Only six readers such as, e.g., the SolvingTimeReader and the PrimalBoundHistoryReader can be used for all of them, while the majority of readers will proceed with their SCIP setup.

- The possibility of user-defined transformations to the data, and an application of statistical hypothesis tests.

- An automatic generation of command line scripts to repeat an evaluation with other log files in an almost automatic fashion. This is partly possible through the Comparator object, by removing all TestRuns and saving the resulting Comparator object. All readers, filters, aggregations are maintained and can be reused.

- Another future possibility includes the development of a benchmark library that the user interface can be connected with. Interesting benchmark results could be compared to past results stored in such a library.

The IPET is currently in a beta-stadium, but will be made available upon request from the author of this thesis.

Chapter 7

Summary

In this thesis, we partitioned the branch-and-bound solving process into three solving phases and developed a MIP solver that adapts to the current solving phase. Three main contributions for working with solving phases are presented in this thesis:

i. an improved use of MIP solver components for the individual phase objectives with speed-ups of more than 50 % for the Feasibility phase,

ii. computational results that show an improved *overall* performance for a phase-based solver that combines phase-specific settings for each phase, in particular an improved running-time of 11 % over a test set of instances that we classified as hard, as well as

iii. the introduction and evaluation of three possible criteria for a heuristic phase transition. Switching to settings for the Proof phase guided by one of these criteria, we could observe a similar overall speed-up as for the phase-based solver that can exactly determine the phase transitions.

The phase-specific settings, which emphasized different component classes after every phase transition, can certainly be further improved, especially for the second phase, for which we could not find a setting that could outperform the running time of SCIP with default settings significantly. Apart from a concrete realization inside a phase-based solver, the view on the overall optimization process as a set of phases emphasizing different objectives helps in understanding the actual impact of the different component classes on the solving process inside MIP solvers such as SCIP. Such an understanding is necessary for the development of new components and a more dynamic use of existing ones.

During the course of this thesis, we extended SCIP by several components. We implemented a branching rule together with a diving heuristic, a Large Neighborhood Search heuristic, and a node selection rule from the recent MIP literature. Furthermore, we developed two modifications to the hybrid reliability pseudo-cost/inference branching rule. All components will be made available as default

plug-ins for the SCIP Optimization Suite. Furthermore, we developed the `ipet`-package for the work with MIP benchmark data. It is written in the Python programming language and comes with a graphical user interface.

For practical applications, it is often sufficient to require only a feasible solution to a problem, or one that is good enough, whenever a proof of optimality needs to be sacrificed due to time or memory restrictions. The phase experiments in this thesis may serve as a guideline to the practitioner how to make better use of SCIP for a particular purpose.

Bibliography

[AA95] Erling D. Andersen and Knut D. Andersen. Presolving in linear programming. *Mathematical programming*, 71:221–245, 1995.

[ABCC95] David L. Applegate, Robert E. Bixby, Vasek Chvátal, and William J. Cook. Finding cuts in the TSP (A preliminary report). Technical Report 95-05, DIMACS, 1995.

[ABH12] Tobias Achterberg, Timo Berthold, and Gregor Hendel. Rounding and propagation heuristics for mixed integer programming. In Diethard Klatte, Hans-Jakob Lüthi, and Karl Schmedders, editors, *Operations Research Proceedings 2011*, pages 71–76. Springer Berlin Heidelberg, 2012.

[Ach07] Tobias Achterberg. *Constraint Integer Programming.* PhD thesis, Technische Universität Berlin, 2007.

[Ach09] Tobias Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[ADL06] Belarmino Adenso-Diaz and Manuel Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1):99–114, 2006.

[AKM04] Tobias Achterberg, Thorsten Koch, and Alexander Martin. Branching rules revisited. *Operations Research Letters*, 33(1):42–54, 2004.

[AKM06] Tobias Achterberg, Thorsten Koch, and Alexander Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):1–12, 2006.

[BCMS98] Robert E. Bixby, Sebastiàn Ceria, Cassandra M. McZeal, and Martin W.P. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, 58:12–15, 1998.

[BDV14] Michael R. Bussieck, Steven P. Dirkse, and Stefan Vigerske. Paver 2.0: An open source environment for automated performance analysis of benchmarking data. *Journal of Global Optimization*, 59(2-3):259–275, 2014.

[Ber06] Timo Berthold. Primal heuristics for mixed integer programs. Diploma thesis, Technische Universität Berlin, 2006.

[BGG+71] Michel Bénichou, Jean-Michel Gauthier, Paul Girodet, Gerard Hentges, Gerard Ribière, and O. Vincent. Experiments in mixed-integer programming. *Mathematical Programming*, 1:76–94, 1971.

[BGS14] Timo Berthold, Gerald Gamrath, and Domenico Salvagnin. Cloud branching. Presentation slides from Mixed Integer Programming Workshop at Ohio State University. `https://mip2014.engineering.osu.edu/sites/mip2014.engineering.osu.edu/files/uploads/Berthold_MIP2014_Cloud.pdf`, 2014.

[BMW75] A.L. Brearley, G. Mitra, and H.P. Williams. Analysis of mathematical programming problems prior to applying the simplex algorithm. *Mathematical Programming*, 8:54–83, 1975.

[BT08] Dimitri P. Bertsekas and John N. Tsitsiklis. *Introduction to Probability*. Athena Scientific, 2nd edition, 2008.

[cbc] COIN-OR branch-and-cut MIP solver. `https://projects.coin-or.org/Cbc`.

[CDM78] Harlan P. Crowder, Ron S. Dembo, and John M. Mulvey. Reporting computational experiments in mathematical programming. *Mathematical Programming*, 15(1):316–329, 1978.

[Che92] Pang C. Chen. Heuristic sampling: A method for predicting the performance of tree searching programs. *SIAM Journal on Computing*, 21(2):295–315, 1992.

[Chv73] Vasek Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4(4):305–337, 1973.

[CKL06] Gérard Cornuéjols, Miroslav Karamanov, and Yanjun Li. Early estimates of the size of branch-and-bound trees. *INFORMS Journal on Computing*, 18(1):86–96, 2006.

[Coh95] Paul R. Cohen. *Empirical Methods for Artificial Intelligence*. MIT Press, Cambridge, MA, USA, 1995.

[cpl] IBM ILOG CPLEX Optimizer. `http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/`.

[CS00] Marie Coffin and Matthew J. Saltzman. Statistical analysis of computational tests of algorithms and heuristics. *INFORMS Journal on Computing*, 12(1):24–44, 2000.

[Dan08] Emilie Danna. Performance variability in mixed integer programming. Presentation slides from MIP workshop in New York City. `http://coral.ie.lehigh.edu/~jeff/mip-2008/program.pdf`, 2008.

[FBM03] Michael Falk, Rainer Becker, and Frank Marohn. *Angewandte Statistik: Eine Einführung mit Programmbeispielen in SAS*. Springer-Verlag GmbH, 2003.

[fic] FICO Xpress-Optimizer. `http://www.fico.com/en/Products/DMTools/xpress-overview/Pages/Xpress-Optimizer.aspx`.

[FL03] Matteo Fischetti and Andrea Lodi. Local branching. *Mathematical Programming*, 98(1-3):23–47, 2003.

[FL10] Matteo Fischetti and Andrea Lodi. Heuristics in mixed integer programming. In James J. Cochran, Louis A. Cox, Pinar Keskinocak, Jeffrey P. Kharoufeh, and J. Cole Smith, editors, *Wiley Encyclopedia of Operations Research and Management Science*. John Wiley & Sons, Inc., 2010. Online publication.

[FM14] Matteo Fischetti and Michele Monaci. Proximity search for 0-1 mixed-integer convex programming. Technical report, DEI - Università di Padova, 2014.

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1979.

[Gom58] Ralph E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[gur] GUROBI Optimizer. `http://www.gurobi.com/products/gurobi-optimizer/gurobi-overview`.

[Hen11] Gregor Hendel. New rounding and propagation heuristics for mixed integer programming. Bachelor thesis, 2011.

[HHLBS09] Frank Hutter, Holger H. Hoos, Kevin Leyton-Brown, and Thomas Stützle. ParamILS: An automatic algorithm configuration framework. *Journal of Artificial Intelligence Research*, 36(1):267–306, 2009.

[Hoo94] John Hooker. Needed: An empirical science of algorithms. *Operations Research*, 42(2):201–212, 1994.

[Hoo95] John Hooker. Testing heuristics: We have it all wrong. *Journal of Heuristics*, 1(1):33–42, 1995.

[Hun07] John D. Hunter. Matplotlib: A 2D graphics environment. *Computing In Science & Engineering*, 9(3):90–95, 2007.

[JP82] Ellis L. Johnson and Manfred W. Padberg. Degree two inequalities, clique facets, and biperfect graphs. *Annals of Discrete Mathematics*, 16:169–187, 1982.

[KAA+11] Thorsten Koch, Tobias Achterberg, Erling Andersen, Oliver Bastert, Timo Berthold, Robert E. Bixby, Emilie Danna, Gerald Gamrath, Ambros M. Gleixner, Stefan Heinz, Andrea Lodi, Hans Mittelmann, Ted Ralphs, Domenico Salvagnin, Daniel E. Steffy, and Kati Wolter. MIPLIB 2010. *Mathematical Programming Computation*, 3(2):103–163, 2011.

[Kha79] Leonid G. Khachiyan. A polynomial algorithm in linear programming. *Doklady Akademii Nauk SSSR*, 244(5):1093–1096, 1979. english translation in Soviet Math. Dokl. 20(1):191–194, 1979.

[KMP13] Thorsten Koch, Alexander Martin, and Marc E. Pfetsch. Progress in academic computational integer programming. In Michael Jünger and Gerhard Reinelt, editors, *Facets of Combinatorial Optimization*, pages 483–506. Springer, 2013.

[Knu74] Donald E. Knuth. Estimating the efficiency of backtrack programs. Technical report, Stanford University, Stanford, CA, USA, 1974.

[KW52] William H. Kruskal and W. Allen Wallis. Use of Ranks in One-Criterion Variance Analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952.

[LS97] Jeff T. Linderoth and Martin W. P. Savelsbergh. A computational study of search strategies for mixed integer programming. *INFORMS Journal on Computing*, 11:173–187, 1997.

[McG96] Catherine C. McGeoch. Toward an experimental method for algorithm simulation. *INFORMS Journal on Computing*, 8(1):1–15, 1996.

[McK12] Wes McKinney. *Python for Data Analysis: Data Wrangling with Pandas, NumPy, and IPython.* O'Reilly Media, 2012.

[McN47] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

[PC11] Jennifer Pryor and John W. Chinneck. Faster integer-feasibility in mixed-integer linear programs by branching to force change. *Computers & Operations Research*, 38(8):1143 – 1152, 2011.

[PQ08] Gilles Pesant and Claude-Guy Quimper. Counting solutions of knapsack constraints. In Laurent Perron and Michael A. Trick, editors, *CPAIOR*, volume 5015 of *Lecture Notes in Computer Science*, pages 203–217. Springer, 2008.

[SAB+12] Yuji Shinano, Tobias Achterberg, Timo Berthold, Stefan Heinz, and Thorsten Koch. ParaSCIP – a parallel extension of SCIP. In Christian Bischof, Heinz-Gerd Hegering, Wolfgang E. Nagel, and Gabriel Wittum, editors, *Competence in High Performance Computing 2010*, pages 135–148. Springer, 2012.

[Sav94] Martin W. P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *ORSA Journal on Computing*, 6:445–454, 1994.

[sci] SCIP. Solving Constraint Integer Programs. `http://scip.zib.de/`.

[Ser08] Robert J. Serfling. *Approximation Theorems of Mathematical Statistics*. John Wiley & Sons, Inc., 2008.

[sop] SoPlex. An open source LP solver implementing the revised simplex algorithm. `http://soplex.zib.de/`.

[SSR12] Ashish Sabharwal, Horst Samulowitz, and Chandra Reddy. Guiding combinatorial optimization with UCT. In Nicolas Beldiceanu, Narendra Jussien, and Eric Pinson, editors, *CPAIOR*, volume 7298 of *Lecture Notes in Computer Science*, pages 356–361. Springer, 2012.

[Wol06] Kati Wolter. Implementation of cutting plane separators for mixed integer programs. Diploma thesis, Technische Universität Berlin, 2006.

Chapter H

Appendix

H.1 Special settings files

This section covers settings files used for the experiments which influence many parameters at once. Parameter values are only shown for parameters that are different from the default parameter value.

H.1.1 The setting `agg`

The following settings are used inside an aggressive heuristic setting `agg`. It is also the basis for the two settings $\texttt{agg}_{05}$ and `agg l-uct`. The subset of the parameter values affecting only diving heuristics is `aggrdive`, and the subset that affects LNS-heuristics is `l-agg`.

```
# SCIP version 3.1.0.1

# frequency for calling primal heuristic <actconsdiving> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/actconsdiving/freq = 20

# frequency for calling primal heuristic <clique> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/clique/freq = 20

# frequency for calling primal heuristic <coefdiving> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: 10]
heuristics/coefdiving/freq = 5

# maximal fraction of diving LP iterations compared to node LP iterations
# [type: real, range: [0,1.797693134862232e+308], default: 0.05]
heuristics/coefdiving/maxlpiterquot = 0.075

# additional number of allowed LP iterations
# [type: int, range: [0,2147483647], default: 1000]
heuristics/coefdiving/maxlpiterofs = 1500

# frequency for calling primal heuristic <crossover> (-1: never, 0: only at depth
    ↪  freqofs)
# [type: int, range: [-1,2147483647], default: 30]
heuristics/crossover/freq = 15

# number of nodes without incumbent change that heuristic should wait
# [type: longint, range: [0,9223372036854775807], default: 200]
```

```
heuristics/crossover/nwaitingnodes = 20

# contingent of sub problem nodes in relation to the number of nodes of the
    ↪ original problem
# [type: real, range: [0,1], default: 0.1]
heuristics/crossover/nodesquot = 0.15

# minimum percentage of integer variables that have to be fixed
# [type: real, range: [0,1], default: 0.666]
heuristics/crossover/minfixingrate = 0.5

# should the nwaitingnodes parameter be ignored at the root node?
# [type: bool, range: {TRUE,FALSE}, default: FALSE]
heuristics/crossover/dontwaitatroot = TRUE

# frequency for calling primal heuristic <dins> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/dins/freq = 20

# frequency for calling primal heuristic <distributiondiving> (-1: never, 0: only
    ↪  at depth freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/distributiondiving/freq = 20

# frequency for calling primal heuristic <feaspump> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: 20]
heuristics/feaspump/freq = 10

# maximal fraction of diving LP iterations compared to node LP iterations
# [type: real, range: [0,1.797693134862320e+308], default: 0.01]
heuristics/feaspump/maxlpiterquot = 0.015

# additional number of allowed LP iterations
# [type: int, range: [0,2147483647], default: 1000]
heuristics/feaspump/maxlpiterofs = 1500

# frequency for calling primal heuristic <fixandinfer> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/fixandinfer/freq = 20

# frequency for calling primal heuristic <fracdiving> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: 10]
heuristics/fracdiving/freq = 5

# maximal fraction of diving LP iterations compared to node LP iterations
# [type: real, range: [0,1.797693134862320e+308], default: 0.05]
heuristics/fracdiving/maxlpiterquot = 0.075

# additional number of allowed LP iterations
# [type: int, range: [0,2147483647], default: 1000]
heuristics/fracdiving/maxlpiterofs = 1500

# frequency for calling primal heuristic <guideddiving> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: 10]
heuristics/guideddiving/freq = 5

# maximal fraction of diving LP iterations compared to node LP iterations
# [type: real, range: [0,1.797693134862320e+308], default: 0.05]
heuristics/guideddiving/maxlpiterquot = 0.075

# additional number of allowed LP iterations
# [type: int, range: [0,2147483647], default: 1000]
heuristics/guideddiving/maxlpiterofs = 1500

# frequency for calling primal heuristic <zeroobj> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/zeroobj/freq = 20

# frequency for calling primal heuristic <intdiving> (-1: never, 0: only at depth
    ↪  freqofs)
```

```
# [type: int, range: [-1,2147483647], default: -1]
heuristics/intdiving/freq = 20

# frequency for calling primal heuristic <intshifting> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: 10]
heuristics/intshifting/freq = 5

# frequency for calling primal heuristic <linesearchdiving> (-1: never, 0: only
    ↪ at depth freqofs)
# [type: int, range: [-1,2147483647], default: 10]
heuristics/linesearchdiving/freq = 5

# maximal fraction of diving LP iterations compared to node LP iterations
# [type: real, range: [0,1.79769313486232e+308], default: 0.05]
heuristics/linesearchdiving/maxlpiterquot = 0.075

# additional number of allowed LP iterations
# [type: int, range: [0,2147483647], default: 1000]
heuristics/linesearchdiving/maxlpiterofs = 1500

# frequency for calling primal heuristic <localbranching> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/localbranching/freq = 20

# frequency for calling primal heuristic <nlpdiving> (-1: never, 0: only at depth
    ↪  freqofs)
# [type: int, range: [-1,2147483647], default: 10]
heuristics/nlpdiving/freq = 5

# frequency for calling primal heuristic <mutation> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/mutation/freq = 20

# frequency for calling primal heuristic <objpscostdiving> (-1: never, 0: only at
    ↪  depth freqofs)
# [type: int, range: [-1,2147483647], default: 20]
heuristics/objpscostdiving/freq = 10

# maximal fraction of diving LP iterations compared to total iteration number
# [type: real, range: [0,1], default: 0.01]
heuristics/objpscostdiving/maxlpiterquot = 0.015

# additional number of allowed LP iterations
# [type: int, range: [0,2147483647], default: 1000]
heuristics/objpscostdiving/maxlpiterofs = 1500

# frequency for calling primal heuristic <octane> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/octane/freq = 20

# frequency for calling primal heuristic <proximity> (-1: never, 0: only at depth
    ↪  freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/proximity/freq = 20

# frequency for calling primal heuristic <pscostdiving> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: 10]
heuristics/pscostdiving/freq = 5

# maximal fraction of diving LP iterations compared to node LP iterations
# [type: real, range: [0,1.79769313486232e+308], default: 0.05]
heuristics/pscostdiving/maxlpiterquot = 0.075

# additional number of allowed LP iterations
# [type: int, range: [0,2147483647], default: 1000]
heuristics/pscostdiving/maxlpiterofs = 1500

# frequency for calling primal heuristic <randrounding> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: 20]
heuristics/randrounding/freq = 10
```

```
# frequency for calling primal heuristic <rens> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: 0]
heuristics/rens/freq = 20

# minimum percentage of integer variables that have to be fixable
# [type: real, range: [0,1], default: 0.5]
heuristics/rens/minfixingrate = 0.3

# number of nodes added to the contingent of the total nodes
# [type: longint, range: [0,9223372036854775807], default: 500]
heuristics/rens/nodesofs = 2000

# frequency for calling primal heuristic <rins> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: 25]
heuristics/rins/freq = 13

# frequency for calling primal heuristic <rootsoldiving> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: 20]
heuristics/rootsoldiving/freq = 10

# maximal fraction of diving LP iterations compared to node LP iterations
# [type: real, range: [0,1.797693134862326e+308], default: 0.01]
heuristics/rootsoldiving/maxlpiterquot = 0.015

# additional number of allowed LP iterations
# [type: int, range: [0,2147483647], default: 1000]
heuristics/rootsoldiving/maxlpiterofs = 1500

# frequency for calling primal heuristic <shiftandpropagate> (-1: never, 0: only
    ↪ at depth freqofs)
# [type: int, range: [-1,2147483647], default: 0]
heuristics/shiftandpropagate/freq = 20

# frequency for calling primal heuristic <shifting> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: 10]
heuristics/shifting/freq = 5

# frequency for calling primal heuristic <trivial> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: 0]
heuristics/trivial/freq = 20

# frequency for calling primal heuristic <twoopt> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/twoopt/freq = 20

# frequency for calling primal heuristic <undercover> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: 0]
heuristics/undercover/freq = 20

# frequency for calling primal heuristic <vbounds> (-1: never, 0: only at depth
    ↪ freqofs)
# [type: int, range: [-1,2147483647], default: -1]
heuristics/vbounds/freq = 20

# frequency for calling primal heuristic <veclendiving> (-1: never, 0: only at
    ↪ depth freqofs)
# [type: int, range: [-1,2147483647], default: 10]
heuristics/veclendiving/freq = 5

# maximal fraction of diving LP iterations compared to node LP iterations
# [type: real, range: [0,1.797693134862326e+308], default: 0.05]
heuristics/veclendiving/maxlpiterquot = 0.075

# additional number of allowed LP iterations
# [type: int, range: [0,2147483647], default: 1000]
heuristics/veclendiving/maxlpiterofs = 1500
```

H.1.2 The setting sepa

The following non-default parameters were used inside an aggressive cutting plane separation setting:

```
# SCIP version 3.1.0.1

# minimal orthogonality for a cut to enter the LP in the root node
# [type: real, range: [0,1], default: 0.5]
separating/minorthoroot = 0.1

# maximal number of separation rounds in the root node of a subsequent run (-1:
    ↪ unlimited)
# [type: int, range: [-1,2147483647], default: 1]
separating/maxroundsrootsubrun = 5

# maximal additional number of separation rounds in subsequent price-and-cut
    ↪ loops (-1: no additional restriction)
# [type: int, range: [-1,2147483647], default: 1]
separating/maxaddrounds = 5

# maximal number of separated cuts at the root node (0: disable root node
    ↪ separation)
# [type: int, range: [0,2147483647], default: 2000]
separating/maxcutsroot = 5000

# separation frequency for the global cut pool (-1: disable global cut pool, 0:
    ↪ only separate pool at the root)
# [type: int, range: [-1,2147483647], default: 0]
separating/poolfreq = 10

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
constraints/linear/sepafreq = 10

# maximal number of cuts separated per separation round in the root node
# [type: int, range: [0,2147483647], default: 200]
constraints/linear/maxsepacutsroot = 500

# should all constraints be subject to cardinality cut generation instead of only
    ↪ the ones with non-zero dual value?
# [type: bool, range: {TRUE,FALSE}, default: FALSE]
constraints/linear/separateall = TRUE

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
constraints/bounddisjunction/sepafreq = 0

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
constraints/conjunction/sepafreq = 0

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
constraints/countsols/sepafreq = 0

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
constraints/disjunction/sepafreq = 0

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
constraints/integral/sepafreq = 0

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
constraints/knapsack/sepafreq = 10

# maximal number of cuts separated per separation round in the root node
# [type: int, range: [0,2147483647], default: 200]
constraints/knapsack/maxsepacutsroot = 500

# frequency for separating cuts (-1: never, 0: only in root node)
```

```
# [type: int, range: [-1,2147483647], default: 0]
constraints/logicor/sepafreq = 10

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
constraints/or/sepafreq = 10

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
constraints/pseudoboolean/sepafreq = 0

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
constraints/setppc/sepafreq = 10

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
constraints/SOS1/sepafreq = 10

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
constraints/SOS2/sepafreq = 10

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
constraints/superindicator/sepafreq = 0

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
constraints/varbound/sepafreq = 10

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
constraints/xor/sepafreq = 10

# frequency for separating cuts (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
constraints/linprojection/sepafreq = 0

# frequency for calling separator <clique> (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
separating/clique/freq = 20

# frequency for calling separator <closecuts> (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
separating/closecuts/freq = 0

# frequency for calling separator <cmir> (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
separating/cmir/freq = 20

# maximal number of cmir separation rounds in the root node (-1: unlimited)
# [type: int, range: [-1,2147483647], default: 10]
separating/cmir/maxroundsroot = 15

# maximal number of consecutive unsuccessful aggregation tries in the root node
    ↪ (-1: unlimited)
# [type: int, range: [-1,2147483647], default: 100]
separating/cmir/maxfailsroot = 200

# maximal number of cmir cuts separated per separation round in the root node
# [type: int, range: [0,2147483647], default: 500]
separating/cmir/maxsepacutsroot = 1000

# frequency for calling separator <flowcover> (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
separating/flowcover/freq = 20

# maximal number of separation rounds in the root node (-1: unlimited)
# [type: int, range: [-1,2147483647], default: 15]
separating/flowcover/maxroundsroot = 22

# maximal number of flow cover cuts separated per separation round in the root
# [type: int, range: [0,2147483647], default: 200]
separating/flowcover/maxsepacutsroot = 400
```

```
# frequency for calling separator <gomory> (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
separating/gomory/freq = 20

# maximal number of gomory separation rounds in the root node (-1: unlimited)
# [type: int, range: [-1,2147483647], default: 10]
separating/gomory/maxroundsroot = 15

# maximal number of gomory cuts separated per separation round in the root node
# [type: int, range: [0,2147483647], default: 200]
separating/gomory/maxsepacutsroot = 400

# frequency for calling separator <impliedbounds> (-1: never, 0: only in root
    ↪ node)
# [type: int, range: [-1,2147483647], default: 0]
separating/impliedbounds/freq = 20

# frequency for calling separator <mcf> (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
separating/mcf/freq = 20

# maximal number of different deltas to try (-1: unlimited) -- default
    ↪ separation
# [type: int, range: [-1,2147483647], default: 20]
separating/mcf/maxtestdelta = -1

# should negative values also be tested in scaling?
# [type: bool, range: {TRUE,FALSE}, default: FALSE]
separating/mcf/trynegscaling = TRUE

# maximal number of mcf cuts separated per separation round in the root node --
    ↪ default separation
# [type: int, range: [-1,2147483647], default: 200]
separating/mcf/maxsepacutsroot = 400

# frequency for calling separator <oddcycle> (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
separating/oddcycle/freq = 0

# maximal number of oddcycle cuts separated per separation round in the root node
# [type: int, range: [0,2147483647], default: 5000]
separating/oddcycle/maxsepacutsroot = 10000

# maximal number of oddcycle separation rounds in the root node (-1: unlimited)
# [type: int, range: [-1,2147483647], default: 10]
separating/oddcycle/maxroundsroot = 15

# frequency for calling separator <rapidlearning> (-1: never, 0: only in root
    ↪ node)
# [type: int, range: [-1,2147483647], default: -1]
separating/rapidlearning/freq = 0

# frequency for calling separator <strongcg> (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: 0]
separating/strongcg/freq = 20

# maximal number of strong CG separation rounds in the root node (-1: unlimited)
# [type: int, range: [-1,2147483647], default: 20]
separating/strongcg/maxroundsroot = 30

# maximal number of strong CG cuts separated per separation round in the root
    ↪ node
# [type: int, range: [0,2147483647], default: 500]
separating/strongcg/maxsepacutsroot = 1000

# frequency for calling separator <zerohalf> (-1: never, 0: only in root node)
# [type: int, range: [-1,2147483647], default: -1]
separating/zerohalf/freq = 0

# maximal number of zerohalf separation rounds in the root node (-1: unlimited)
# [type: int, range: [-1,2147483647], default: 10]
separating/zerohalf/maxroundsroot = 15

# maximal number of {0,1/2}-cuts separated per separation round in the root node
# [type: int, range: [0,2147483647], default: 500]
separating/zerohalf/maxsepacutsroot = 1000
```

H.2 Experimental results

The tables of this section contain the complete experimental data used in this thesis.

Table H.1: The additional time $t_{>0}$ after the branch and bound search started.

	act&dist	aggrdive	br-rdfs inf	default	dfs inf	inf	rdfs inf	uct inf	uct-rdfs inf
10teams	10.90	11.00	3.40	12.20	0.10	4.20	0.10	6.90	4.10
acc-tight5	385.30	672.50	108.90	1408.20	431.10	402.60	393.00	196.70	53.10
arki001	27.00	6.10	3.10	1.70	14.50	22.90	1.80	17.20	2.70
atlanta-ip	492.80	422.60	169.40	524.90	122.90	740.40	241.20	98.00	97.90
bab5	110.00	109.80	80.50	121.70	7.00	88.00	7.00	202.80	84.80
bnatt350	469.30	646.80	677.30	1458.10	790.60	551.60	634.90	280.40	810.50
csched010	93.10	111.10	44.30	63.30	3593.80	436.50	11.80	54.40	24.00
danoint	5.80	5.90	0.00	6.10	53.40	0.10	1.60	0.20	0.20
dcmulti	0.20	0.20	0.10	0.10	0.10	0.00	0.10	0.10	0.10
enigma	0.70	0.60	0.20	0.50	0.10	0.40	0.00	0.20	0.60
flugpl	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
l152lav	0.40	0.50	0.10	0.70	0.10	0.00	0.10	0.20	0.20
lectsched-4-obj	16.60	16.60	18.00	16.80	6.30	16.50	3.60	18.30	12.80
misc03	0.00	0.10	0.10	0.00	0.10	0.10	0.10	0.00	0.10
misc07	0.10	0.20	0.10	0.10	0.10	0.10	0.10	0.00	0.10
momentum2	2206.80	889.70	426.40	358.70	3547.20	1265.40	433.40	90.60	246.90
msc98-ip	2956.00	1411.70	260.00	2920.80	59.90	88.60	178.30	3115.10	509.60
neos-1109824	1.70	1.70	3.90	1.70	0.10	0.10	0.10	4.40	4.20
neos-1337307	77.70	78.90	39.60	53.80	4.10	35.00	4.30	32.20	36.60
neos-1601936	287.20	288.10	247.70	274.40	85.40	200.10	161.20	205.50	205.70
neos-686190	16.70	17.30	3.10	27.50	3598.20	6.90	12.10	1.40	1.80
neos-849702	426.20	214.30	1028.60	166.30	2054.90	394.60	207.90	480.70	556.30
net12	93.40	93.70	6.40	842.90	72.20	6.40	93.80	6.40	6.50
ns1208400	1735.30	1641.90	373.50	1781.20	3179.70	3580.10	495.80	937.10	250.50
ns1688347	337.10	331.30	27.10	171.40	10.20	13.50	6.10	25.10	25.80
ns1830653	7.80	7.60	6.00	7.60	0.90	14.30	0.90	8.90	6.70
rd-rplusc-21	438.30	438.70	174.10	437.30	219.20	3540.80	3541.30	239.20	776.10
rocII-4-11	16.70	17.10	9.40	17.30	0.90	7.00	0.90	8.40	7.70
timtab1	0.30	0.30	0.30	0.30	0.10	0.30	0.10	0.20	0.10
timtab2	5.50	8.80	0.90	10.40	0.80	1.30	0.30	1.60	0.90
triptim1	937.60	934.80	246.80	1148.70	142.10	380.10	140.30	185.00	188.50
vpphard	227.00	269.60	112.80	172.90	32.60	79.70	32.40	74.90	74.70

Table H.2: The number of nodes spent during the Feasibility phase.

	act&dist	aggrdive	br-rdfs inf	default	dfs inf	inf	rdfs inf	uct inf	uct-rdfs inf
10teams	125	125	101	266	21	587	21	306	257
acc-tight5	3843	7131	876	16931	5401	3628	5084	1473	608
arki001	3261	177	261	32	1941	4776	469	3501	224
atlanta-ip	192	32	18	128	522	266	880	15	15
bab5	47	47	98	61	98	84	98	426	116
bnatt350	4829	9574	12586	21343	16077	8298	10577	2775	14092
csched010	6368	8265	11934	4270	896863	55621	2626	3570	5167
danoint	16	16	3	16	12845	9	312	6	6
dcmulti	15	15	3	15	31	7	31	4	4
enigma	954	954	1172	954	870	1851	460	568	2759
flugpl	4	4	3	4	46	52	46	3	3
l152lav	6	6	4	6	8	6	8	4	4
lectsched-4-obj	289	289	548	289	2002	512	414	214	309
misc03	2	2	3	2	3	3	3	3	3
misc07	4	4	4	4	5	5	5	4	4
momentum2	5695	3303	4635	928	81871	10078	5872	70	2891
msc98-ip	2099	537	1460	1444	1456	23	3344	1147	4341
neos-1109824	10	10	16	10	12	6	12	14	14
neos-1337307	109	101	44	42	82	46	82	42	303
neos-1601936	215	215	17	179	744	98	420	17	17
neos-686190	49	49	16	381	1396308	105	3729	5	5
neos-849702	22854	6255	72407	6115	111664	20666	19942	26089	48917
net12	19	19	3	1000	459	3	440	3	3
ns1208400	1597	1565	3335	3027	64184	9908	7694	4909	2455
ns1688347	2258	2258	53	1065	394	147	263	100	142
ns1830653	13	13	116	13	139	367	139	134	135
rd-rplusc-21	122	122	81	122	7140	78072	93542	301	10863
rocII-4-11	12	12	124	12	125	204	125	60	63
timtab1	14	14	78	14	86	30	86	18	18
timtab2	833	1337	379	2392	2215	313	577	258	147
triptim1	23	23	3	33	54	30	54	4	4
vpphard	52	46	16	39	134	65	134	16	16

Table H.3: The number of LP iterations after branching started until a solution has been found, including those spent on *strong branching*

	act&dist	aggrdive	br-rdfs inf	default	dfs inf	inf	rdfs inf	uct inf	uct-rdfs inf
10teams	132328	132328	26774	146780	1814	52147	1814	60488	33113
acc-tight5	2685808	4676262	728932	9763187	3039025	2704908	2689526	1301975	349948
arki001	199295	44564	23863	12111	89430	190826	10290	148202	20381
atlanta-ip	975485	824201	270167	974012	284074	1429744	562993	157153	157153
bab5	230708	230708	115983	247392	11357	133240	11357	327763	120301
bnatt350	3263248	4520593	5153166	10608167	6210402	4029155	4822553	1861039	6194612
csched010	1256583	1492877	517756	854521	43136873	5511879	146198	674911	259212
danoint	122863	122863	5683	122863	891403	6114	30241	7628	7628
dcmulti	16127	16127	799	16127	831	516	831	820	820
enigma	14503	14503	5835	14503	2284	5844	1180	4617	9321
flugpl	236	236	21	236	34	102	34	21	21
l152lav	6385	6385	987	6385	165	317	165	987	987
lectsched-4-obj	177094	177094	138713	177094	38763	138486	27615	140213	97818
misc03	3870	3870	99	3870	89	89	89	99	99
misc07	6979	6979	815	6979	184	184	184	815	815
momentum2	2356747	1217060	453393	483432	2500624	1453667	517274	117314	282625
msc98-ip	7027954	3103686	600518	6716372	154466	185136	464943	6456065	1294336
neos-1109824	13695	13695	8762	13695	559	791	559	9028	9028
neos-1337307	304002	305880	186912	217930	14296	170437	14296	116252	127924
neos-1601936	1011263	1011263	1002805	955051	290308	652186	529575	763845	763845
neos-686190	105035	105035	9444	152070	8109551	27429	44575	4338	4338
neos-849702	3694447	1786567	9352204	1315111	18356317	3603406	1812627	4210679	4820557
net12	376492	376492	22231	2922665	238984	22231	298946	22231	22231
ns1208400	7823712	7788604	1854466	8564525	17711433	17559188	2685722	4314902	1230806
ns1688347	2075161	2075161	139728	1183777	61325	68753	35560	138090	143065
ns1830653	101685	101685	43502	101685	7314	116665	7314	73338	54780
rd-rplusc-21	138461	138461	26823	138461	15966	215430	257311	34465	84890
rocII-4-11	142944	142944	33654	142944	3734	34346	3734	33800	28699
timtab1	13737	13737	8972	13737	846	9464	846	6252	6252
timtab2	126279	199301	19207	251069	7622	30191	6493	38916	21023
triptim1	657029	657029	149615	807937	96948	234727	96948	117313	117313
vpphard	267362	312803	87272	206045	39069	80990	39069	52835	52835

Table H.4: The number of LP iterations after branching started until a solution has been found, including those spent on *strong branching*

	act&dist	aggrdive	br-rdfs inf	default	dfs inf	inf	rdfs inf	uct inf	uct-rdfs inf
10teams	17664	17664	26774	27160	1814	52147	1814	60488	33113
acc-tight5	1739405	3117724	728932	7320476	3039025	2704908	2689526	1301975	349948
arki001	180434	36306	23863	5635	89430	190826	10290	148202	20381
atlanta-ip	709463	592387	270167	670096	284074	1429744	562993	157153	157153
bab5	67208	67208	115983	79557	11357	133240	11357	327763	120301
bnatt350	2354753	3423008	5153166	8579123	6210402	4029155	4822553	1861039	6194612
csched010	833871	1004216	517756	504954	43136873	5511879	146198	674911	259212
danoint	23924	23924	5683	23924	891403	6114	30241	7628	7628
dcmulti	6159	6159	799	6159	831	516	831	820	820
enigma	6321	6321	5835	6321	2284	5844	1180	4617	9321
flugpl	96	96	21	96	34	102	34	21	21
l152lav	1055	1055	987	1055	165	317	165	987	987
lectsched-4-obj	32235	32235	138713	32235	38763	138486	27615	140213	97818
misc03	1953	1953	99	1953	89	89	89	99	99
misc07	2912	2912	815	2912	184	184	184	815	815
momentum2	1778688	798491	453393	279335	2500624	1453667	517274	117314	282625
msc98-ip	5234230	2456402	600518	5019177	154466	185136	464943	6456065	1294336
neos-1109824	4097	4097	8762	4097	559	791	559	9028	9028
neos-1337307	129896	131774	186912	61712	14296	170437	14296	116252	127924
neos-1601936	594799	594799	1002805	557693	290308	652186	529575	763845	763845
neos-686190	17826	17826	9444	35549	8109551	27429	44575	4338	4338
neos-849702	2744880	1151180	9352204	825834	18356317	3603406	1812627	4210679	4820557
net12	186951	186951	22231	2087361	238984	22231	298946	22231	22231
ns1208400	5775883	5781632	1854466	6322817	17711433	17559188	2685722	4314902	1230806
ns1688347	1229174	1229174	139728	622213	61325	68753	35560	138090	143065
ns1830653	22441	22441	43502	22441	7314	116665	7314	73338	54780
rd-rplusc-21	29750	29750	26823	29750	15966	215430	257311	34465	84890
rocII-4-11	19300	19300	33654	19300	3734	34346	3734	33800	28699
timtab1	2770	2770	8972	2770	846	9464	846	6252	6252
timtab2	47027	119584	19207	170855	7622	30191	6493	38916	21023
triptim1	308777	308777	149615	446417	96948	234727	96948	117313	117313
vpphard	110211	146629	87272	58440	39069	80990	39069	52835	52835

Table H.5: Successful termination of the Feasibility phase for every combination instance/setting. Missing check marks indicate that no solution was found until the solver hit the time limit of 1h.

	act&dist	aggrdive	br-rdfs inf	default	dfs inf	inf	rdfs inf	uct inf	uct-rdfs inf
10teams	✓	✓	✓	✓	✓	✓	✓	✓	✓
acc-tight5	✓	✓	✓	✓	✓	✓	✓	✓	✓
arki001	✓	✓	✓	✓	✓	✓	✓	✓	✓
atlanta-ip	✓	✓	✓	✓	✓	✓	✓	✓	✓
bab5	✓	✓	✓	✓	✓	✓	✓	✓	✓
bnatt350	✓	✓	✓	✓	✓	✓	✓	✓	✓
csched010	✓	✓	✓	✓		✓	✓	✓	✓
danoint	✓	✓	✓	✓	✓	✓	✓	✓	✓
dcmulti	✓	✓	✓	✓	✓	✓	✓	✓	✓
enigma	✓	✓	✓	✓	✓	✓	✓	✓	✓
flugpl	✓	✓	✓	✓	✓	✓	✓	✓	✓
l152lav	✓	✓	✓	✓	✓	✓	✓	✓	✓
lectsched-4-obj	✓	✓	✓	✓	✓	✓	✓	✓	✓
misc03	✓	✓	✓	✓	✓	✓	✓	✓	✓
misc07	✓	✓	✓	✓	✓	✓	✓	✓	✓
momentum2	✓	✓	✓	✓		✓	✓	✓	✓
msc98-ip	✓	✓	✓	✓	✓	✓	✓	✓	✓
neos-1109824	✓	✓	✓	✓	✓	✓	✓	✓	✓
neos-1337307	✓	✓	✓	✓	✓	✓	✓	✓	✓
neos-1601936	✓	✓	✓	✓	✓	✓	✓	✓	✓
neos-686190	✓	✓	✓	✓		✓	✓	✓	✓
neos-849702	✓	✓	✓	✓	✓	✓	✓	✓	✓
net12	✓	✓	✓	✓	✓	✓	✓	✓	✓
ns1208400	✓	✓	✓	✓	✓		✓	✓	✓
ns1688347	✓	✓	✓	✓	✓	✓	✓	✓	✓
ns1830653	✓	✓	✓	✓	✓	✓	✓	✓	✓
rd-rplusc-21	✓	✓	✓	✓	✓			✓	✓
rocII-4-11	✓	✓	✓	✓	✓	✓	✓	✓	✓
timtab1	✓	✓	✓	✓	✓	✓	✓	✓	✓
timtab2	✓	✓	✓	✓	✓	✓	✓	✓	✓
triptim1	✓	✓	✓	✓	✓	✓	✓	✓	✓
vpphard	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table H.6: Instancewise results for the Improvement phase experiment, see Section 5.1.2. For each instance and setting, we show the number of branch-and-bound nodes $n_{\mathcal{P}_2}$, the solving time $t_{\mathcal{P}_2}$, and the value of the primal integral $\Gamma_{\mathcal{P}_2}(T)$ during the Improvement phase, as well as the primal bound $c^t\hat{y}$ at termination.

		default	agg	agg l-uct	agg_{05}	l-agg	l-agg l-uct	l-uct
10teams	$n_{\mathcal{P}_2}$	931.0	872.0	872.0	872.0	973.0	973.0	931.0
	$t_{\mathcal{P}_2}$	10.7	18.9	20.4	18.7	24.9	27.1	11.0
	$c^t\hat{y}$	924.0	924.0	924.0	924.0	924.0	924.0	924.0
	$\Gamma_{\mathcal{P}_2}(T)$	50.0	24.5	24.9	24.3	125.8	137.6	48.8
a1c1s1	$n_{\mathcal{P}_2}$	289271.0	156359.0	172980.0	140762.0	195321.0	205853.0	234413.0
	$t_{\mathcal{P}_2}$	3599.6	3599.6	3599.5	3599.6	3599.6	3599.6	3599.6
	$c^t\hat{y}$	11711.5	11505.4	11531.5	11507.4	11624.4	11587.8	11513.4
	$\Gamma_{\mathcal{P}_2}(T)$	9958.3	6312.0	8028.2	6546.3	9567.0	9028.4	4956.6
aflow30a	$n_{\mathcal{P}_2}$	124.0	0.0	0.0	0.0	0.0	0.0	124.0
	$t_{\mathcal{P}_2}$	10.2	5.4	5.3	5.6	6.0	6.1	9.7
	$c^t\hat{y}$	1158.0	1158.0	1158.0	1158.0	1158.0	1158.0	1158.0
	$\Gamma_{\mathcal{P}_2}(T)$	373.7	99.1	99.1	102.0	412.3	410.1	378.2
aflow40b	$n_{\mathcal{P}_2}$	5546.0	20063.0	10146.0	20063.0	5085.0	22089.0	46898.0
	$t_{\mathcal{P}_2}$	124.1	319.1	193.5	321.5	153.9	286.2	554.0
	$c^t\hat{y}$	1168.0	1168.0	1168.0	1168.0	1168.0	1168.0	1168.0
	$\Gamma_{\mathcal{P}_2}(T)$	1719.3	2018.6	1841.9	2025.9	1873.9	2148.7	2391.9
air04	$n_{\mathcal{P}_2}$	178.0	162.0	162.0	162.0	162.0	162.0	178.0
	$t_{\mathcal{P}_2}$	64.2	59.7	59.0	61.4	59.2	59.8	63.6
	$c^t\hat{y}$	56137.0	56137.0	56137.0	56137.0	56137.0	56137.0	56137.0
	$\Gamma_{\mathcal{P}_2}(T)$	246.4	172.0	169.2	177.8	170.6	176.1	244.1
air05	$n_{\mathcal{P}_2}$	128.0	128.0	128.0	128.0	128.0	128.0	128.0
	$t_{\mathcal{P}_2}$	32.6	34.1	33.2	34.1	33.1	33.2	32.8
	$c^t\hat{y}$	26374.0	26374.0	26374.0	26374.0	26374.0	26374.0	26374.0
	$\Gamma_{\mathcal{P}_2}(T)$	194.6	200.9	195.9	195.8	194.9	192.6	192.9
app1-2	$n_{\mathcal{P}_2}$	3.0	3.0	3.0	3.0	3.0	3.0	3.0
	$t_{\mathcal{P}_2}$	589.3	809.5	953.6	809.6	774.6	944.3	587.9
	$c^t\hat{y}$	-41.0	-41.0	-41.0	-41.0	-41.0	-41.0	-41.0
	$\Gamma_{\mathcal{P}_2}(T)$	25887.3	34201.0	40511.2	34188.6	32676.5	40105.3	25805.9
arki001	$n_{\mathcal{P}_2}$	58768.0	65452.0	136688.0	40475.0	54894.0	90292.0	32223.0
	$t_{\mathcal{P}_2}$	405.1	681.9	1546.6	350.9	548.0	772.3	292.5
	$c^t\hat{y}$	7580813.0	7580813.0	7580813.0	7580813.0	7580813.0	7580813.0	7580813.0
	$\Gamma_{\mathcal{P}_2}(T)$	4.5	0.0	0.5	4.0	0.0	0.5	4.6
atlanta-ip	$n_{\mathcal{P}_2}$	4317.0	3754.0	3396.0	3327.0	4094.0	2993.0	3974.0
	$t_{\mathcal{P}_2}$	2967.0	2967.0	2969.2	2971.1	2969.4	2970.7	2968.4
	$c^t\hat{y}$	91.0	93.0	94.0	95.0	95.0	92.0	94.0
	$\Gamma_{\mathcal{P}_2}(T)$	10492.1	17110.1	19493.7	17988.0	20972.8	16603.4	14759.6
bab5	$n_{\mathcal{P}_2}$	11295.0	5684.0	5191.0	7057.0	8165.0	9269.0	9219.0
	$t_{\mathcal{P}_2}$	3397.3	3396.8	3396.4	3393.1	3398.5	3397.5	3396.5
	$c^t\hat{y}$	-106199.4	-106253.6	-106254.1	-106195.7	-106261.4	-106247.3	-106246.9
	$\Gamma_{\mathcal{P}_2}(T)$	2146.1	2235.7	2497.5	2632.8	2190.4	1891.9	2207.4
beasleyC3	$n_{\mathcal{P}_2}$	366597.0	354561.0	418463.0	393899.0	459286.0	466682.0	517819.0
	$t_{\mathcal{P}_2}$	3600.0	3599.9	3599.9	3599.9	3599.9	3599.9	3599.9
	$c^t\hat{y}$	765.0	759.0	759.0	761.0	761.0	758.0	759.0
	$\Gamma_{\mathcal{P}_2}(T)$	5925.9	3402.9	3177.0	4072.7	4346.4	3553.2	4060.1
bell5	$n_{\mathcal{P}_2}$	145.0	127.0	127.0	127.0	101.0	101.0	145.0
	$t_{\mathcal{P}_2}$	0.2	0.2	0.3	0.2	0.2	0.2	0.2
	$c^t\hat{y}$	8966406.5	8966406.5	8966406.5	8966406.5	8966406.5	8966406.5	8966406.5
	$\Gamma_{\mathcal{P}_2}(T)$	0.1	0.0	0.2	0.0	0.2	0.2	0.1
biella1	$n_{\mathcal{P}_2}$	1371.0	4975.0	4371.0	1783.0	673.0	1586.0	1832.0
	$t_{\mathcal{P}_2}$	576.9	1549.6	2348.4	1217.0	413.0	1047.3	474.2
	$c^t\hat{y}$	3065005.8	3065005.8	3065005.8	3065005.8	3065005.8	3065005.8	3065005.8
	$\Gamma_{\mathcal{P}_2}(T)$	5637.8	4843.1	5964.2	5148.4	4106.9	5666.8	3087.3
bienst2	$n_{\mathcal{P}_2}$	13355.0	14435.0	10952.0	14435.0	48169.0	60019.0	13355.0
	$t_{\mathcal{P}_2}$	86.7	97.0	86.5	97.8	195.9	241.0	86.8
	$c^t\hat{y}$	54.6	54.6	54.6	54.6	54.6	54.6	54.6
	$\Gamma_{\mathcal{P}_2}(T)$	230.0	239.8	236.7	239.1	294.9	245.2	231.7
binkar10_1	$n_{\mathcal{P}_2}$	731.0	4206.0	3457.0	3289.0	49512.0	2624.0	858.0
	$t_{\mathcal{P}_2}$	5.5	32.6	33.6	30.4	113.6	31.9	5.5
	$c^t\hat{y}$	6742.2	6742.2	6742.2	6742.2	6742.2	6742.2	6742.2
	$\Gamma_{\mathcal{P}_2}(T)$	62.0	76.0	68.1	64.9	62.9	57.7	51.3
blend2	$n_{\mathcal{P}_2}$	67.0	108.0	108.0	108.0	67.0	67.0	67.0
	$t_{\mathcal{P}_2}$	0.4	0.7	0.7	0.4	0.6	0.5	0.2

continued on next page

Table H.6: Instancewise results for the Improvement phase experiment, see Section 5.1.2.

		default	agg	agg l-uct	agg_{05}	l-agg	l-agg l-uct	l-uct
	$c^t\hat{y}$	7.6	7.6	7.6	7.6	7.6	7.6	7.6
	$\Gamma_{\mathcal{P}_2}(T)$	0.0	0.3	0.0	0.0	0.0	0.0	0.4
bley_xl1	$n_{\mathcal{P}_2}$	19.0	28.0	28.0	28.0	19.0	19.0	19.0
	$t_{\mathcal{P}_2}$	102.3	136.4	137.5	136.7	101.6	102.5	100.7
	$c^t\hat{y}$	190.0	190.0	190.0	190.0	190.0	190.0	190.0
	$\Gamma_{\mathcal{P}_2}(T)$	4208.4	1853.0	1829.9	1855.5	4186.1	4198.4	4144.9
cap6000	$n_{\mathcal{P}_2}$	1611.0	1365.0	1377.0	1377.0	1165.0	1165.0	1611.0
	$t_{\mathcal{P}_2}$	2.0	3.2	3.6	3.4	3.3	3.3	1.9
	$c^t\hat{y}$	-2451377.0	-2451377.0	-2451377.0	-2451377.0	-2451377.0	-2451377.0	-2451377.0
	$\Gamma_{\mathcal{P}_2}(T)$	12.4	40.2	52.1	48.6	13.2	3.9	4.6
core2536-691	$n_{\mathcal{P}_2}$	217.0	283.0	283.0	283.0	186.0	186.0	217.0
	$t_{\mathcal{P}_2}$	308.3	371.2	378.4	371.8	335.2	345.5	312.1
	$c^t\hat{y}$	689.0	689.0	689.0	689.0	689.0	689.0	689.0
	$\Gamma_{\mathcal{P}_2}(T)$	320.9	364.9	371.0	363.8	321.0	352.8	341.0
cov1075	$n_{\mathcal{P}_2}$	255.0	90.0	90.0	90.0	150.0	150.0	255.0
	$t_{\mathcal{P}_2}$	15.3	13.4	12.6	12.7	13.5	13.9	14.9
	$c^t\hat{y}$	20.0	20.0	20.0	20.0	20.0	20.0	20.0
	$\Gamma_{\mathcal{P}_2}(T)$	254.4	140.7	130.3	142.8	177.8	186.7	239.0
csched010	$n_{\mathcal{P}_2}$	71248.0	14690.0	71124.0	13934.0	33398.0	52256.0	34613.0
	$t_{\mathcal{P}_2}$	618.3	221.4	710.3	188.1	417.8	490.3	309.7
	$c^t\hat{y}$	408.0	408.0	408.0	408.0	408.0	408.0	408.0
	$\Gamma_{\mathcal{P}_2}(T)$	1786.4	2653.8	4278.8	2330.4	4131.7	2753.6	1482.1
danoint	$n_{\mathcal{P}_2}$	1121.0	1038.0	500.0	1038.0	686.0	686.0	1121.0
	$t_{\mathcal{P}_2}$	24.8	31.6	18.6	33.8	23.5	21.8	24.6
	$c^t\hat{y}$	65.7	65.7	65.7	65.7	65.7	65.7	65.7
	$\Gamma_{\mathcal{P}_2}(T)$	137.3	131.9	121.2	134.8	127.1	118.3	136.9
dcmulti	$n_{\mathcal{P}_2}$	292.0	292.0	292.0	292.0	292.0	292.0	292.0
	$t_{\mathcal{P}_2}$	0.4	0.4	0.5	0.5	0.4	0.5	0.4
	$c^t\hat{y}$	188182.0	188182.0	188182.0	188182.0	188182.0	188182.0	188182.0
	$\Gamma_{\mathcal{P}_2}(T)$	0.0	0.6	0.5	0.0	0.6	0.0	0.5
dfn-gwin-UUM	$n_{\mathcal{P}_2}$	2409.0	2214.0	364.0	1317.0	600.0	2473.0	1396.0
	$t_{\mathcal{P}_2}$	26.4	48.3	34.5	43.4	32.6	54.3	18.9
	$c^t\hat{y}$	38752.0	38752.0	38752.0	38752.0	38752.0	38752.0	38752.0
	$\Gamma_{\mathcal{P}_2}(T)$	438.6	1198.3	1109.3	1176.7	864.2	1024.8	412.9
ds	$n_{\mathcal{P}_2}$	327.0	269.0	269.0	262.0	239.0	272.0	318.0
	$t_{\mathcal{P}_2}$	3594.0	3594.0	3594.0	3594.0	3593.9	3594.0	3594.0
	$c^t\hat{y}$	458.4	358.1	353.0	323.2	316.4	368.9	336.6
	$\Gamma_{\mathcal{P}_2}(T)$	297823.4	290672.8	283503.2	276653.9	274168.8	283498.3	278749.0
dsbmip	$n_{\mathcal{P}_2}$	14.0	0.0	0.0	0.0	0.0	0.0	14.0
	$t_{\mathcal{P}_2}$	1.0	0.7	0.9	0.7	0.9	0.9	1.1
	$c^t\hat{y}$	-305.2	-305.2	-305.2	-305.2	-305.2	-305.2	-305.2
	$\Gamma_{\mathcal{P}_2}(T)$	47.4	37.1	52.0	38.1	46.4	45.3	53.3
eil33-2	$n_{\mathcal{P}_2}$	362.0	362.0	362.0	362.0	362.0	362.0	362.0
	$t_{\mathcal{P}_2}$	40.2	43.5	43.0	43.0	41.0	41.3	39.6
	$c^t\hat{y}$	934.0	934.0	934.0	934.0	934.0	934.0	934.0
	$\Gamma_{\mathcal{P}_2}(T)$	439.2	492.4	476.7	478.4	437.1	443.6	425.8
eilB101	$n_{\mathcal{P}_2}$	7804.0	3768.0	3768.0	3768.0	4447.0	4447.0	5858.0
	$t_{\mathcal{P}_2}$	424.0	345.4	358.8	346.6	361.5	369.6	342.1
	$c^t\hat{y}$	1216.9	1216.9	1216.9	1216.9	1216.9	1216.9	1216.9
	$\Gamma_{\mathcal{P}_2}(T)$	1245.4	1312.0	1349.9	1306.6	1342.4	1375.8	1252.0
fast0507	$n_{\mathcal{P}_2}$	799.0	349.0	349.0	349.0	233.0	233.0	799.0
	$t_{\mathcal{P}_2}$	494.7	209.7	211.8	210.7	163.2	163.8	494.2
	$c^t\hat{y}$	174.0	174.0	174.0	174.0	174.0	174.0	174.0
	$\Gamma_{\mathcal{P}_2}(T)$	904.7	640.8	642.2	644.8	609.1	614.8	916.1
flugpl	$n_{\mathcal{P}_2}$	12.0	16.0	16.0	16.0	12.0	12.0	12.0
	$t_{\mathcal{P}_2}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$c^t\hat{y}$	1201500.0	1201500.0	1201500.0	1201500.0	1201500.0	1201500.0	1201500.0
	$\Gamma_{\mathcal{P}_2}(T)$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
gesa2-o	$n_{\mathcal{P}_2}$	1.0	0.0	0.0	0.0	0.0	0.0	1.0
	$t_{\mathcal{P}_2}$	1.5	0.8	1.1	1.0	0.8	0.8	1.3
	$c^t\hat{y}$	25779856.4	25779856.4	25779856.4	25779856.4	25779856.4	25779856.4	25779856.4
	$\Gamma_{\mathcal{P}_2}(T)$	6.4	5.0	6.4	6.4	4.4	5.2	7.1
gesa3_o	$n_{\mathcal{P}_2}$	2.0	2.0	2.0	2.0	2.0	2.0	2.0
	$t_{\mathcal{P}_2}$	1.8	1.9	1.9	1.9	1.8	1.6	1.7
	$c^t\hat{y}$	27991042.6	27991042.6	27991042.6	27991042.6	27991042.6	27991042.6	27991042.6
	$\Gamma_{\mathcal{P}_2}(T)$	9.6	9.5	9.5	9.5	7.2	7.4	9.5

continued on next page

Table H.6: Instancewise results for the Improvement phase experiment, see Section 5.1.2.

		default	agg	agg l-uct	agg_{05}	l-agg	l-agg l-uct	l-uct
glass4	$n_{\mathcal{P}_2}$	8127712.0	6274756.0	7334537.0	6579326.0	7639007.0	6777628.0	7703955.0
	$t_{\mathcal{P}_2}$	3598.2	3598.3	3598.5	3598.4	3598.3	3598.2	3598.4
	$c^t\hat{y}$	1550014550.0	1500013450.0	1500012966.7	1475013050.0	1600013400.0	1550015800.0	1566682933.3
	$\Gamma_{\mathcal{P}_2}(T)$	84430.6	81001.4	82703.6	69623.7	90965.3	89142.9	86870.0
gmu-35-40	$n_{\mathcal{P}_2}$	5973738.0	5894603.0	5882942.0	5705752.0	5050597.0	6157128.0	6924541.0
	$t_{\mathcal{P}_2}$	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0
	$c^t\hat{y}$	-2406528.8	-2405954.4	-2405976.2	-2406037.4	-2406287.5	-2406412.1	-2406328.9
	$\Gamma_{\mathcal{P}_2}(T)$	89.4	200.8	218.5	165.0	105.2	88.0	98.3
iis-100-0-cov	$n_{\mathcal{P}_2}$	518.0	316.0	316.0	316.0	101.0	101.0	796.0
	$t_{\mathcal{P}_2}$	49.5	51.4	48.1	49.9	32.4	32.8	55.9
	$c^t\hat{y}$	29.0	29.0	29.0	29.0	29.0	29.0	29.0
	$\Gamma_{\mathcal{P}_2}(T)$	633.1	701.0	665.8	674.3	548.6	554.4	648.4
iis-bupa-cov	$n_{\mathcal{P}_2}$	826.0	111.0	1.0	111.0	1.0	1.0	472.0
	$t_{\mathcal{P}_2}$	150.8	206.0	110.7	207.0	93.6	93.6	130.8
	$c^t\hat{y}$	36.0	36.0	36.0	36.0	36.0	36.0	36.0
	$\Gamma_{\mathcal{P}_2}(T)$	1434.3	2627.0	2446.1	2645.4	2344.7	2345.6	1112.2
iis-pima-cov	$n_{\mathcal{P}_2}$	17161.0	216.0	200.0	216.0	1.0	17224.0	17110.0
	$t_{\mathcal{P}_2}$	1230.3	199.3	161.1	200.1	57.1	1579.6	1241.7
	$c^t\hat{y}$	33.0	33.0	33.0	33.0	33.0	33.0	33.0
	$\Gamma_{\mathcal{P}_2}(T)$	4285.5	1790.6	1812.9	1789.7	1316.0	6095.5	4412.0
l152lav	$n_{\mathcal{P}_2}$	38.0	38.0	38.0	38.0	38.0	38.0	38.0
	$t_{\mathcal{P}_2}$	1.5	1.5	1.5	1.8	1.5	1.6	1.5
	$c^t\hat{y}$	4722.0	4722.0	4722.0	4722.0	4722.0	4722.0	4722.0
	$\Gamma_{\mathcal{P}_2}(T)$	0.0	1.0	1.0	0.2	1.0	1.0	0.0
lectsched-4-obj	$n_{\mathcal{P}_2}$	23933.0	9509.0	4008.0	9509.0	43767.0	13007.0	11640.0
	$t_{\mathcal{P}_2}$	367.3	328.0	157.8	329.1	489.9	286.3	246.0
	$c^t\hat{y}$	4.0	4.0	4.0	4.0	4.0	4.0	4.0
	$\Gamma_{\mathcal{P}_2}(T)$	18713.8	20311.2	8868.6	20359.9	23010.4	15937.6	13773.2
lseu	$n_{\mathcal{P}_2}$	282.0	200.0	398.0	200.0	398.0	376.0	282.0
	$t_{\mathcal{P}_2}$	0.7	0.8	0.9	1.1	0.8	0.9	0.5
	$c^t\hat{y}$	1120.0	1120.0	1120.0	1120.0	1120.0	1120.0	1120.0
	$\Gamma_{\mathcal{P}_2}(T)$	1.5	2.1	2.4	2.8	2.2	2.2	1.2
m100n500k4r1	$n_{\mathcal{P}_2}$	3671393.0	3346326.0	3320497.0	3315109.0	3581072.0	3526732.0	3413839.0
	$t_{\mathcal{P}_2}$	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0
	$c^t\hat{y}$	-24.0	-24.0	-24.0	-24.0	-24.0	-24.0	-24.0
	$\Gamma_{\mathcal{P}_2}(T)$	14555.0	14670.6	14687.6	14679.8	14774.0	14809.2	14506.2
macrophage	$n_{\mathcal{P}_2}$	540923.0	414801.0	392595.0	375258.0	530241.0	549327.0	513521.0
	$t_{\mathcal{P}_2}$	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0
	$c^t\hat{y}$	380.0	376.0	375.0	377.0	376.0	377.0	376.0
	$\Gamma_{\mathcal{P}_2}(T)$	7014.1	3434.7	2965.6	4854.1	5231.3	3974.2	3081.1
map18	$n_{\mathcal{P}_2}$	83.0	0.0	0.0	0.0	0.0	0.0	0.0
	$t_{\mathcal{P}_2}$	269.6	107.6	85.5	108.3	86.8	76.5	75.8
	$c^t\hat{y}$	-847.0	-847.0	-847.0	-847.0	-847.0	-847.0	-847.0
	$\Gamma_{\mathcal{P}_2}(T)$	3451.5	4595.0	3988.6	4635.1	3987.1	3689.7	3669.2
markshare2	$n_{\mathcal{P}_2}$	33127583.0	27233740.0	26205218.0	27069884.0	28161250.0	28563823.0	32241464.0
	$t_{\mathcal{P}_2}$	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0
	$c^t\hat{y}$	18.0	15.0	12.0	17.0	15.0	14.0	17.0
	$\Gamma_{\mathcal{P}_2}(T)$	340708.4	339448.0	337551.5	341147.6	340217.2	337002.5	340273.9
mas74	$n_{\mathcal{P}_2}$	36761.0	1151.0	286613.0	1151.0	159291.0	1046767.0	36761.0
	$t_{\mathcal{P}_2}$	21.0	5.9	134.9	6.7	84.8	327.2	21.4
	$c^t\hat{y}$	11801.2	11801.2	11801.2	11801.2	11801.2	11801.2	11801.2
	$\Gamma_{\mathcal{P}_2}(T)$	83.3	40.8	250.8	45.8	339.2	401.1	86.2
mcsched	$n_{\mathcal{P}_2}$	16140.0	7927.0	7927.0	14008.0	10340.0	10340.0	16140.0
	$t_{\mathcal{P}_2}$	174.6	122.8	130.1	212.2	138.1	146.2	177.6
	$c^t\hat{y}$	211913.0	211913.0	211913.0	211913.0	211913.0	211913.0	211913.0
	$\Gamma_{\mathcal{P}_2}(T)$	234.8	233.4	233.6	189.0	237.8	237.0	231.3
mine-166-5	$n_{\mathcal{P}_2}$	1717.0	262.0	67.0	4391.0	82.0	146.0	1717.0
	$t_{\mathcal{P}_2}$	30.0	33.7	35.4	75.3	32.5	32.9	30.2
	$c^t\hat{y}$	-566395707.9	-566395707.9	-566395707.9	-566395707.9	-566395707.9	-566395707.9	-566395707.9
	$\Gamma_{\mathcal{P}_2}(T)$	1604.2	1990.9	1969.6	5071.2	1948.0	1883.6	1602.9
mine-90-10	$n_{\mathcal{P}_2}$	44693.0	426195.0	404159.0	56632.0	157675.0	131019.0	53503.0
	$t_{\mathcal{P}_2}$	177.9	1495.8	1458.1	274.1	679.7	485.8	189.1
	$c^t\hat{y}$	-784302337.6	-784302337.6	-784302337.6	-784302337.6	-784302337.6	-784302337.6	-784302337.6
	$\Gamma_{\mathcal{P}_2}(T)$	2078.3	2681.4	2699.1	3239.9	2929.1	2492.4	1830.9
misc07	$n_{\mathcal{P}_2}$	51.0	51.0	51.0	51.0	51.0	51.0	51.0
	$t_{\mathcal{P}_2}$	1.1	0.7	0.7	0.7	0.7	0.7	0.8

continued on next page

Table H.6: Instancewise results for the Improvement phase experiment, see Section 5.1.2.

		default	agg	agg l-uct	agg_{05}	l-agg	l-agg l-uct	l-uct
	$c^t\hat{y}$	2810.0	2810.0	2810.0	2810.0	2810.0	2810.0	2810.0
	$\Gamma_{\mathcal{P}_2}(T)$	5.0	4.7	4.7	2.8	4.6	5.2	6.1
mkc	$n_{\mathcal{P}_2}$	1098250.0	1100760.0	938504.0	913397.0	1106178.0	933574.0	1311091.0
	$t_{\mathcal{P}_2}$	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0
	$c^t\hat{y}$	-562.7	-563.2	-562.8	-561.3	-563.3	-562.1	-563.6
	$\Gamma_{\mathcal{P}_2}(T)$	1830.7	1396.2	1638.2	2220.7	1555.7	1802.6	1916.3
mod011	$n_{\mathcal{P}_2}$	324.0	1.0	78.0	1.0	84.0	101.0	324.0
	$t_{\mathcal{P}_2}$	104.7	27.8	141.0	28.3	199.4	158.1	104.8
	$c^t\hat{y}$	-54558535.0	-54558535.0	-54558535.0	-54558535.0	-54558535.0	-54558535.0	-54558535.0
	$\Gamma_{\mathcal{P}_2}(T)$	1579.5	1317.5	1399.7	1343.4	1352.2	1536.3	1570.6
modglob	$n_{\mathcal{P}_2}$	261.0	0.0	0.0	0.0	0.0	0.0	261.0
	$t_{\mathcal{P}_2}$	1.2	0.9	1.2	1.1	0.9	0.7	1.1
	$c^t\hat{y}$	20740508.1	20740508.1	20740508.1	20740508.1	20740508.1	20740508.1	20740508.1
	$\Gamma_{\mathcal{P}_2}(T)$	0.0	0.0	7.9	7.9	0.0	0.0	0.0
momentum1	$n_{\mathcal{P}_2}$	15865.0	7410.0	8104.0	7422.0	9938.0	5639.0	7469.0
	$t_{\mathcal{P}_2}$	3594.7	3594.5	3594.7	3594.6	3594.8	3594.8	3594.6
	$c^t\hat{y}$	115610.8	128486.2	128465.5	128488.6	122040.7	128470.5	115648.2
	$\Gamma_{\mathcal{P}_2}(T)$	93143.3	63772.2	65476.5	63798.9	46004.6	62819.2	70560.2
momentum2	$n_{\mathcal{P}_2}$	12265.0	7262.0	9310.0	6223.0	13108.0	11870.0	10926.0
	$t_{\mathcal{P}_2}$	3184.3	3181.8	3182.9	3179.8	3182.1	3181.7	3179.6
	$c^t\hat{y}$	13812.6	12415.8	13812.3	15421.6	13814.5	13813.0	13813.0
	$\Gamma_{\mathcal{P}_2}(T)$	41827.4	54587.9	49326.8	68882.4	41247.1	42700.1	61972.9
msc98-ip	$n_{\mathcal{P}_2}$	285.0	296.0	305.0	308.0	239.0	269.0	284.0
	$t_{\mathcal{P}_2}$	628.8	624.5	634.5	635.3	625.2	634.6	619.5
	$c^t\hat{y}$	25250796.0	30681010.0	30681010.0	30681010.0	30681010.0	30681010.0	26778562.0
	$\Gamma_{\mathcal{P}_2}(T)$	18586.9	22064.7	22420.5	22448.0	22093.3	22400.5	19530.9
mzzv11	$n_{\mathcal{P}_2}$	1038.0	913.0	913.0	1171.0	802.0	1000.0	1038.0
	$t_{\mathcal{P}_2}$	197.5	264.8	260.4	278.9	335.2	322.7	198.0
	$c^t\hat{y}$	-21718.0	-21718.0	-21718.0	-21718.0	-21718.0	-21718.0	-21718.0
	$\Gamma_{\mathcal{P}_2}(T)$	12441.0	12063.9	12071.9	12049.4	12551.0	12603.8	12445.0
mzzv42z	$n_{\mathcal{P}_2}$	513.0	257.0	257.0	257.0	131.0	131.0	513.0
	$t_{\mathcal{P}_2}$	329.4	137.8	137.4	137.3	162.1	163.9	331.2
	$c^t\hat{y}$	-20540.0	-20540.0	-20540.0	-20540.0	-20540.0	-20540.0	-20540.0
	$\Gamma_{\mathcal{P}_2}(T)$	12011.8	7698.5	7779.0	7790.4	8297.2	8399.4	12110.5
n3div36	$n_{\mathcal{P}_2}$	112508.0	93000.0	90445.0	92365.0	20392.0	15398.0	112648.0
	$t_{\mathcal{P}_2}$	3597.8	3597.7	3597.8	3597.7	1175.9	976.5	3597.4
	$c^t\hat{y}$	131000.0	131000.0	132800.0	131000.0	130800.0	130800.0	131000.0
	$\Gamma_{\mathcal{P}_2}(T)$	5009.8	3357.4	8929.4	3359.1	5178.9	5042.7	5010.7
n3seq24	$n_{\mathcal{P}_2}$	145.0	213.0	198.0	229.0	86.0	81.0	145.0
	$t_{\mathcal{P}_2}$	3576.9	3576.0	3577.2	3578.4	3577.8	3577.3	3577.3
	$c^t\hat{y}$	62800.0	54400.0	53000.0	55400.0	60800.0	59000.0	62800.0
	$\Gamma_{\mathcal{P}_2}(T)$	113142.8	147594.9	153717.2	150378.0	119564.6	120273.0	114442.9
n4-3	$n_{\mathcal{P}_2}$	933.0	467.0	459.0	364.0	435.0	512.0	968.0
	$t_{\mathcal{P}_2}$	61.8	73.1	92.4	60.9	72.9	80.5	65.0
	$c^t\hat{y}$	8993.0	8993.0	8993.0	8993.0	8993.0	8993.0	8993.0
	$\Gamma_{\mathcal{P}_2}(T)$	1397.8	811.7	875.7	703.8	894.5	837.8	1412.0
neos-1109824	$n_{\mathcal{P}_2}$	7.0	7.0	7.0	7.0	7.0	7.0	7.0
	$t_{\mathcal{P}_2}$	2.3	2.5	2.3	2.3	2.3	2.6	2.3
	$c^t\hat{y}$	378.0	378.0	378.0	378.0	378.0	378.0	378.0
	$\Gamma_{\mathcal{P}_2}(T)$	28.5	29.4	28.5	26.9	28.3	29.8	29.3
neos-1337307	$n_{\mathcal{P}_2}$	2657.0	454.0	946.0	454.0	953.0	608.0	2657.0
	$t_{\mathcal{P}_2}$	236.8	131.6	206.8	129.6	143.9	131.2	236.6
	$c^t\hat{y}$	-202319.0	-202319.0	-202319.0	-202319.0	-202319.0	-202319.0	-202319.0
	$\Gamma_{\mathcal{P}_2}(T)$	10.8	8.9	10.6	8.9	17.1	20.8	6.8
neos-1396125	$n_{\mathcal{P}_2}$	20598.0	7395.0	13159.0	14371.0	17843.0	10237.0	16446.0
	$t_{\mathcal{P}_2}$	423.0	352.5	470.1	540.2	670.0	443.0	434.6
	$c^t\hat{y}$	3000.0	3000.0	3000.0	3000.0	3000.0	3000.0	3000.0
	$\Gamma_{\mathcal{P}_2}(T)$	2794.8	637.2	555.1	4074.6	4225.6	644.2	4202.7
neos-1601936	$n_{\mathcal{P}_2}$	2689.0	1948.0	1948.0	1948.0	2119.0	1915.0	2689.0
	$t_{\mathcal{P}_2}$	3242.9	3243.5	3240.6	3242.0	3244.5	3244.7	3243.1
	$c^t\hat{y}$	4.0	4.0	4.0	4.0	4.0	5.0	4.0
	$\Gamma_{\mathcal{P}_2}(T)$	125832.1	133812.1	134306.1	133788.0	148522.4	150318.8	125927.7
neos-476283	$n_{\mathcal{P}_2}$	295.0	130.0	105.0	105.0	175.0	175.0	295.0
	$t_{\mathcal{P}_2}$	168.4	211.7	225.8	214.8	245.1	267.0	171.1
	$c^t\hat{y}$	406.4	406.4	406.4	406.4	406.4	406.4	406.4
	$\Gamma_{\mathcal{P}_2}(T)$	593.3	592.9	610.9	582.0	608.4	624.2	629.4

continued on next page

Table H.6: Instancewise results for the Improvement phase experiment, see Section 5.1.2.

		default	agg	agg l-uct	agg_{05}	l-agg	l-agg l-uct	l-uct
neos-686190	$n_{\mathcal{P}_2}$	5008.0	6739.0	3689.0	6188.0	5272.0	5272.0	5328.0
	$t_{\mathcal{P}_2}$	50.3	81.9	61.9	77.4	59.0	67.4	53.2
	$c^t\hat{y}$	6730.0	6730.0	6730.0	6730.0	6730.0	6730.0	6730.0
	$\Gamma_{\mathcal{P}_2}(T)$	883.8	1103.0	1166.7	1057.4	587.6	710.9	881.4
neos-916792	$n_{\mathcal{P}_2}$	25938.0	1081.0	26032.0	17855.0	75929.0	30819.0	53258.0
	$t_{\mathcal{P}_2}$	208.9	60.3	261.3	214.3	406.0	302.6	257.9
	$c^t\hat{y}$	31.9	31.9	31.9	31.9	31.9	31.9	31.9
	$\Gamma_{\mathcal{P}_2}(T)$	835.3	715.5	1055.7	961.9	1036.8	1315.0	795.3
neos-934278	$n_{\mathcal{P}_2}$	377.0	573.0	203.0	573.0	348.0	331.0	264.0
	$t_{\mathcal{P}_2}$	3600.0	3600.0	3599.9	3600.0	3599.9	3599.9	3599.9
	$c^t\hat{y}$	283.0	263.0	280.0	263.0	270.0	266.0	1290.0
	$\Gamma_{\mathcal{P}_2}(T)$	108576.1	52359.2	81597.8	52360.3	63700.1	56271.1	294029.7
neos13	$n_{\mathcal{P}_2}$	3963.0	2213.0	0.0	453.0	559.0	584.0	3014.0
	$t_{\mathcal{P}_2}$	1479.3	3598.6	1046.9	3598.7	3598.8	3598.7	1695.9
	$c^t\hat{y}$	-95.5	-94.1	-95.5	-94.4	-93.3	-95.3	-95.5
	$\Gamma_{\mathcal{P}_2}(T)$	40712.1	54987.8	25205.0	56138.2	69156.5	56470.5	44522.9
neos18	$n_{\mathcal{P}_2}$	340.0	103.0	103.0	209.0	149.0	149.0	340.0
	$t_{\mathcal{P}_2}$	13.2	12.3	13.2	15.6	14.2	14.3	13.2
	$c^t\hat{y}$	16.0	16.0	16.0	16.0	16.0	16.0	16.0
	$\Gamma_{\mathcal{P}_2}(T)$	356.2	225.4	254.7	221.7	244.5	256.2	360.0
net12	$n_{\mathcal{P}_2}$	265.0	170.0	170.0	170.0	404.0	404.0	265.0
	$t_{\mathcal{P}_2}$	168.8	140.6	141.5	141.1	334.6	340.2	169.4
	$c^t\hat{y}$	214.0	214.0	214.0	214.0	214.0	214.0	214.0
	$\Gamma_{\mathcal{P}_2}(T)$	4596.2	3620.7	3658.7	3623.8	7057.7	7150.0	4607.7
netdiversion	$n_{\mathcal{P}_2}$	58.0	36.0	36.0	36.0	54.0	58.0	58.0
	$t_{\mathcal{P}_2}$	3595.0	3603.0	3581.8	3593.5	3582.0	3603.2	3587.4
	$c^t\hat{y}$	251.0	4900438.0	4900438.0	4900438.0	251.0	251.0	251.0
	$\Gamma_{\mathcal{P}_2}(T)$	343615.5	358171.3	358159.3	358177.3	346994.8	344382.8	342847.2
newdano	$n_{\mathcal{P}_2}$	538282.0	409477.0	358327.0	376318.0	381392.0	55822.0	538282.0
	$t_{\mathcal{P}_2}$	1158.5	1065.3	951.4	944.1	988.4	225.0	1160.7
	$c^t\hat{y}$	65.7	65.7	65.7	65.7	65.7	65.7	65.7
	$\Gamma_{\mathcal{P}_2}(T)$	1825.5	2161.2	1793.2	1430.9	1654.9	717.1	1841.5
noswot	$n_{\mathcal{P}_2}$	260.0	261.0	96.0	171.0	260.0	200.0	260.0
	$t_{\mathcal{P}_2}$	0.9	0.9	1.5	1.3	1.1	1.5	0.8
	$c^t\hat{y}$	-41.0	-41.0	-41.0	-41.0	-41.0	-41.0	-41.0
	$\Gamma_{\mathcal{P}_2}(T)$	8.9	7.8	8.9	9.2	7.5	8.5	8.2
ns1688347	$n_{\mathcal{P}_2}$	2701.0	5581.0	5581.0	6170.0	495.0	495.0	2701.0
	$t_{\mathcal{P}_2}$	411.0	560.8	565.5	764.3	228.7	244.8	412.6
	$c^t\hat{y}$	27.0	27.0	27.0	27.0	27.0	27.0	27.0
	$\Gamma_{\mathcal{P}_2}(T)$	7825.0	6978.4	7067.9	12926.0	4419.3	4745.5	7841.2
ns1758913	$n_{\mathcal{P}_2}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$t_{\mathcal{P}_2}$	2930.1	2929.9	2929.4	2929.3	2929.4	2928.6	2928.8
	$c^t\hat{y}$	-236.8	-285.5	-285.5	-285.5	-236.8	-236.8	-236.8
	$\Gamma_{\mathcal{P}_2}(T)$	245314.6	235487.8	235452.6	235450.9	245256.0	245182.2	245200.2
ns1830653	$n_{\mathcal{P}_2}$	13090.0	25963.0	25188.0	25963.0	7920.0	39663.0	13090.0
	$t_{\mathcal{P}_2}$	198.9	396.9	404.2	400.2	201.4	541.8	200.0
	$c^t\hat{y}$	20622.0	20622.0	20622.0	20622.0	20622.0	20622.0	20622.0
	$\Gamma_{\mathcal{P}_2}(T)$	7272.0	10699.5	10186.2	10847.6	8456.7	12214.1	7293.9
nsrand-ipx	$n_{\mathcal{P}_2}$	696764.0	541349.0	503443.0	523390.0	782402.0	582085.0	661141.0
	$t_{\mathcal{P}_2}$	3599.0	3599.2	3599.0	3599.1	3599.1	3599.0	3599.3
	$c^t\hat{y}$	51840.0	52000.0	51360.0	52000.0	51360.0	51840.0	51680.0
	$\Gamma_{\mathcal{P}_2}(T)$	7372.6	6978.6	5262.5	6981.8	5521.2	6447.8	6945.7
opm2-z7-s2	$n_{\mathcal{P}_2}$	2056.0	1041.0	3613.0	2530.0	1830.0	1894.0	2056.0
	$t_{\mathcal{P}_2}$	784.6	442.6	914.0	692.9	568.9	565.3	787.9
	$c^t\hat{y}$	-10280.0	-10280.0	-10280.0	-10280.0	-10280.0	-10280.0	-10280.0
	$\Gamma_{\mathcal{P}_2}(T)$	8024.1	10591.0	10727.5	11884.7	11106.8	11352.3	8054.7
p0201	$n_{\mathcal{P}_2}$	34.0	6.0	6.0	6.0	7.0	7.0	34.0
	$t_{\mathcal{P}_2}$	1.3	1.1	1.4	1.3	1.1	1.1	1.4
	$c^t\hat{y}$	7615.0	7615.0	7615.0	7615.0	7615.0	7615.0	7615.0
	$\Gamma_{\mathcal{P}_2}(T)$	17.1	10.1	18.1	17.4	8.2	12.4	17.6
p2756	$n_{\mathcal{P}_2}$	135.0	2.0	2.0	2.0	5.0	5.0	135.0
	$t_{\mathcal{P}_2}$	1.6	1.3	1.3	1.4	1.3	1.3	1.6
	$c^t\hat{y}$	3124.0	3124.0	3124.0	3124.0	3124.0	3124.0	3124.0
	$\Gamma_{\mathcal{P}_2}(T)$	10.1	1.9	2.9	2.7	8.2	8.4	6.5
pg5_34	$n_{\mathcal{P}_2}$	241713.0	79857.0	274157.0	79846.0	71854.0	165395.0	179289.0
	$t_{\mathcal{P}_2}$	1185.7	644.8	1643.5	648.1	625.2	1057.5	1045.2

continued on next page

Table H.6: Instancewise results for the Improvement phase experiment, see Section 5.1.2.

		default	agg	agg l-uct	agg_{05}	l-agg	l-agg l-uct	l-uct
	$c^t\hat{y}$	-14339.4	-14339.4	-14339.4	-14339.4	-14339.4	-14339.4	-14339.4
	$\Gamma_{\mathcal{P}_2}(T)$	172.7	175.2	208.4	166.3	146.7	176.5	186.0
pigeon-10	$n_{\mathcal{P}_2}$	200.0	1.0	1.0	1.0	1.0	1.0	200.0
	$t_{\mathcal{P}_2}$	4.9	1.4	1.7	1.6	1.8	1.5	4.9
	$c^t\hat{y}$	-9000.0	-9000.0	-9000.0	-9000.0	-9000.0	-9000.0	-9000.0
	$\Gamma_{\mathcal{P}_2}(T)$	489.0	140.0	169.0	160.0	180.0	150.0	490.0
pk1	$n_{\mathcal{P}_2}$	15661.0	43672.0	40463.0	16279.0	49691.0	20648.0	15661.0
	$t_{\mathcal{P}_2}$	9.1	41.4	36.7	22.3	32.5	20.8	9.3
	$c^t\hat{y}$	11.0	11.0	11.0	11.0	11.0	11.0	11.0
	$\Gamma_{\mathcal{P}_2}(T)$	324.7	1140.4	821.1	723.4	728.7	655.9	322.9
pp08a	$n_{\mathcal{P}_2}$	132.0	132.0	132.0	132.0	132.0	132.0	132.0
	$t_{\mathcal{P}_2}$	1.3	1.5	1.4	1.3	1.1	1.3	1.4
	$c^t\hat{y}$	7350.0	7350.0	7350.0	7350.0	7350.0	7350.0	7350.0
	$\Gamma_{\mathcal{P}_2}(T)$	26.2	26.3	27.9	26.2	14.8	21.8	27.7
pp08aCUTS	$n_{\mathcal{P}_2}$	83.0	83.0	83.0	83.0	83.0	83.0	83.0
	$t_{\mathcal{P}_2}$	1.1	1.8	1.6	1.6	2.1	1.5	1.1
	$c^t\hat{y}$	7350.0	7350.0	7350.0	7350.0	7350.0	7350.0	7350.0
	$\Gamma_{\mathcal{P}_2}(T)$	18.8	38.7	23.3	30.7	34.2	21.9	21.9
protfold	$n_{\mathcal{P}_2}$	5511.0	3963.0	2589.0	2859.0	5438.0	5777.0	5511.0
	$t_{\mathcal{P}_2}$	3599.9	3599.8	3599.7	3599.9	3599.8	3599.8	3599.9
	$c^t\hat{y}$	-23.0	-22.0	-26.0	-25.0	-24.0	-24.0	-23.0
	$\Gamma_{\mathcal{P}_2}(T)$	158297.1	114870.3	73553.1	80700.6	104861.9	120935.1	158632.6
pw-myciel4	$n_{\mathcal{P}_2}$	532.0	310.0	310.0	632.0	100.0	100.0	532.0
	$t_{\mathcal{P}_2}$	52.5	55.1	65.0	76.2	38.3	38.5	51.6
	$c^t\hat{y}$	10.0	10.0	10.0	10.0	10.0	10.0	10.0
	$\Gamma_{\mathcal{P}_2}(T)$	1802.3	1536.6	1656.5	1723.0	1270.7	1277.6	1755.2
qiu	$n_{\mathcal{P}_2}$	776.0	824.0	638.0	824.0	589.0	589.0	776.0
	$t_{\mathcal{P}_2}$	22.5	26.0	25.5	26.7	23.7	23.7	22.7
	$c^t\hat{y}$	-132.9	-132.9	-132.9	-132.9	-132.9	-132.9	-132.9
	$\Gamma_{\mathcal{P}_2}(T)$	2128.9	2063.1	1853.0	2037.4	2183.4	2174.4	2150.0
qnet1	$n_{\mathcal{P}_2}$	30.0	3.0	3.0	3.0	4.0	4.0	30.0
	$t_{\mathcal{P}_2}$	8.3	4.7	4.0	4.6	4.7	4.5	8.2
	$c^t\hat{y}$	16029.7	16029.7	16029.7	16029.7	16029.7	16029.7	16029.7
	$\Gamma_{\mathcal{P}_2}(T)$	79.7	23.3	18.6	23.1	22.5	22.9	79.6
qnet1_o	$n_{\mathcal{P}_2}$	13.0	0.0	0.0	0.0	0.0	0.0	13.0
	$t_{\mathcal{P}_2}$	7.5	2.1	2.2	2.2	2.2	2.4	6.8
	$c^t\hat{y}$	16029.7	16029.7	16029.7	16029.7	16029.7	16029.7	16029.7
	$\Gamma_{\mathcal{P}_2}(T)$	74.1	9.5	9.5	10.5	8.5	10.5	61.7
rail507	$n_{\mathcal{P}_2}$	293.0	1014.0	1014.0	1014.0	293.0	293.0	293.0
	$t_{\mathcal{P}_2}$	146.5	447.3	443.5	449.1	159.0	158.2	142.3
	$c^t\hat{y}$	174.0	174.0	174.0	174.0	174.0	174.0	174.0
	$\Gamma_{\mathcal{P}_2}(T)$	381.7	589.8	575.8	595.7	386.4	384.6	361.0
ran16x16	$n_{\mathcal{P}_2}$	17474.0	16502.0	33779.0	37899.0	83374.0	39734.0	54013.0
	$t_{\mathcal{P}_2}$	32.3	43.3	70.0	72.6	112.3	71.4	69.4
	$c^t\hat{y}$	3823.0	3823.0	3823.0	3823.0	3823.0	3823.0	3823.0
	$\Gamma_{\mathcal{P}_2}(T)$	69.0	99.2	146.3	137.4	113.9	89.0	103.8
rd-rplusc-21	$n_{\mathcal{P}_2}$	30555.0	24194.0	22595.0	20464.0	24084.0	21895.0	30589.0
	$t_{\mathcal{P}_2}$	3027.3	3031.4	3030.1	3022.3	3028.2	3027.0	3030.1
	$c^t\hat{y}$	166009.7	166227.5	165797.0	166009.7	167644.0	168174.6	166009.7
	$\Gamma_{\mathcal{P}_2}(T)$	2730.6	3757.3	2471.1	2191.2	5561.5	6109.1	2711.7
reblock67	$n_{\mathcal{P}_2}$	8158.0	8638.0	25750.0	26525.0	47596.0	21751.0	8158.0
	$t_{\mathcal{P}_2}$	58.0	55.4	108.2	104.5	157.2	105.0	57.9
	$c^t\hat{y}$	-34630648.4	-34630648.4	-34630648.4	-34630648.4	-34630648.4	-34630648.4	-34630648.4
	$\Gamma_{\mathcal{P}_2}(T)$	1653.7	838.3	853.3	955.6	1696.6	1685.5	1608.0
rgn	$n_{\mathcal{P}_2}$	0.0	0.0	0.0	0.0	0.0	0.0	0.0
	$t_{\mathcal{P}_2}$	0.1	0.4	0.4	0.4	0.3	0.3	0.2
	$c^t\hat{y}$	82.2	82.2	82.2	82.2	82.2	82.2	82.2
	$\Gamma_{\mathcal{P}_2}(T)$	7.2	20.5	20.5	19.5	22.5	22.5	15.3
rmatr100-p10	$n_{\mathcal{P}_2}$	161.0	0.0	0.0	0.0	0.0	0.0	161.0
	$t_{\mathcal{P}_2}$	65.7	76.5	69.0	76.5	67.5	77.0	65.4
	$c^t\hat{y}$	423.0	423.0	423.0	423.0	423.0	423.0	423.0
	$\Gamma_{\mathcal{P}_2}(T)$	842.1	977.5	895.4	974.2	876.6	980.6	840.0
rmatr100-p5	$n_{\mathcal{P}_2}$	98.0	0.0	0.0	0.0	0.0	0.0	0.0
	$t_{\mathcal{P}_2}$	189.2	48.1	55.4	48.1	44.1	49.5	47.3
	$c^t\hat{y}$	976.0	976.0	976.0	976.0	976.0	976.0	976.0
	$\Gamma_{\mathcal{P}_2}(T)$	1426.3	1281.4	1469.8	1282.7	1174.6	1318.7	1261.3

continued on next page

Table H.6: Instancewise results for the Improvement phase experiment, see Section 5.1.2.

		default	agg	agg l-uct	agg_{05}	l-agg	l-agg l-uct	l-uct
rmine6	$n_{\mathcal{P}_2}$	1076121.0	444842.0	974763.0	788139.0	280041.0	321144.0	323495.0
	$t_{\mathcal{P}_2}$	3600.0	1822.8	3600.0	2864.2	1155.8	1257.7	1161.8
	$c^t\hat{y}$	-457.2	-457.2	-457.2	-457.2	-457.2	-457.2	-457.2
	$\Gamma_{\mathcal{P}_2}(T)$	335.4	360.5	353.8	350.3	339.8	353.5	312.4
rocII-4-11	$n_{\mathcal{P}_2}$	40433.0	22151.0	29346.0	38557.0	4343.0	19430.0	40433.0
	$t_{\mathcal{P}_2}$	433.2	383.1	445.6	499.2	118.4	353.9	431.9
	$c^t\hat{y}$	-6.7	-6.7	-6.7	-6.7	-6.7	-6.7	-6.7
	$\Gamma_{\mathcal{P}_2}(T)$	8933.3	11746.5	10824.7	12850.8	5167.2	9120.2	8922.3
rococoC10-001000	$n_{\mathcal{P}_2}$	98574.0	115614.0	30311.0	13619.0	218079.0	48479.0	53888.0
	$t_{\mathcal{P}_2}$	647.1	822.8	354.3	238.8	1433.8	408.3	392.9
	$c^t\hat{y}$	11460.0	11460.0	11460.0	11460.0	11460.0	11460.0	11460.0
	$\Gamma_{\mathcal{P}_2}(T)$	745.3	1035.5	1233.9	848.9	1824.5	787.3	916.5
roll3000	$n_{\mathcal{P}_2}$	1285318.0	50239.0	44506.0	138637.0	34994.0	156994.0	293299.0
	$t_{\mathcal{P}_2}$	3595.5	373.6	346.0	722.5	291.2	646.0	1034.6
	$c^t\hat{y}$	12899.0	12890.0	12890.0	12890.0	12890.0	12890.0	12890.0
	$\Gamma_{\mathcal{P}_2}(T)$	866.3	769.1	511.8	886.5	608.7	441.2	486.7
rout	$n_{\mathcal{P}_2}$	12519.0	858.0	4219.0	858.0	10407.0	5161.0	6448.0
	$t_{\mathcal{P}_2}$	27.7	10.0	23.7	10.4	31.6	28.4	24.3
	$c^t\hat{y}$	1077.6	1077.6	1077.6	1077.6	1077.6	1077.6	1077.6
	$\Gamma_{\mathcal{P}_2}(T)$	156.0	115.2	111.9	105.7	172.0	155.1	190.5
satellites1-25	$n_{\mathcal{P}_2}$	1297.0	200.0	200.0	200.0	845.0	845.0	1297.0
	$t_{\mathcal{P}_2}$	475.9	341.9	342.2	342.3	474.2	486.1	477.0
	$c^t\hat{y}$	-5.0	-5.0	-5.0	-5.0	-5.0	-5.0	-5.0
	$\Gamma_{\mathcal{P}_2}(T)$	30130.0	22916.0	22943.0	22926.0	28109.0	28579.0	30226.0
set1ch	$n_{\mathcal{P}_2}$	4.0	0.0	0.0	0.0	0.0	0.0	4.0
	$t_{\mathcal{P}_2}$	1.1	1.1	1.0	1.1	0.9	0.7	0.9
	$c^t\hat{y}$	54537.8	54537.8	54537.8	54537.8	54537.8	54537.8	54537.8
	$\Gamma_{\mathcal{P}_2}(T)$	24.3	25.3	24.7	27.2	20.3	13.7	21.3
seymour	$n_{\mathcal{P}_2}$	65643.0	63487.0	54138.0	62953.0	60203.0	62467.0	67338.0
	$t_{\mathcal{P}_2}$	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0	3600.0
	$c^t\hat{y}$	426.0	424.0	424.0	424.0	424.0	426.0	424.0
	$\Gamma_{\mathcal{P}_2}(T)$	3332.3	2886.4	2838.7	2886.5	2508.3	3731.7	2136.0
sp97ar	$n_{\mathcal{P}_2}$	2217.0	1860.0	2128.0	2002.0	1866.0	2022.0	2385.0
	$t_{\mathcal{P}_2}$	3598.4	3598.3	3598.4	3598.1	3598.3	3598.3	3598.3
	$c^t\hat{y}$	710063477.9	685616215.2	674738738.8	681876238.1	687058465.1	693990313.0	713253304.8
	$\Gamma_{\mathcal{P}_2}(T)$	34864.4	16031.0	11543.4	12960.1	18511.9	20612.7	36885.6
sp98ic	$n_{\mathcal{P}_2}$	52188.0	55923.0	45032.0	58237.0	68697.0	59754.0	60231.0
	$t_{\mathcal{P}_2}$	3598.4	3598.2	3598.5	3598.5	3598.5	3598.5	3598.5
	$c^t\hat{y}$	461315936.0	452326876.8	450686589.4	452153055.0	457372164.0	450852689.3	453891327.8
	$\Gamma_{\mathcal{P}_2}(T)$	12605.0	4170.0	6595.4	3624.2	7961.1	3069.0	6748.5
sp98ir	$n_{\mathcal{P}_2}$	6877.0	1692.0	1216.0	1153.0	1222.0	2639.0	8148.0
	$t_{\mathcal{P}_2}$	95.4	61.8	55.3	50.1	52.2	79.7	103.6
	$c^t\hat{y}$	219676790.4	219676790.4	219676790.4	219676790.4	219676790.4	219676790.4	219676790.4
	$\Gamma_{\mathcal{P}_2}(T)$	266.0	240.4	218.0	212.2	208.9	220.4	267.3
stein45	$n_{\mathcal{P}_2}$	1399.0	2739.0	201.0	2739.0	2045.0	2045.0	1399.0
	$t_{\mathcal{P}_2}$	1.4	2.3	1.4	2.8	2.3	2.2	1.4
	$c^t\hat{y}$	30.0	30.0	30.0	30.0	30.0	30.0	30.0
	$\Gamma_{\mathcal{P}_2}(T)$	4.7	7.0	5.1	9.8	8.6	8.9	5.4
swath	$n_{\mathcal{P}_2}$	564580.0	246660.0	505684.0	129245.0	42489.0	77464.0	545394.0
	$t_{\mathcal{P}_2}$	3599.8	1927.8	3599.9	1184.5	514.1	825.2	3599.8
	$c^t\hat{y}$	472.6	467.4	478.0	467.4	467.4	467.4	483.8
	$\Gamma_{\mathcal{P}_2}(T)$	10765.5	6908.1	10872.3	5336.8	3742.7	3366.4	16144.6
tanglegram1	$n_{\mathcal{P}_2}$	29.0	26.0	26.0	26.0	29.0	29.0	29.0
	$t_{\mathcal{P}_2}$	1059.3	1023.7	1026.3	1022.1	1079.8	1076.1	1064.3
	$c^t\hat{y}$	5182.0	5182.0	5182.0	5182.0	5182.0	5182.0	5182.0
	$\Gamma_{\mathcal{P}_2}(T)$	15585.5	16195.5	16227.7	16183.7	15933.2	15898.6	15623.5
tanglegram2	$n_{\mathcal{P}_2}$	3.0	2.0	2.0	2.0	2.0	2.0	3.0
	$t_{\mathcal{P}_2}$	10.4	8.3	8.3	8.3	8.2	8.2	10.3
	$c^t\hat{y}$	443.0	443.0	443.0	443.0	443.0	443.0	443.0
	$\Gamma_{\mathcal{P}_2}(T)$	704.1	556.4	556.5	562.1	547.7	547.8	697.5
timtab1	$n_{\mathcal{P}_2}$	68220.0	201503.0	98553.0	17768.0	65269.0	45101.0	68220.0
	$t_{\mathcal{P}_2}$	39.3	136.6	78.5	25.3	52.9	48.8	37.6
	$c^t\hat{y}$	764772.0	764772.0	764772.0	764772.0	764772.0	764772.0	764772.0
	$\Gamma_{\mathcal{P}_2}(T)$	282.0	201.3	142.3	176.0	207.8	262.6	263.7
timtab2	$n_{\mathcal{P}_2}$	4167352.0	3616261.0	3804951.0	3356244.0	3739038.0	3819528.0	4065888.0
	$t_{\mathcal{P}_2}$	3578.0	3577.7	3579.7	3280.0	3579.4	3579.2	3576.7

continued on next page

Table H.6: Instancewise results for the Improvement phase experiment, see Section 5.1.2.

		default	agg	agg l-uct	agg_{05}	l-agg	l-agg l-uct	l-uct
	$c^t\hat{y}$	1136721.0	1145403.0	1150047.0	1096557.0	1146606.0	1115393.0	1139205.0
	$\Gamma_{\mathcal{P}_2}(T)$	24388.5	19433.4	25848.0	12650.4	26426.2	18974.6	27150.3
tr12-30	$n_{\mathcal{P}_2}$	186641.0	48399.0	227530.0	348284.0	577.0	66111.0	488956.0
	$t_{\mathcal{P}_2}$	280.0	145.6	426.1	590.2	29.5	157.8	677.1
	$c^t\hat{y}$	130596.0	130596.0	130596.0	130596.0	130596.0	130596.0	130596.0
	$\Gamma_{\mathcal{P}_2}(T)$	50.9	69.3	77.4	85.2	72.4	64.3	55.9
triptim1	$n_{\mathcal{P}_2}$	14.0	14.0	14.0	14.0	14.0	14.0	14.0
	$t_{\mathcal{P}_2}$	842.4	908.2	896.2	889.5	838.2	839.5	834.9
	$c^t\hat{y}$	22.9	22.9	22.9	22.9	22.9	22.9	22.9
	$\Gamma_{\mathcal{P}_2}(T)$	1.5	43.6	27.0	42.7	33.3	35.3	6.1
unitcal_7	$n_{\mathcal{P}_2}$	1964.0	4108.0	2133.0	3543.0	165.0	4241.0	1110.0
	$t_{\mathcal{P}_2}$	363.5	867.0	936.2	901.7	301.9	1098.5	280.9
	$c^t\hat{y}$	19635558.2	19635558.2	19635558.2	19635558.2	19635558.2	19635558.2	19635558.2
	$\Gamma_{\mathcal{P}_2}(T)$	190.5	346.1	325.7	307.0	256.6	273.1	190.0
vpphard	$n_{\mathcal{P}_2}$	2690.0	941.0	1660.0	941.0	1376.0	1078.0	2689.0
	$t_{\mathcal{P}_2}$	3339.7	3341.9	3340.0	3341.7	3341.9	3343.2	3341.6
	$c^t\hat{y}$	9.0	12.0	8.0	12.0	9.0	22.0	9.0
	$\Gamma_{\mathcal{P}_2}(T)$	216253.8	239513.0	228421.2	239532.5	207614.1	275050.0	216303.5
zib54-UUE	$n_{\mathcal{P}_2}$	2543.0	1538.0	1401.0	1318.0	2333.0	4788.0	1028.0
	$t_{\mathcal{P}_2}$	77.1	111.0	93.1	78.9	92.0	125.8	51.9
	$c^t\hat{y}$	10334015.8	10334015.8	10334015.8	10334015.8	10334015.8	10334015.8	10334015.8
	$\Gamma_{\mathcal{P}_2}(T)$	1066.3	1287.3	1373.8	1312.3	1153.7	1091.5	1067.3

Table H.7: Instancewise results of the experiment in Section 5.1.3. Each column shows the results of a different SCIP-setting during the Proof phase. For each instance, we show the solving time t (sec), the solving time during the Proof phase $t_{\mathcal{P}_3}$, and the number of branch-and-bound nodes during the Proof phase denoted by $n_{\mathcal{P}_3}$.

		weights	combined	default	dfsHeuroff	error based	sepa
10teams	t (sec)	25.5	25.0	25.5	26.7	25.8	26.1
	$t_{\mathcal{P}_3}$	0.5	0.0	0.4	0.5	0.9	0.0
	$n_{\mathcal{P}_3}$	75.0	19.0	75.0	115.0	126.0	7.0
30n20b8	t (sec)	182.4	135.4	185.6	238.1	156.2	137.9
	$t_{\mathcal{P}_3}$	99.3	52.6	100.8	154.2	73.3	54.2
	$n_{\mathcal{P}_3}$	13.0	3.0	13.0	58.0	10.0	3.0
aflow30a	t (sec)	13.7	13.4	14.1	13.9	16.2	13.9
	$t_{\mathcal{P}_3}$	3.4	3.1	3.3	3.4	5.5	3.3
	$n_{\mathcal{P}_3}$	1997.0	1673.0	1993.0	1979.0	1465.0	1717.0
aflow40b	t (sec)	891.0	520.0	928.6	860.3	963.5	574.2
	$t_{\mathcal{P}_3}$	764.0	395.0	800.6	733.3	836.5	449.2
	$n_{\mathcal{P}_3}$	142369.0	78894.0	145556.0	153515.0	153666.0	76843.0
air04	t (sec)	71.4	71.3	70.6	70.7	71.4	70.5
	$t_{\mathcal{P}_3}$	3.2	3.2	2.2	2.6	2.4	2.2
	$n_{\mathcal{P}_3}$	36.0	36.0	34.0	34.0	36.0	34.0
air05	t (sec)	38.2	37.9	37.5	38.2	37.9	37.2
	$t_{\mathcal{P}_3}$	2.1	2.1	2.2	2.1	2.3	2.1
	$n_{\mathcal{P}_3}$	50.0	48.0	52.0	50.0	48.0	52.0
app1-2	t (sec)	1128.4	1063.2	1121.8	1024.3	1040.3	1198.1
	$t_{\mathcal{P}_3}$	503.4	426.2	499.8	407.3	423.3	578.1
	$n_{\mathcal{P}_3}$	423.0	426.0	423.0	436.0	368.0	415.0
arki001	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
	$t_{\mathcal{P}_3}$	6789.0	6790.0	6796.0	6786.0	6790.0	6789.0
	$n_{\mathcal{P}_3}$	1016463.0	882508.0	1108529.0	906787.0	882182.0	987325.0
bell3a	t (sec)	6.1	4.3	6.7	4.0	5.6	6.0
	$t_{\mathcal{P}_3}$	6.0	4.2	6.6	4.0	5.5	5.9
	$n_{\mathcal{P}_3}$	22486.0	22578.0	22486.0	22586.0	22936.0	23067.0
bell5	t (sec)	0.8	0.6	0.4	0.4	0.6	0.8
	$t_{\mathcal{P}_3}$	0.7	0.5	0.3	0.3	0.5	0.7
	$n_{\mathcal{P}_3}$	1121.0	1121.0	1121.0	1158.0	1167.0	1097.0

continued on next page

Table H.7: Instancewise results for the Proof phase experiment, see Section 5.1.3.

		weights	combined	default	dfsHeuroff	error based	sepa
biella1	t (sec)	797.0	767.2	784.3	756.5	741.1	791.7
	$t_{\mathcal{P}_3}$	214.0	185.2	200.3	173.5	158.1	207.7
	$n_{\mathcal{P}_3}$	873.0	777.0	761.0	804.0	709.0	774.0
bienst2	t (sec)	300.1	301.3	298.1	283.8	307.1	335.2
	$t_{\mathcal{P}_3}$	210.3	214.4	210.7	195.7	219.5	248.0
	$n_{\mathcal{P}_3}$	81091.0	79287.0	80632.0	81336.0	80728.0	76638.0
binkar10_1	t (sec)	187.9	136.5	179.0	134.3	170.8	176.0
	$t_{\mathcal{P}_3}$	182.5	131.0	173.6	128.1	165.4	170.3
	$n_{\mathcal{P}_3}$	139140.0	116431.0	138055.0	118764.0	112987.0	122102.0
blend2	t (sec)	0.6	1.1	0.6	0.8	1.1	1.1
	$t_{\mathcal{P}_3}$	0.2	0.3	0.2	0.4	0.3	0.3
	$n_{\mathcal{P}_3}$	343.0	339.0	344.0	401.0	198.0	322.0
cap6000	t (sec)	2.7	2.8	2.7	2.4	2.9	3.0
	$t_{\mathcal{P}_3}$	0.5	0.4	0.5	0.3	0.5	0.8
	$n_{\mathcal{P}_3}$	2189.0	2154.0	2176.0	2150.0	1133.0	2174.0
cov1075	t (sec)	7200.0	6968.9	7200.0	6778.5	7200.0	7200.0
	$t_{\mathcal{P}_3}$	7185.0	6953.3	7184.7	6763.4	7185.1	7184.8
	$n_{\mathcal{P}_3}$	1559708.0	1635054.0	1564904.0	1590516.0	1551384.0	1556693.0
csched010	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
	$t_{\mathcal{P}_3}$	6491.0	6490.0	6506.0	6487.0	6491.0	6490.0
	$n_{\mathcal{P}_3}$	840645.0	883405.0	851244.0	882829.0	783082.0	806194.0
danoint	t (sec)	5190.9	4281.1	5090.7	4230.0	5100.3	5145.0
	$t_{\mathcal{P}_3}$	5157.0	4247.5	5057.6	4196.6	5066.8	5111.7
	$n_{\mathcal{P}_3}$	1054624.0	990088.0	1048903.0	968642.0	1027955.0	1021086.0
dcmulti	t (sec)	1.6	1.5	1.4	1.6	1.7	1.4
	$t_{\mathcal{P}_3}$	0.0	0.0	0.0	0.0	0.0	0.0
	$n_{\mathcal{P}_3}$	15.0	11.0	15.0	15.0	5.0	11.0
dfn-gwin-UUM	t (sec)	139.3	118.7	140.6	122.9	142.2	119.1
	$t_{\mathcal{P}_3}$	113.0	92.5	114.4	96.7	115.9	93.1
	$n_{\mathcal{P}_3}$	63050.0	59988.0	64526.0	64852.0	69707.0	44884.0
eil33-2	t (sec)	53.4	79.8	52.6	52.5	59.3	78.8
	$t_{\mathcal{P}_3}$	12.7	39.1	12.0	11.5	19.0	38.2
	$n_{\mathcal{P}_3}$	446.0	974.0	372.0	372.0	2056.0	914.0
eilB101	t (sec)	443.5	445.4	436.4	441.6	444.6	444.8
	$t_{\mathcal{P}_3}$	14.5	14.4	14.4	14.6	13.6	14.8
	$n_{\mathcal{P}_3}$	223.0	205.0	223.0	205.0	433.0	223.0
enlight13	t (sec)	8.9	119.9	8.6	660.7	343.7	4.2
	$t_{\mathcal{P}_3}$	8.9	119.9	8.6	660.7	343.7	4.2
	$n_{\mathcal{P}_3}$	13478.0	270889.0	13478.0	1938511.0	797246.0	3593.0
fast0507	t (sec)	574.8	574.3	578.5	574.8	568.2	574.1
	$t_{\mathcal{P}_3}$	81.8	80.3	81.5	81.8	74.2	80.1
	$n_{\mathcal{P}_3}$	577.0	549.0	577.0	543.0	511.0	565.0
fiber	t (sec)	2.2	3.3	1.4	1.4	1.6	2.9
	$t_{\mathcal{P}_3}$	0.9	2.1	0.5	0.4	0.7	1.8
	$n_{\mathcal{P}_3}$	7.0	4.0	7.0	10.0	7.0	3.0
fixnet6	t (sec)	3.5	7.2	3.2	2.9	3.1	6.2
	$t_{\mathcal{P}_3}$	1.9	5.5	1.3	1.3	1.6	4.3
	$n_{\mathcal{P}_3}$	8.0	7.0	8.0	8.0	8.0	6.0
flugpl	t (sec)	0.1	0.0	0.0	0.0	0.1	0.0
	$t_{\mathcal{P}_3}$	0.0	0.0	0.0	0.0	0.0	0.0
	$n_{\mathcal{P}_3}$	235.0	259.0	235.0	259.0	47.0	235.0
gesa2-o	t (sec)	1.4	1.4	1.4	1.4	1.2	1.2
	$t_{\mathcal{P}_3}$	0.0	0.0	0.0	0.0	0.0	0.0
	$n_{\mathcal{P}_3}$	3.0	3.0	3.0	3.0	3.0	3.0
gesa3	t (sec)	1.8	2.8	1.7	1.8	1.5	2.9
	$t_{\mathcal{P}_3}$	0.3	1.1	0.3	0.2	0.2	1.5
	$n_{\mathcal{P}_3}$	6.0	5.0	6.0	6.0	12.0	6.0
gesa3_o	t (sec)	1.8	1.6	1.6	1.7	1.8	1.9
	$t_{\mathcal{P}_3}$	0.0	0.0	0.0	0.0	0.0	0.0
	$n_{\mathcal{P}_3}$	5.0	5.0	5.0	5.0	5.0	5.0
harp2	t (sec)	3684.3	3595.1	3687.0	3618.7	3734.0	3725.6
	$t_{\mathcal{P}_3}$	3633.8	3545.2	3636.1	3568.7	3683.6	3674.8
	$n_{\mathcal{P}_3}$	12504398.0	12259818.0	12568370.0	12342086.0	12080257.0	12564197.0
iis-100-0-cov	t (sec)	1669.1	1244.7	1665.4	1264.5	1663.5	1652.8
	$t_{\mathcal{P}_3}$	1619.7	1195.8	1616.2	1215.1	1614.7	1603.8
	$n_{\mathcal{P}_3}$	101696.0	83584.0	102216.0	83150.0	105996.0	101316.0
iis-bupa-cov	t (sec)	6302.1	5190.8	6160.1	5434.7	5935.7	6129.0
	$t_{\mathcal{P}_3}$	6151.1	5042.8	6012.1	5283.7	5788.7	5981.0
	$n_{\mathcal{P}_3}$	180628.0	167214.0	181708.0	169198.0	183144.0	181708.0

continued on next page

Table H.7: Instancewise results for the Proof phase experiment, see Section 5.1.3.

		weights	combined	default	dfsHeuroff	error based	sepa
iis-pima-cov	t (sec)	1387.4	1378.4	1384.7	1383.8	1463.2	1387.3
	$t_{\mathcal{P}_3}$	154.4	152.4	154.7	152.8	229.2	155.3
	$n_{\mathcal{P}_3}$	3203.0	3179.0	3203.0	3179.0	3167.0	3203.0
l152lav	t (sec)	2.5	2.3	2.7	2.9	2.6	2.8
	$t_{\mathcal{P}_3}$	0.1	0.1	0.1	0.0	0.1	0.1
	$n_{\mathcal{P}_3}$	5.0	5.0	5.0	5.0	5.0	5.0
lseu	t (sec)	0.4	0.8	0.6	0.7	0.6	0.8
	$t_{\mathcal{P}_3}$	0.0	0.0	0.0	0.0	0.0	0.0
	$n_{\mathcal{P}_3}$	53.0	51.0	53.0	54.0	17.0	51.0
map18	t (sec)	421.5	426.5	431.0	434.1	469.3	428.6
	$t_{\mathcal{P}_3}$	147.5	153.5	150.0	154.1	191.3	152.6
	$n_{\mathcal{P}_3}$	309.0	287.0	309.0	289.0	239.0	279.0
map20	t (sec)	333.7	335.2	333.0	288.2	389.9	440.1
	$t_{\mathcal{P}_3}$	277.7	278.3	277.5	232.8	334.3	382.8
	$n_{\mathcal{P}_3}$	298.0	318.0	298.0	330.0	270.0	364.0
mas74	t (sec)	593.8	481.3	566.7	481.2	591.3	587.4
	$t_{\mathcal{P}_3}$	572.4	459.8	547.9	459.2	569.0	566.0
	$n_{\mathcal{P}_3}$	2791971.0	2764370.0	2797757.0	2758737.0	2785374.0	2794939.0
mas76	t (sec)	68.3	50.4	67.5	45.1	66.5	79.4
	$t_{\mathcal{P}_3}$	67.6	49.8	66.9	44.4	65.9	78.7
	$n_{\mathcal{P}_3}$	404938.0	350248.0	404938.0	362355.0	392712.0	398256.0
mcsched	t (sec)	212.3	211.7	212.7	211.3	225.1	215.7
	$t_{\mathcal{P}_3}$	37.3	37.7	37.7	37.3	50.1	37.7
	$n_{\mathcal{P}_3}$	3366.0	3378.0	3366.0	3378.0	2838.0	3366.0
mik-250-1-100-1	t (sec)	371.0	238.6	364.5	228.5	284.2	184.2
	$t_{\mathcal{P}_3}$	370.3	238.0	364.2	227.9	283.6	183.6
	$n_{\mathcal{P}_3}$	943439.0	595706.0	943439.0	613729.0	770065.0	381395.0
mine-166-5	t (sec)	30.5	30.7	30.6	31.1	30.8	30.8
	$t_{\mathcal{P}_3}$	0.6	0.5	0.5	0.5	0.9	0.5
	$n_{\mathcal{P}_3}$	352.0	280.0	328.0	296.0	198.0	318.0
mine-90-10	t (sec)	258.1	242.7	256.0	268.4	307.8	250.4
	$t_{\mathcal{P}_3}$	82.1	63.7	83.0	88.4	130.8	72.4
	$n_{\mathcal{P}_3}$	32627.0	26514.0	33091.0	37170.0	18490.0	27506.0
misc03	t (sec)	1.6	1.4	1.2	1.7	1.5	1.3
	$t_{\mathcal{P}_3}$	1.1	0.9	0.9	1.2	1.0	0.8
	$n_{\mathcal{P}_3}$	137.0	218.0	137.0	182.0	53.0	120.0
misc07	t (sec)	15.5	14.5	14.5	11.5	13.0	16.3
	$t_{\mathcal{P}_3}$	14.2	13.4	13.1	10.1	11.4	15.1
	$n_{\mathcal{P}_3}$	24294.0	20859.0	21666.0	18052.0	9572.0	20989.0
mod008	t (sec)	0.7	1.3	1.1	1.1	1.2	0.7
	$t_{\mathcal{P}_3}$	0.0	0.2	0.0	0.0	0.1	0.1
	$n_{\mathcal{P}_3}$	6.0	3.0	6.0	6.0	6.0	3.0
mod010	t (sec)	0.6	2.7	0.8	0.6	0.8	2.4
	$t_{\mathcal{P}_3}$	0.0	2.0	0.1	0.0	0.1	2.0
	$n_{\mathcal{P}_3}$	1.0	1.0	1.0	1.0	1.0	1.0
mod011	t (sec)	177.2	151.2	177.0	150.4	174.7	169.3
	$t_{\mathcal{P}_3}$	73.2	47.2	73.0	46.4	70.7	65.3
	$n_{\mathcal{P}_3}$	905.0	697.0	905.0	811.0	481.0	787.0
modglob	t (sec)	1.3	1.3	1.6	1.5	1.6	1.6
	$t_{\mathcal{P}_3}$	0.4	0.2	0.3	0.2	0.5	0.3
	$n_{\mathcal{P}_3}$	643.0	477.0	643.0	641.0	165.0	485.0
mspp16	t (sec)	2591.8	5701.5	2576.9	2675.3	2838.8	6959.4
	$t_{\mathcal{P}_3}$	2097.5	5205.7	2086.8	2184.1	2348.9	6468.2
	$n_{\mathcal{P}_3}$	50.0	68.0	50.0	48.0	138.0	291.0
mzzv11	t (sec)	269.1	259.2	267.0	267.7	253.7	257.1
	$t_{\mathcal{P}_3}$	72.1	60.2	69.0	67.7	54.7	59.1
	$n_{\mathcal{P}_3}$	958.0	870.0	961.0	691.0	869.0	786.0
mzzv42z	t (sec)	338.9	337.4	338.4	339.0	338.3	338.6
	$t_{\mathcal{P}_3}$	8.9	8.4	9.4	9.0	9.3	8.6
	$n_{\mathcal{P}_3}$	21.0	23.0	21.0	21.0	29.0	23.0
n4-3	t (sec)	541.4	463.7	543.1	480.1	559.7	524.8
	$t_{\mathcal{P}_3}$	479.5	399.1	481.1	417.6	497.6	462.9
	$n_{\mathcal{P}_3}$	31303.0	26825.0	31297.0	30079.0	30651.0	27721.0
neos-1109824	t (sec)	158.0	151.0	156.0	272.0	171.4	139.3
	$t_{\mathcal{P}_3}$	148.2	141.2	146.2	262.1	161.6	129.4
	$n_{\mathcal{P}_3}$	21655.0	17160.0	21910.0	43026.0	22075.0	14694.0
neos-1337307	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
	$t_{\mathcal{P}_3}$	6881.0	6879.0	6882.0	6882.0	6883.0	6880.0
	$n_{\mathcal{P}_3}$	365844.0	533044.0	368446.0	518018.0	240327.0	351008.0

continued on next page

Table H.7: Instancewise results for the Proof phase experiment, see Section 5.1.3.

		weights	combined	default	dfsHeuroff	error based	sepa
neos-1396125	t (sec)	774.3	710.1	768.0	755.9	917.5	721.8
	$t_{\mathcal{P}_3}$	500.3	436.1	496.0	482.9	643.5	448.8
	$n_{\mathcal{P}_3}$	56975.0	48551.0	56834.0	56232.0	40550.0	48157.0
neos-476283	t (sec)	276.0	271.3	276.9	271.7	268.2	276.6
	$t_{\mathcal{P}_3}$	24.0	19.3	23.9	18.7	16.2	24.6
	$n_{\mathcal{P}_3}$	397.0	371.0	389.0	349.0	99.0	389.0
neos-686190	t (sec)	96.0	97.2	96.4	94.7	101.9	96.5
	$t_{\mathcal{P}_3}$	13.4	13.7	14.0	12.9	18.7	13.9
	$n_{\mathcal{P}_3}$	1892.0	1883.0	1875.0	1853.0	1384.0	1903.0
neos-916792	t (sec)	411.0	354.8	406.2	361.0	408.7	482.1
	$t_{\mathcal{P}_3}$	202.0	150.8	200.2	153.0	199.7	273.1
	$n_{\mathcal{P}_3}$	80891.0	81953.0	80533.0	82969.0	77223.0	82943.0
neos13	t (sec)	1543.3	1547.0	1541.7	1583.8	1558.8	1547.6
	$t_{\mathcal{P}_3}$	64.3	100.0	64.7	100.8	65.8	64.6
	$n_{\mathcal{P}_3}$	458.0	458.0	458.0	458.0	316.0	458.0
neos18	t (sec)	28.5	30.4	32.0	29.8	35.1	57.7
	$t_{\mathcal{P}_3}$	15.0	17.0	18.5	16.2	21.6	44.2
	$n_{\mathcal{P}_3}$	4956.0	4339.0	6437.0	5926.0	4952.0	7408.0
net12	t (sec)	2301.6	2675.7	2531.6	2653.1	2978.9	2702.9
	$t_{\mathcal{P}_3}$	1254.6	1627.7	1491.6	1610.1	1924.9	1658.9
	$n_{\mathcal{P}_3}$	1629.0	1758.0	2599.0	2372.0	2440.0	2321.0
newdano	t (sec)	3596.3	3698.6	3579.3	3438.4	3556.4	5561.1
	$t_{\mathcal{P}_3}$	2430.3	2543.6	2418.3	2273.4	2395.4	4399.1
	$n_{\mathcal{P}_3}$	1549407.0	1455498.0	1545121.0	1543344.0	1547506.0	1017309.0
noswot	t (sec)	197.1	174.4	177.0	173.3	163.7	204.9
	$t_{\mathcal{P}_3}$	196.2	173.5	176.0	172.4	162.8	203.8
	$n_{\mathcal{P}_3}$	928902.0	436695.0	829282.0	895797.0	553378.0	489560.0
ns1208400	t (sec)	1879.9	1880.3	1870.7	1881.3	1879.0	1892.4
	$t_{\mathcal{P}_3}$	61.9	63.3	61.7	62.4	61.3	61.6
	$n_{\mathcal{P}_3}$	91.0	91.0	91.0	91.0	91.0	91.0
ns1688347	t (sec)	731.5	718.3	739.7	718.4	737.5	740.7
	$t_{\mathcal{P}_3}$	57.5	44.3	69.7	37.4	61.5	64.7
	$n_{\mathcal{P}_3}$	1862.0	1735.0	2901.0	2306.0	974.0	1795.0
ns1830653	t (sec)	446.9	438.9	442.9	433.6	587.4	461.6
	$t_{\mathcal{P}_3}$	230.9	223.9	227.9	218.6	371.4	243.6
	$n_{\mathcal{P}_3}$	27998.0	23985.0	28115.0	26684.0	27965.0	25714.0
nw04	t (sec)	24.8	24.9	24.3	23.7	24.2	25.7
	$t_{\mathcal{P}_3}$	2.1	2.6	2.0	1.3	1.9	3.2
	$n_{\mathcal{P}_3}$	10.0	5.0	10.0	11.0	10.0	5.0
opm2-z7-s2	t (sec)	788.7	788.8	790.4	804.6	789.8	801.8
	$t_{\mathcal{P}_3}$	2.7	2.8	2.4	2.6	1.8	7.8
	$n_{\mathcal{P}_3}$	38.0	38.0	36.0	36.0	26.0	36.0
p0201	t (sec)	1.6	1.6	1.6	1.8	1.6	1.8
	$t_{\mathcal{P}_3}$	0.3	0.3	0.3	0.4	0.3	0.3
	$n_{\mathcal{P}_3}$	26.0	24.0	32.0	38.0	10.0	30.0
p0282	t (sec)	0.3	0.4	0.3	0.4	0.5	0.3
	$t_{\mathcal{P}_3}$	0.0	0.1	0.0	0.0	0.0	0.0
	$n_{\mathcal{P}_3}$	2.0	0.0	2.0	1.0	2.0	0.0
pg5_34	t (sec)	1285.2	1273.9	1292.9	1279.9	1282.2	1279.2
	$t_{\mathcal{P}_3}$	112.2	104.9	111.9	107.9	111.2	108.2
	$n_{\mathcal{P}_3}$	49529.0	46145.0	49529.0	48719.0	43701.0	46335.0
pigeon-10	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
	$t_{\mathcal{P}_3}$	7195.0	7194.9	7194.8	7195.0	7194.9	7195.3
	$n_{\mathcal{P}_3}$	17293610.0	10244034.0	17257299.0	17972276.0	17348948.0	9903541.0
pk1	t (sec)	64.8	53.6	63.7	54.5	63.7	65.7
	$t_{\mathcal{P}_3}$	55.0	44.2	54.7	44.6	54.2	55.9
	$n_{\mathcal{P}_3}$	268983.0	265669.0	268661.0	265801.0	271945.0	268935.0
pp08a	t (sec)	1.1	1.4	1.2	1.3	1.2	1.4
	$t_{\mathcal{P}_3}$	0.0	0.0	0.0	0.0	0.1	0.1
	$n_{\mathcal{P}_3}$	88.0	86.0	88.0	88.0	42.0	86.0
pp08aCUTS	t (sec)	1.2	1.3	1.1	1.3	1.2	1.3
	$t_{\mathcal{P}_3}$	0.1	0.1	0.0	0.1	0.1	0.1
	$n_{\mathcal{P}_3}$	110.0	104.0	110.0	112.0	30.0	104.0
pw-myciel4	t (sec)	3674.6	2192.0	3531.1	2418.4	7200.0	5624.6
	$t_{\mathcal{P}_3}$	3622.9	2140.3	3479.1	2365.9	7146.5	5573.0
	$n_{\mathcal{P}_3}$	637792.0	367822.0	712180.0	481359.0	910448.0	455520.0
qiu	t (sec)	79.8	80.0	80.1	77.3	80.6	82.3
	$t_{\mathcal{P}_3}$	57.5	56.4	57.7	54.8	58.1	59.4
	$n_{\mathcal{P}_3}$	11807.0	11841.0	11827.0	11853.0	11759.0	11957.0

continued on next page

Table H.7: Instancewise results for the Proof phase experiment, see Section 5.1.3.

		weights	combined	default	dfsHeuroff	error based	sepa
qnet1	t (sec)	8.4	8.4	8.4	8.4	8.3	8.7
	$t_{\mathcal{P}_3}$	0.0	0.0	0.0	0.0	0.0	0.0
	$n_{\mathcal{P}_3}$	5.0	5.0	5.0	5.0	5.0	5.0
qnet1_o	t (sec)	7.3	7.0	7.0	7.1	7.0	6.8
	$t_{\mathcal{P}_3}$	0.0	0.0	0.0	0.0	0.0	0.0
	$n_{\mathcal{P}_3}$	2.0	2.0	2.0	2.0	2.0	2.0
rail507	t (sec)	238.9	236.9	244.1	234.5	238.6	242.0
	$t_{\mathcal{P}_3}$	85.9	82.9	90.1	81.5	86.6	89.0
	$n_{\mathcal{P}_3}$	505.0	561.0	505.0	507.0	523.0	525.0
ran16x16	t (sec)	292.1	238.2	293.6	275.9	286.5	284.9
	$t_{\mathcal{P}_3}$	260.6	206.5	261.7	244.2	254.2	253.1
	$n_{\mathcal{P}_3}$	347677.0	289001.0	350547.0	351487.0	359019.0	339289.0
reblock67	t (sec)	274.8	174.0	250.0	232.9	301.3	187.7
	$t_{\mathcal{P}_3}$	217.1	115.2	191.1	175.7	243.8	129.8
	$n_{\mathcal{P}_3}$	116413.0	48914.0	101506.0	109362.0	84466.0	52170.0
rentacar	t (sec)	2.6	3.7	2.8	2.9	2.5	3.8
	$t_{\mathcal{P}_3}$	1.4	2.2	1.6	1.5	1.4	2.4
	$n_{\mathcal{P}_3}$	3.0	3.0	3.0	1.0	3.0	3.0
rmatr100-p10	t (sec)	135.7	130.6	134.9	131.1	147.8	135.1
	$t_{\mathcal{P}_3}$	68.6	64.2	68.5	64.8	81.4	68.6
	$n_{\mathcal{P}_3}$	689.0	747.0	689.0	747.0	645.0	689.0
rmatr100-p5	t (sec)	302.8	284.7	310.7	281.3	330.6	303.3
	$t_{\mathcal{P}_3}$	119.8	100.7	122.7	98.3	146.6	120.3
	$n_{\mathcal{P}_3}$	321.0	325.0	321.0	325.0	213.0	321.0
rmine6	t (sec)	6118.2	6043.8	6078.4	6055.2	6056.7	6085.1
	$t_{\mathcal{P}_3}$	1756.2	1687.8	1750.4	1689.2	1723.7	1752.1
	$n_{\mathcal{P}_3}$	662569.0	648455.0	664491.0	665141.0	681635.0	648509.0
rocII-4-11	t (sec)	465.3	461.2	466.2	464.3	464.4	467.1
	$t_{\mathcal{P}_3}$	0.3	0.2	0.2	0.3	1.4	0.1
	$n_{\mathcal{P}_3}$	39.0	36.0	32.0	33.0	11.0	32.0
rococoC10-001000	t (sec)	1217.4	1114.7	1224.9	1166.6	1451.2	1156.7
	$t_{\mathcal{P}_3}$	566.4	463.7	578.9	521.6	805.2	510.7
	$n_{\mathcal{P}_3}$	100925.0	64750.0	104626.0	100571.0	91682.0	64687.0
rout	t (sec)	39.0	38.1	38.8	38.5	51.9	39.0
	$t_{\mathcal{P}_3}$	11.3	10.3	11.6	10.9	24.2	11.1
	$n_{\mathcal{P}_3}$	13945.0	12868.0	14144.0	14764.0	9735.0	12629.0
satellites1-25	t (sec)	551.2	522.0	660.5	568.1	562.9	534.0
	$t_{\mathcal{P}_3}$	64.2	34.0	172.5	81.1	75.9	46.0
	$n_{\mathcal{P}_3}$	619.0	139.0	1766.0	1108.0	1160.0	331.0
set1ch	t (sec)	0.9	0.9	0.7	0.9	0.9	1.1
	$t_{\mathcal{P}_3}$	0.0	0.0	0.0	0.0	0.0	0.0
	$n_{\mathcal{P}_3}$	4.0	4.0	4.0	4.0	4.0	4.0
sp98ir	t (sec)	106.5	104.1	107.6	107.1	110.3	107.6
	$t_{\mathcal{P}_3}$	9.6	7.8	9.8	8.3	13.3	9.8
	$n_{\mathcal{P}_3}$	1322.0	1344.0	1332.0	1356.0	642.0	1324.0
stein27	t (sec)	0.9	1.1	1.1	1.1	0.9	1.2
	$t_{\mathcal{P}_3}$	0.9	1.1	1.1	1.1	0.9	1.2
	$n_{\mathcal{P}_3}$	3904.0	3972.0	3904.0	4064.0	4074.0	3958.0
stein45	t (sec)	12.6	11.7	12.7	11.1	12.9	12.7
	$t_{\mathcal{P}_3}$	11.2	10.5	11.1	9.7	11.4	11.5
	$n_{\mathcal{P}_3}$	45885.0	48937.0	45953.0	47399.0	48689.0	46985.0
tanglegram1	t (sec)	1142.9	1143.1	1136.0	1167.4	1135.5	1167.1
	$t_{\mathcal{P}_3}$	68.9	68.1	69.0	72.4	68.5	70.1
	$n_{\mathcal{P}_3}$	8.0	8.0	8.0	8.0	8.0	8.0
tanglegram2	t (sec)	15.2	14.6	14.6	14.6	15.3	14.5
	$t_{\mathcal{P}_3}$	4.4	4.4	4.4	4.3	4.5	4.4
	$n_{\mathcal{P}_3}$	2.0	2.0	2.0	2.0	2.0	2.0
timtab1	t (sec)	394.8	388.8	391.9	383.7	412.9	391.1
	$t_{\mathcal{P}_3}$	355.8	350.0	353.6	344.8	374.5	352.4
	$n_{\mathcal{P}_3}$	804931.0	872201.0	802127.0	889956.0	827675.0	723702.0
tr12-30	t (sec)	1855.7	1720.2	1823.2	1679.3	1843.3	1853.5
	$t_{\mathcal{P}_3}$	1575.7	1441.2	1543.2	1398.3	1563.3	1572.5
	$n_{\mathcal{P}_3}$	1319065.0	1281105.0	1285089.0	1288959.0	1400769.0	1240823.0
unitcal_7	t (sec)	1266.3	996.9	1302.9	1209.8	1410.7	1095.4
	$t_{\mathcal{P}_3}$	826.3	556.9	860.9	768.8	969.7	653.4
	$n_{\mathcal{P}_3}$	20261.0	12394.0	21311.0	21804.0	16844.0	12982.0
vpm2	t (sec)	1.0	1.4	1.1	1.1	0.9	1.6
	$t_{\mathcal{P}_3}$	0.4	0.7	0.4	0.4	0.4	0.7
	$n_{\mathcal{P}_3}$	293.0	201.0	293.0	249.0	151.0	213.0

continued on next page

Table H.7: Instancewise results for the Proof phase experiment, see Section 5.1.3.

		weights	combined	default	dfsHeuroff	error based	sepa
zib54-UUE	t (sec)	3867.3	2675.1	3842.7	3695.0	3881.4	2752.3
	$t_{\mathcal{P}_3}$	3790.2	2598.3	3765.8	3618.0	3803.5	2675.1
	$n_{\mathcal{P}_3}$	545767.0	317386.0	537200.0	564634.0	539539.0	256146.0

Table H.8: Instancewise results for the phase transition experiment in Section 5.2. Shown are the time until an optimal solution is found, t_2^*, as well as the primal-optimal and primal-dual gaps at termination, $\gamma(T)$ and $\bar{\gamma}(T)$, for each instance. It follows for each of the four criteria the time until the criterion was reached for the first time during the solving process, as well as the primal $\gamma(t_2^{\text{crit}})$ and dual gap $\bar{\gamma}(t_2^{\text{crit}})$ at that time. Entries are left blank if the corresponding data were not observed for the corresponding instances.

		General			`estim`			`rank-1`			`log-n`			`log-it`	
	t_2^*	$\gamma(T)$	$\bar{\gamma}(T)$	t_2^{estim}	$\gamma(t_2^{\text{estim}})$	$\bar{\gamma}(t_2^{\text{estim}})$	$t_2^{\text{rank-1}}$	$\gamma(t_2^{\text{rank-1}})$	$\bar{\gamma}(t_2^{\text{rank-1}})$	$t_2^{\text{log-n}}$	$\gamma(t_2^{\text{log-n}})$	$\bar{\gamma}(t_2^{\text{log-n}})$	$t_2^{\text{log-it}}$	$\gamma(t_2^{\text{log-it}})$	$\bar{\gamma}(t_2^{\text{log-it}})$
10teams	24.5	0	0	24.9	0	0.3247	23.2	1.282	1.603						
30n20b8	83.5	0	0												
a1c1s1		1.191	15.74	265.2	2.75	24.7	199.6	4.565	26.27	34.3	28.9	48.07	94.5	8.207	31.2
acc-tight5	1423.0	0	0	1423.0	0	0	1423.0	0	0						
aflow30a	10.6	0	0	10.3	0.1724	5.095	10.4	0.1724	5.095	12.9	0	3.974	11.8	0	3.974
aflow40b	125.6	0	0	238.6	0	3.209	62.6	1.849	6.514				62.0	1.849	6.514
air04	68.2	0	0	53.4	4.057	4.738	61.4	0.4027	1.109						
air05	35.3	0	0	31.4	0.2798	1.402	31.4	0.2798	1.402	31.3	0.2798	1.402			
app1-2	619.1	0	0	1110.4	0	25.45	919.0	0	25.45						
arki001	2841.6	4.498e-10	0.004319	2230.7	6.055e-07	0.00525	25.9	0.04388	0.05566	26.1	0.04388	0.05566	192.7	0.003323	0.01103
atlanta-ip		1.099	5.658	640.4	9.999	18.13	1110.0	7.216	14.81	1678.0	6.249	13.73	4929.7	1.099	8.685
bab5		0.1937	0.9073							1182.1	0.2778	1.021	3290.7	0.1996	0.9246
beasleyC3		0.9198	13.12				85.9	2.96	17.35	1064.4	1.438	14.35	29.9	7.257	21.3
bell3a	0.1	0	0	5.7	1.821e-06	0.2715	0.2	1.821e-06	0.528						
bell5	0.1	0	0	0.4	5.482e-06	0.01922	0.1	1.308	1.426	0.1	1.308	1.426	0.1	8.374e-05	0.0255
biella1	586.6	0	0	518.5	0.1011	0.2367	601.2	7.178e-06	0.123	224.2	3.086	3.233	224.1	3.086	3.233
bienst2	86.4	0	0	68.6	1.176	30.43	10.2	3.97	35.97	115.8	0	25.82	129.9	0	25.82
binkar10_1	5.4	0	0	6.5	3.56e-07	0.4553	7.3	3.56e-07	0.4553	6.8	3.56e-07	0.4553	4.1	2.922	3.401
blend2	0.7	0	0	0.9	0	6.092	0.7	1.798	10.04						
bley_xl1	429.5	0	0												
bnatt350	1471.6	0	0	1471.6	0	0	1471.6	0	0						
cap6000	2.4	0	0	2.8	0	0.001673	1.6	0.003019	0.006078	1.8	0.002815	0.004936	1.7	0.003019	0.006078
core2536-691	322.9	0	0	241.2	0.1449	0.1955	241.2	0.1449	0.1955	270.8	0.1449	0.1955	112.7	1.712	1.783
cov1075	411.6	0	6.864	6932.8	0	7.046	297.5	0	10.64	27.9	0	13.38	13.3	4.762	18.21
csched010	689.9	7.384e-13	4.538	119.0	2.625	13.37	103.3	2.625	13.5	205.5	2.625	12.87	243.3	2.625	12.4
danoint	246.4	0	0	41.1	4.569e-06	4.382	32.4	1.99	6.319	50.0	4.569e-06	4.382	56.1	4.569e-06	4.382
dcmulti	1.7	0	0	1.5	1.117	1.482	1.7	0.09545	0.2871						
dfn-gwin-UUM	25.8	0	0	62.2	0	4.954	20.7	2.594	8.91	7.2	3.342	11.67	11.2	2.741	10.44
ds		74.09	83.55										492.3	91.98	95.06
dsbmip	1.4	0	0												
eil33-2	40.3	0	0	15.5	8.645	20.07	15.5	8.645	20.07						
eilB101	428.4	0	0	404.3	0.1687	4.325	66.5	3.058	10.22						
enigma	0.4	0	0	0.4	0	0	0.4	0	0						

continued on next page

		General			estim			rank-1			log-n			log-it	
	t_2^*	$\gamma(T)$	$\bar{\gamma}(T)$	t_2^{estim}	$\gamma(t_2^{\text{estim}})$	$\bar{\gamma}(t_2^{\text{estim}})$	$t_2^{\text{rank-1}}$	$\gamma(t_2^{\text{rank-1}})$	$\bar{\gamma}(t_2^{\text{rank-1}})$	$t_2^{\text{log-n}}$	$\gamma(t_2^{\text{log-n}})$	$\bar{\gamma}(t_2^{\text{log-n}})$	$t_2^{\text{log-it}}$	$\gamma(t_2^{\text{log-it}})$	$\bar{\gamma}(t_2^{\text{log-it}})$
enlight13	0.0	0	0	8.4	0	71.48	3.8	0	93.13						
fast0507	493.5	0	0	100.0	1.136	2.085	451.9	0.5714	1.475				411.5	0.5714	1.475
fiber	1.1	0	0												
fixnet6	1.5	0	0							3.1	0	0.7478			
flugpl	0.0	0	0	0.1	0	2.215	0.0	0	2.215						
gesa2	1.0	0	0												
gesa2-o	1.3	0	0										1.4	1.407e-05	0.008417
gesa3	1.6	0	0							1.8	9.46e-06	0.05691	1.3	0.001384	0.06141
gesa3_o	1.8	0	0										1.5	0.001384	0.05745
glass4		22.58	35.48							34.7	33.33	55.56	18.9	33.33	55.56
gmu-35-40		0.008501	0.01611				67.5	0.05661	0.06423	16.8	0.1242	0.1318	1.6	0.2064	0.2142
harp2	3172.0	0	0	2984.5	2.706e-06	0.005155	6.1	0.2369	0.5045	114.8	2.706e-06	0.08706	9.4	0.2369	0.5045
iis-100-0-cov	49.0	0	0	29.9	17.14	51.43	39.6	3.333	36.05	110.1	0	30.78	127.0	0	30.78
iis-bupa-cov	148.1	0	0	73.9	5.263	29.93	112.0	2.703	22.93	507.3	0	19.25	225.7	0	19.25
iis-pima-cov	1233.5	0	0	77.8	2.941	21.35	96.7	2.941	21.35				341.3	2.941	21.35
khb05250	0.6	0	0												
l152lav	2.4	0	0												
lectsched-4-obj	396.3	0	0	387.0	20	20				73.7	63.64	63.64	163.4	60	60
lseu	0.5	0	0	0.5	0	2.288	0.3	0.7092	6.135				0.4	0.7092	6.135
m100n500k4r1		4	4				31.6	4	4	51.6	4	4	7.2	8	8
macrophage		0.2667	18.45	17.7	19.91	46.85				54.0	2.857	32.71	67.2	2.857	32.71
map18	274.8	0	0	275.1	0	4.173	271.2	0.2361	4.399	365.9	0	4.173			
map20	55.4	0	0	193.2	0	6.24	237.1	0	6.24						
markshare1		80	100	1163.5	87.5	100	6.0	96.97	100	2.6	97.78	100	0.3	98.89	100
markshare2		92.31	100				3.8	97.37	100	22.2	97.37	100	0.5	99.01	100
mas74	18.7	0	0	270.7	3.644e-05	3.697	18.7	3.644e-05	6.938				1.1	8.195	18.19
mas76	0.5	0	0	37.1	1.025e-05	1.099	1.2	1.025e-05	2.624				0.7	1.025e-05	2.624
mcsched	175.6	0	0	16.2	1.776	10.17	20.9	0.9058	9.372	16.0	1.836	10.23	22.5	0.9058	9.372
mik-250-1-100-1	0.5	0	0	16.9	0	2.988	2.3	0	4.372				0.9	0	4.372
mine-166-5	30.1	0	0	27.0	0.2634	1.427	30.6	1.39e-06	0.2482	29.8	0.05754	0.4998			
mine-90-10	176.8	0	0	37.9	0.8418	3.149	81.9	0.2212	0.7084	32.7	6.408	9.762			
misc03	0.5	0	0	1.9	0	34.42	1.7	0	34.42						
misc06	0.6	0	0												
misc07	1.2	0	0	12.7	0	21.8	3.5	0	49.04	2.1	0	49.04	2.1	0	49.04
mkc		0.03441	0.1944				5603.9	0.2118	0.3715	1261.4	0.3891	0.5486	40.3	3.319	4.386
mod008	1.1	0	0												
mod010	0.4	0	0												
mod011	145.2	0	0	150.6	9.139e-06	1.218	96.0	0.2498	3.159				125.4	9.139e-06	1.817
modglob	1.1	0	0	1.4	9.643e-06	0.1244	0.9	0.05976	0.2241				0.8	0.2227	0.455
momentum1		5.594	11.12				2322.9	5.613	16.62	610.6	37.06	44.4	631.4	37.06	44.4
momentum2		0.001531	0.0336	4217.4	0.8312	13.48	4219.0	0.8312	13.48	1180.6	10.86	22.48	1449.6	10.86	22.48
msc98-ip		8.384	9.015												
mspp16	496.3	0	0	2607.0	0	6.061	2607.0	0	6.061						

continued on next page

		General		estim			rank-1			log-n			log-it		
	t_2^*	$\gamma(T)$	$\bar{\gamma}(T)$	t_2^{estim}	$\gamma(t_2^{\text{estim}})$	$\bar{\gamma}(t_2^{\text{estim}})$	$t_2^{\text{rank-1}}$	$\gamma(t_2^{\text{rank-1}})$	$\bar{\gamma}(t_2^{\text{rank-1}})$	$t_2^{\text{log-n}}$	$\gamma(t_2^{\text{log-n}})$	$\bar{\gamma}(t_2^{\text{log-n}})$	$t_2^{\text{log-it}}$	$\gamma(t_2^{\text{log-it}})$	$\bar{\gamma}(t_2^{\text{log-it}})$
mzzv11	198.4	0	0	134.9	3.591	4.537	228.4	0	0.8883	136.4	3.591	4.537			
mzzv42z	330.0	0	0	137.5	2.824	3.874	219.1	1.947	2.967	142.8	2.775	3.826			
n3div36		0.1527	5.589							391.8	3.965	10.99	335.6	3.965	10.99
n3seq24		15.26	15.58												
n4-3	62.1	0	0	51.4	1.759	10.65	52.3	1.759	10.65	32.8	25.28	32.96	156.2	0	8.227
neos-1109824	9.8	0	0	125.4	0	4.568	30.3	0	10.63	17.5	0	10.63	25.2	0	10.63
neos-1337307	316.3	0	0.03699	103.3	0.0687	0.4566	1675.5	0	0.1765	143.1	0.02076	0.4047	515.2	0	0.3274
neos-1396125	437.5	0	0	231.2	0.0002554	35.35	71.4	21.05	50.04	145.4	0.0002554	35.35	128.6	0.0002554	35.35
neos-1601936		25	25	1800.9	50	50	1406.7	50	50	1674.2	50	50	4155.4	25	25
neos-476283	252.7	0	0	250.2	0.000392	0.01225	242.3	0.03584	0.04866						
neos-686190	81.9	0	0	65.1	3.026	13.84	61.5	3.026	13.84						
neos-849702	176.2	0	0	176.2	0	0	176.2	0	0						
neos-916792	217.6	0	0	139.3	0.522	10.63	47.8	4.162	19.1	411.1	6.275e-06	2.286	24.6	6.353	21.19
neos-934278		5.455	5.636										238.6	99.55	99.56
neos13	1452.4	0	0	1471.3	3.142e-06	3.298	1333.5	19.25	38.9	605.3	20.11	39.55	1095.5	19.25	38.9
neos18	13.4	0	0	26.3	0	12.5	19.7	0	12.5	28.9	0	12.5	19.1	0	12.5
net12	1045.0	0	0	1045.0	16.08	56.71	1557.0	0	48.41	2372.5	0	48.41			
netdiversion		3.586	6.729	3561.3	3.586	7.171									
newdano	1164.7	0	0	38.8	5.028	50.71	45.4	4.831	50.61	9.3	14.72	56.06	173.8	1.99	44.78
noswot	79.8	0	0	21.5	0	4.651	1.5	0	4.651	1.6	0	4.651	0.6	12.2	16.28
ns1208400	1806.5	0	0	1806.5	0	100	1806.5	0	100						
ns1688347	674.4	0	0	648.8	10	23.2	646.8	10	23.2						
ns1758913		83.72	83.72												
ns1830653	214.8	0	0	92.7	55.77	76.68	44.7	55.77	76.68						
nsrand-ipx		1.235	1.89							85.1	4.762	6.086	88.3	4.762	6.086
nw04	23.1	0	0												
opm2-z7-s2	786.8	0	0	157.6	1.673	19.74	270.0	1.469	15.04	151.0	23.27	38.47			
p0201	1.4	0	0	1.8	0	2.342	1.8	0	2.342						
p0282	0.3	0	0										0.3	4.088	4.437
p2756	1.9	0	0	1.9	0	0.002465	1.9	0	0.002465						
pg5_34	1173.7	0	0	1191.7	2.404e-05	0.06885	60.1	0.1014	0.2787	55.5	0.1014	0.2787	25.3	0.1014	0.2787
pigeon-10	4736.4	0	10							2329.4	0	10	2332.0	0	10
pk1	8.9	0	0	21.2	0	61.96	9.7	0	77.72				0.8	42.11	100
pp08a	1.1	0	0	1.1	0.1359	2.079	1.1	0.1359	2.079						
pp08aCUTS	1.1	0	0	1.1	0.1359	1.62	1.0	0.1359	1.62						
protfold		25.81	37.67												
pw-myciel4	52.2	0	0	52.2	0	60	57.4	0	60	37.7	9.091	63.64	605.9	0	50
qiu	43.5	0	0	19.2	100.5	100.5	20.6	89.14	97.9	39.5	2.785e-05	79.36	14.7	113.2	113.2
qnet1	8.1	0	0												
qnet1_o	6.8	0	0												
rail507	153.2	0	0	98.6	0.5714	1.525	156.1	0	0.9056	217.2	0	0.9056	126.2	0.5714	1.525
ran16x16	32.0	0	0	126.0	0	2.77	10.2	2.325	9.317				4.8	2.325	9.317
rd-rplusc-21		0.3258	99.94				660.2	7.799	99.94	677.7	7.799	99.94	1251.1	0.6282	99.94

continued on next page

		General			estim			rank-1			log-n			log-it	
	t_2^*	$\gamma(T)$	$\bar{\gamma}(T)$	t_2^{estim}	$\gamma(t_2^{\text{estim}})$	$\bar{\gamma}(t_2^{\text{estim}})$	$t_2^{\text{rank-1}}$	$\gamma(t_2^{\text{rank-1}})$	$\bar{\gamma}(t_2^{\text{rank-1}})$	$t_2^{\text{log-n}}$	$\gamma(t_2^{\text{log-n}})$	$\bar{\gamma}(t_2^{\text{log-n}})$	$t_2^{\text{log-it}}$	$\gamma(t_2^{\text{log-it}})$	$\bar{\gamma}(t_2^{\text{log-it}})$
reblock67	57.4	0	0	45.5	1.728	3.698	210.2	4.509e-06	0.6325	50.0	0.9079	2.593	72.4	4.509e-06	1.449
rentacar	2.7	0	0												
rmatr100-p10	66.6	0	0	66.7	0	9.346	66.7	0	9.346						
rmatr100-p5	183.3	0	0	231.1	0	11.97	186.0	0	11.97				296.8	0	11.97
rmine6	4322.3	0	0	32.3	0.5909	1.603	2864.8	0.001868	0.07484	293.5	0.001868	0.1526	62.5	0.09798	0.5149
rocII-4-11	465.0	0	0	438.7	14.94	26.68	53.4	45.4	63.01						
rococoC10-001000	649.2	0	0	34.0	4.084	15.03	106.2	0.4776	7.145	176.0	0.4776	5.927	21.2	13.08	23.75
roll3000		0.06977	0.5773	76.9	0.5017	4.504	71.8	0.5017	4.504	112.8	0.3479	3.225	141.8	0.3479	3.225
rout	29.5	0	0	29.5	0.151	3.751	14.6	0.7507	6.5	4.3	10.25	17.88	6.7	7.452	15.19
satellites1-25	487.9	0	0	268.8	60	89.74	292.7	60	89.74						
set1ch	1.1	0	0												
seymour	6801.6	0	1.888	46.2	3.204	6.629				50.0	2.759	6.2	96.1	2.083	5.423
sp97ar		6.951	8.072							1513.2	7.837	8.947	124.8	18.22	19.21
sp98ic		0.7265	1.081							473.4	4.095	4.657	97.3	4.214	4.951
sp98ir	96.4	0	0	52.9	0.2137	0.9942	40.4	0.4436	1.339	13.7	6.181	7.302	94.8	0.1288	0.736
stein27	0.0	0	0	0.4	0	27.78	0.4	0	27.78						
stein45	1.2	0	0	0.8	3.226	29.03	0.8	3.226	29.03				2.9	0	23.33
swath		1.108	13.94				138.1	6.643	23.84	50.4	6.643	23.84	110.4	6.643	23.84
tanglegram1	1064.0	0	0												
tanglegram2	10.1	0	0												
timtab1	38.9	0	0	8.3	13.23	36.53	4.6	22.98	46.32	70.7	0	19.36	81.7	0	18.88
timtab2		3.533	32	33.6	13	52.63	14.0	35.75	65.49	56.3	12.84	52.02	229.7	12.09	48.7
tr12-30	278.1	0	0	76.2	0.003063	0.1566	10.5	0.003063	0.2218	272.9	0.003063	0.1162	8.2	0.5506	0.8418
triptim1	2801.0	0	0												
unitcal_7	807.7	0	0	192.5	0.9471	1.387	1300.3	8.943e-06	0.04696	1064.6	8.943e-06	0.0743			
vpm2	0.8	0	0	1.2	0	5.361	1.1	0	5.361				1.1	0	5.361
vpphard		44.44	100							5513.9	44.44	100			
zib54-UUE	77.6	0	0	49.7	0.1425	24.08	47.8	3.31	27.57	23.5	23.14	44.33	74.7	0.1425	24.08

Table H.9: Instancewise results of the experiment in Section 5.3. Each column except for `default` denotes a run with a particular phase transition criterion. For each instance, we present the values of the dual integral $\Gamma^*(T)$, the number of branch-and-bound nodes n, the achieved primal bound $c^t\hat{y}$ at termination, the value $\Gamma(T)$ of the primal integral, and the overall solving time t (sec).

Settings Problem		default	estim	log-n	log-it	oracle	rank-1
10teams	$\Gamma^*(T)$	58.4	61.6	60.7	60.8	60.9	60.8
	n	1272.0	1612.0	1612.0	1612.0	1552.0	1612.0
	$c^t\hat{y}$	924.0	924.0	924.0	924.0	924.0	924.0
	$\Gamma(T)$	1459.7	648.5	549.0	559.1	599.0	584.4
	t (sec)	25.6	35.4	33.0	33.1	33.3	33.3
30n20b8	$\Gamma^*(T)$	9555.0	8011.9	8052.4	7998.3	6959.1	8029.3
	n	14.0	6.0	6.0	6.0	4.0	6.0
	$c^t\hat{y}$	302.0	302.0	302.0	302.0	302.0	302.0
	$\Gamma(T)$	8418.0	8330.0	8380.0	8346.0	8317.0	8410.0
	t (sec)	183.5	151.7	152.5	151.5	135.6	152.0
a1c1s1	$\Gamma^*(T)$	124305.2	118916.9	156221.5	166693.8	118644.8	162765.1
	n	718786.0	626920.0	2154765.0	1954964.0	640772.0	1815029.0
	$c^t\hat{y}$	11642.1	11701.0	11832.0	12289.0	11701.0	11754.2
	$\Gamma(T)$	14318.3	14326.3	21710.2	47142.5	14270.7	25160.9
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
acc-tight5	$\Gamma^*(T)$	210.0	210.0	200.0	200.0	200.0	220.0
	n	16931.0	608.0	608.0	608.0	608.0	608.0
	$c^t\hat{y}$	0.0	0.0	0.0	0.0	0.0	0.0
	$\Gamma(T)$	143661.0	6170.0	6259.0	6160.0	6160.0	6220.0
	t (sec)	1436.6	61.7	62.5	61.6	61.6	62.2
aflow30a	$\Gamma^*(T)$	89.8	93.1	78.0	87.1	87.7	89.1
	n	2118.0	2506.0	2506.0	1836.0	1852.0	1976.0
	$c^t\hat{y}$	1158.0	1158.0	1158.0	1158.0	1158.0	1158.0
	$\Gamma(T)$	385.8	398.1	363.6	388.3	388.3	390.8
	t (sec)	13.9	14.5	13.6	13.1	13.3	13.5
aflow40b	$\Gamma^*(T)$	2589.6	3114.8	3091.2	15773.6	2798.2	6318.6
	n	151103.0	112411.0	112411.0	846199.0	69617.0	323752.0
	$c^t\hat{y}$	1168.0	1168.0	1168.0	1168.0	1168.0	1168.0
	$\Gamma(T)$	1850.7	2421.6	2426.1	8517.7	2420.7	4094.9
	t (sec)	912.9	941.0	936.2	3198.2	681.0	1328.9
air03	$\Gamma^*(T)$	70.2	80.2	80.1	90.2	70.2	80.1
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	340160.0	340160.0	340160.0	340160.0	340160.0	340160.0
	$\Gamma(T)$	81.5	88.4	84.7	98.4	82.8	84.4
	t (sec)	1.0	1.1	1.1	1.2	1.0	1.1
air04	$\Gamma^*(T)$	453.3	441.4	451.3	461.1	442.4	452.2
	n	213.0	146.0	146.0	146.0	156.0	156.0
	$c^t\hat{y}$	56137.0	56137.0	56137.0	56137.0	56137.0	56137.0
	$\Gamma(T)$	756.8	740.2	747.9	751.7	742.1	747.4
	t (sec)	70.4	64.1	64.2	64.0	65.6	65.5
air05	$\Gamma^*(T)$	166.5	189.7	214.5	200.2	209.6	189.9
	n	181.0	309.0	547.0	309.0	299.0	365.0
	$c^t\hat{y}$	26374.0	26374.0	26374.0	26374.0	26374.0	26374.0
	$\Gamma(T)$	415.9	529.4	560.6	547.8	552.5	530.7
	t (sec)	37.5	39.3	44.1	39.8	39.3	39.4
app1-2	$\Gamma^*(T)$	52653.8	63944.0	63882.4	63880.1	52923.0	52774.2
	n	427.0	429.0	429.0	429.0	306.0	246.0
	$c^t\hat{y}$	-41.0	-41.0	-41.0	-41.0	-41.0	-41.0
	$\Gamma(T)$	28924.9	28904.7	28886.6	28867.0	28710.0	29076.8
	t (sec)	1118.2	1270.6	1270.7	1270.3	976.2	964.7
arki001	$\Gamma^*(T)$	56.9	93.7	147.5	137.2	99.8	103.4
	n	1166439.0	915413.0	880296.0	1246593.0	960617.0	997478.0
	$c^t\hat{y}$	7580813.0	7580813.0	7581551.4	7581558.2	7580813.0	7580813.0
	$\Gamma(T)$	474.5	595.1	684.1	676.5	606.4	606.2
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
ash608gpia-3col	$\Gamma^*(T)$	2330.0	3140.0	3240.0	3160.0	3150.0	3160.0
	n	10.0	27.0	27.0	27.0	27.0	27.0
	$\Gamma(T)$	2330.0	3140.0	3240.0	3160.0	3150.0	3160.0
	t (sec)	23.3	31.4	32.4	31.6	31.5	31.6
atlanta-ip	$\Gamma^*(T)$	55112.1	66498.4	69548.6	69446.6	66528.0	70865.4
	n	14375.0	9861.0	171533.0	142444.0	9852.0	260171.0

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings / Problem		default	estim	log-n	log-it	oracle	rank-1
	$c^t\hat{y}$	91.0	95.0	94.0	95.0	95.0	97.0
	$\Gamma(T)$	77572.2	60360.0	58264.2	60317.7	60360.0	72558.8
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
bab5	$\Gamma^*(T)$	7585.0	13818.2	12153.5	13752.8	13765.1	13844.4
	n	25819.0	17781.0	68392.0	52073.0	17837.0	17728.0
	$c^t\hat{y}$	-106205.7	-106207.2	-106207.2	-106207.2	-106207.2	-106207.2
	$\Gamma(T)$	23143.2	20850.2	20714.3	20649.5	20707.9	20733.8
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
beasleyC3	$\Gamma^*(T)$	91701.2	95215.9	98095.7	107836.7	95218.1	107829.5
	n	796130.0	1106432.0	986437.0	2078457.0	1106852.0	2065757.0
	$c^t\hat{y}$	761.0	759.0	764.0	832.0	759.0	832.0
	$\Gamma(T)$	9542.5	6425.8	10178.1	67599.1	6431.9	67604.0
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
bell3a	$\Gamma^*(T)$	2.3	2.1	2.0	2.2	2.2	2.3
	n	22487.0	23064.0	22573.0	23064.0	22579.0	23083.0
	$c^t\hat{y}$	878430.3	878430.3	878430.3	878430.3	878430.3	878430.3
	$\Gamma(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	t (sec)	6.1	5.7	4.3	6.0	4.2	4.5
bell5	$\Gamma^*(T)$	10.0	0.0	10.0	0.0	10.0	10.0
	n	1140.0	1226.0	1214.0	1154.0	1218.0	1224.0
	$c^t\hat{y}$	8966406.5	8966406.5	8966406.5	8966406.5	8966406.5	8966406.5
	$\Gamma(T)$	6.2	0.2	7.1	0.1	6.2	6.2
	t (sec)	0.8	0.4	0.6	0.4	0.6	0.6
biella1	$\Gamma^*(T)$	649.4	625.1	933.7	745.0	614.8	727.5
	n	2133.0	2538.0	29451.0	17214.0	2436.0	14468.0
	$c^t\hat{y}$	3065005.8	3065005.8	3065005.8	3065005.8	3065005.8	3065005.8
	$\Gamma(T)$	6224.9	3602.3	4882.7	4028.8	3609.6	3983.0
	t (sec)	781.4	578.7	2588.9	1593.8	575.2	1395.4
bienst2	$\Gamma^*(T)$	6024.9	6081.0	7982.4	7255.8	8201.8	9499.7
	n	93988.0	93988.0	92022.0	92786.0	92643.0	106272.0
	$c^t\hat{y}$	54.6	54.6	54.6	54.6	54.6	54.6
	$\Gamma(T)$	250.7	256.4	249.6	241.4	246.0	203.6
	t (sec)	297.4	297.8	301.4	301.7	301.6	291.8
binkar10_1	$\Gamma^*(T)$	52.8	59.4	82.2	80.5	73.4	82.9
	n	138787.0	120843.0	117093.0	114515.0	119374.0	120533.0
	$c^t\hat{y}$	6742.2	6742.2	6742.2	6742.2	6742.2	6742.2
	$\Gamma(T)$	66.1	61.3	57.1	61.4	64.6	69.1
	t (sec)	177.2	158.3	135.0	131.9	147.8	137.2
blend2	$\Gamma^*(T)$	16.5	20.9	7.6	9.3	20.9	19.0
	n	412.0	932.0	932.0	932.0	933.0	634.0
	$c^t\hat{y}$	7.6	7.6	7.6	7.6	7.6	7.6
	$\Gamma(T)$	40.7	41.8	21.3	21.6	41.8	41.4
	t (sec)	0.9	1.4	0.9	1.1	1.4	1.2
bley_xl1	$\Gamma^*(T)$	33176.4	33376.4	33268.2	33076.4	33076.4	33268.6
	n	20.0	20.0	20.0	20.0	20.0	20.0
	$c^t\hat{y}$	190.0	190.0	190.0	190.0	190.0	190.0
	$\Gamma(T)$	37128.9	37425.6	37174.8	37026.5	36974.8	37198.1
	t (sec)	430.6	434.6	430.9	429.6	429.0	431.4
bnatt350	$\Gamma^*(T)$	80.0	70.0	80.0	80.0	70.0	80.0
	n	21343.0	14092.0	14092.0	14092.0	14092.0	14092.0
	$c^t\hat{y}$	0.0	0.0	0.0	0.0	0.0	0.0
	$\Gamma(T)$	147700.0	81900.0	82100.0	81976.0	81894.0	82867.0
	t (sec)	1477.1	819.2	821.1	819.7	818.9	828.6
cap6000	$\Gamma^*(T)$	40.0	50.0	50.0	40.0	40.0	50.0
	n	3788.0	4064.0	5135.0	4999.0	4080.0	4561.0
	$c^t\hat{y}$	-2451377.0	-2451377.0	-2451377.0	-2451377.0	-2451377.0	-2451377.0
	$\Gamma(T)$	36.6	46.2	46.2	36.4	36.4	46.2
	t (sec)	2.7	3.1	2.7	2.7	2.7	2.8
core2536-691	$\Gamma^*(T)$	980.3	990.5	980.2	963.5	1030.7	980.5
	n	218.0	218.0	475.0	191.0	218.0	481.0
	$c^t\hat{y}$	689.0	689.0	689.0	689.0	689.0	689.0
	$\Gamma(T)$	1278.3	1309.6	1297.6	1251.9	1355.2	1299.6
	t (sec)	318.1	321.2	316.4	194.5	324.1	319.6
cov1075	$\Gamma^*(T)$	62170.0	62169.9	86773.6	89515.5	90051.2	76188.9

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings Problem		default	estim	log-n	log-it	oracle	rank-1
	n	1557428.0	1559145.0	1690517.0	1506333.0	1635309.0	1854530.0
	$c^t\hat{y}$	20.0	20.0	20.0	20.0	20.0	20.0
	$\Gamma(T)$	250.8	243.1	255.9	244.1	242.6	239.0
	t (sec)	7200.0	7200.0	6869.1	6588.8	6923.5	7200.0
csched010	$\Gamma^*(T)$	47458.5	76490.1	78698.4	83291.1	63870.3	82644.7
	n	931270.0	1049236.0	1102338.0	1196386.0	997781.0	1128300.0
	$c^t\hat{y}$	408.0	409.0	410.0	410.0	408.0	410.0
	$\Gamma(T)$	8987.5	10338.6	14514.0	21720.7	9311.0	16636.5
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
danoint	$\Gamma^*(T)$	16556.3	15726.8	16828.8	17242.8	16750.4	19819.0
	n	1050040.0	966643.0	891818.0	927764.0	881640.0	1089980.0
	$c^t\hat{y}$	65.7	65.7	65.7	65.7	65.7	65.7
	$\Gamma(T)$	1010.9	291.5	313.1	292.0	294.9	2374.2
	t (sec)	5078.2	4785.0	3879.3	4020.2	3836.3	4451.6
dcmulti	$\Gamma^*(T)$	1.0	0.9	0.9	10.7	0.9	0.9
	n	322.0	316.0	316.0	316.0	306.0	261.0
	$c^t\hat{y}$	188182.0	188182.0	188182.0	188182.0	188182.0	188182.0
	$\Gamma(T)$	119.5	61.0	61.0	67.8	61.0	61.0
	t (sec)	1.6	1.2	1.2	1.2	1.2	1.2
dfn-gwin-UUM	$\Gamma^*(T)$	658.8	669.3	1122.9	3651.8	780.1	733.7
	n	66936.0	61708.0	87137.0	396801.0	63462.0	60074.0
	$c^t\hat{y}$	38752.0	38752.0	38752.0	38752.0	38752.0	38752.0
	$\Gamma(T)$	436.6	437.8	439.9	1095.5	430.8	434.5
	t (sec)	139.6	133.4	133.9	471.3	113.9	115.1
disctom	$\Gamma^*(T)$	190.0	200.0	190.0	190.0	200.0	200.0
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	-5000.0	-5000.0	-5000.0	-5000.0	-5000.0	-5000.0
	$\Gamma(T)$	366.0	390.0	370.0	370.0	398.0	386.0
	t (sec)	3.6	3.9	3.7	3.7	3.9	3.8
ds	$\Gamma^*(T)$	273079.9	274607.1	274635.9	274617.6	274604.6	274595.1
	n	523.0	542.0	547.0	547.0	546.0	546.0
	$c^t\hat{y}$	361.0	316.6	316.6	316.6	316.6	316.6
	$\Gamma(T)$	571215.7	544517.3	544317.2	544306.8	544297.5	544375.4
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
dsbmip	$\Gamma^*(T)$	30.0	40.0	40.0	60.0	50.0	60.0
	n	15.0	11.0	11.0	11.0	11.0	11.0
	$c^t\hat{y}$	-305.2	-305.2	-305.2	-305.2	-305.2	-305.2
	$\Gamma(T)$	75.0	75.9	74.5	89.4	91.0	89.0
	t (sec)	1.2	1.1	1.2	1.3	1.4	1.3
egout	$\Gamma^*(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	568.1	568.1	568.1	568.1	568.1	568.1
	$\Gamma(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	t (sec)	0.0	0.0	0.0	0.0	0.0	0.0
eil33-2	$\Gamma^*(T)$	700.9	728.0	716.8	733.0	969.4	1149.3
	n	735.0	851.0	851.0	851.0	1235.0	1339.0
	$c^t\hat{y}$	934.0	934.0	934.0	934.0	934.0	934.0
	$\Gamma(T)$	518.0	501.3	495.6	511.5	519.8	595.2
	t (sec)	52.8	57.3	57.1	57.0	78.9	96.7
eilB101	$\Gamma^*(T)$	2920.5	2997.4	2957.5	2942.4	2971.5	2276.5
	n	8028.0	8283.0	8283.0	8283.0	8357.0	6776.0
	$c^t\hat{y}$	1216.9	1216.9	1216.9	1216.9	1216.9	1216.9
	$\Gamma(T)$	1262.9	1781.2	1750.3	1743.5	1757.0	822.5
	t (sec)	436.2	477.7	471.6	469.5	474.1	323.3
enigma	$\Gamma^*(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	n	954.0	2759.0	2759.0	2759.0	2759.0	2759.0
	$c^t\hat{y}$	0.0	0.0	0.0	0.0	0.0	0.0
	$\Gamma(T)$	50.0	68.0	60.0	75.0	60.0	78.0
	t (sec)	0.5	0.6	0.6	0.7	0.6	0.7
enlight13	$\Gamma^*(T)$	769.1	1386.2	1345.1	1340.9	10777.8	970.0
	n	13479.0	30211.0	30211.0	30211.0	270890.0	23151.0
	$c^t\hat{y}$	71.0	71.0	71.0	71.0	71.0	71.0
	$\Gamma(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	t (sec)	8.6	16.8	16.3	16.3	119.5	10.7

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings Problem		default	estim	log-n	log-it	oracle	rank-1
enlight14	$\Gamma^*(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$\Gamma(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	t (sec)	0.0	0.0	0.0	0.0	0.0	0.0
ex9	$\Gamma^*(T)$	3820.0	3730.0	3750.0	3750.0	3950.0	3750.0
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	81.0	81.0	81.0	81.0	81.0	81.0
	$\Gamma(T)$	3820.0	3728.0	3745.0	3748.0	3950.0	3746.0
	t (sec)	38.2	37.2	37.4	37.4	39.5	37.4
fast0507	$\Gamma^*(T)$	1250.4	1258.6	1299.6	1265.7	1278.5	1287.0
	n	1376.0	1376.0	1376.0	1779.0	1348.0	1862.0
	$c^t\hat{y}$	174.0	174.0	174.0	174.0	174.0	174.0
	$\Gamma(T)$	907.7	932.2	948.0	915.3	947.8	937.1
	t (sec)	576.8	574.8	576.3	581.4	574.8	615.3
fiber	$\Gamma^*(T)$	10.6	7.5	6.9	8.3	22.1	18.5
	n	8.0	8.0	8.0	8.0	5.0	8.0
	$c^t\hat{y}$	405935.2	405935.2	405935.2	405935.2	405935.2	405935.2
	$\Gamma(T)$	40.8	28.6	25.4	38.9	50.6	50.1
	t (sec)	1.6	1.5	1.4	1.6	3.4	1.7
fixnet6	$\Gamma^*(T)$	15.8	26.3	16.1	15.4	29.6	25.9
	n	9.0	9.0	9.0	9.0	8.0	9.0
	$c^t\hat{y}$	3983.0	3983.0	3983.0	3983.0	3983.0	3983.0
	$\Gamma(T)$	13.8	23.8	15.0	12.6	23.8	22.6
	t (sec)	3.0	3.3	3.1	3.2	7.5	3.3
flugpl	$\Gamma^*(T)$	0.0	0.2	0.2	0.0	0.0	0.2
	n	251.0	115.0	115.0	115.0	115.0	174.0
	$c^t\hat{y}$	1201500.0	1201500.0	1201500.0	1201500.0	1201500.0	1201500.0
	$\Gamma(T)$	0.0	0.3	0.3	0.0	0.0	0.3
	t (sec)	0.0	0.0	0.0	0.0	0.0	0.0
gen	$\Gamma^*(T)$	10.0	10.0	10.0	10.0	10.0	10.0
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	112313.4	112313.4	112313.4	112313.4	112313.4	112313.4
	$\Gamma(T)$	6.0	6.0	5.0	6.0	7.0	6.0
	t (sec)	0.1	0.1	0.1	0.1	0.1	0.1
gesa2	$\Gamma^*(T)$	10.1	10.2	10.0	10.1	10.2	10.2
	n	2.0	2.0	2.0	2.0	2.0	2.0
	$c^t\hat{y}$	25779856.4	25779856.4	25779856.4	25779856.4	25779856.4	25779856.4
	$\Gamma(T)$	69.8	96.2	61.1	78.3	89.4	80.8
	t (sec)	0.9	1.2	0.8	1.0	1.1	1.0
gesa2-o	$\Gamma^*(T)$	10.3	10.4	10.3	20.3	10.4	10.4
	n	5.0	5.0	5.0	3.0	5.0	5.0
	$c^t\hat{y}$	25779856.4	25779856.4	25779856.4	25779856.4	25779856.4	25779856.4
	$\Gamma(T)$	23.2	23.8	14.4	24.4	23.8	25.7
	t (sec)	1.2	1.3	1.2	1.4	1.2	1.6
gesa3	$\Gamma^*(T)$	10.1	10.2	10.2	10.2	10.2	10.2
	n	7.0	7.0	7.0	6.0	6.0	7.0
	$c^t\hat{y}$	27991042.6	27991042.6	27991042.6	27991042.6	27991042.6	27991042.6
	$\Gamma(T)$	13.8	15.2	16.1	15.7	15.7	26.0
	t (sec)	1.6	1.6	1.7	2.4	2.4	2.0
gesa3_o	$\Gamma^*(T)$	10.1	10.2	10.1	10.2	10.3	10.2
	n	8.0	8.0	8.0	6.0	8.0	8.0
	$c^t\hat{y}$	27991042.6	27991042.6	27991042.6	27991042.6	27991042.6	27991042.6
	$\Gamma(T)$	15.5	16.5	14.6	17.4	27.4	24.7
	t (sec)	1.6	1.8	1.6	2.0	1.9	1.7
glass4	$\Gamma^*(T)$	158051.6	210964.4	240008.1	240008.9	211095.6	240008.6
	n	16199130.0	14938613.0	20602927.0	19467769.0	14873922.0	19924545.0
	$c^t\hat{y}$	1550013650.0	1575014925.0	1500012950.0	1600015100.0	1575014925.0	1566682704.5
	$\Gamma(T)$	165720.8	178427.0	149244.3	194442.3	178445.2	175851.2
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
gmu-35-40	$\Gamma^*(T)$	64.9	71.0	64.9	74.9	71.0	74.9
	n	13065327.0	14149261.0	20030837.0	19648359.0	14168881.0	22385757.0
	$c^t\hat{y}$	-2406528.8	-2406328.9	-2405322.2	-2402821.1	-2406328.9	-2404683.6
	$\Gamma(T)$	118.5	159.7	438.2	1242.4	159.5	635.5
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
gt2	$\Gamma^*(T)$	0.0	0.2	0.2	0.2	0.2	0.2

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings / Problem		default	estim	log-n	log-it	oracle	rank-1
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	21166.0	21166.0	21166.0	21166.0	21166.0	21166.0
	$\Gamma(T)$	1.0	5.9	5.9	5.9	5.9	5.9
	t (sec)	0.0	0.1	0.1	0.1	0.1	0.1
harp2	$\Gamma^*(T)$	56.7	48.7	132.8	721.6	48.7	1592.9
	n	12630591.0	11703033.0	4171617.0	12761954.0	11722399.0	22409557.0
	$c^t\hat{y}$	-73899798.0	-73899798.0	-73899798.0	-73899798.0	-73899798.0	-73899797.0
	$\Gamma(T)$	25.4	26.3	25.1	240.8	26.5	615.4
	t (sec)	3700.6	4479.6	1357.6	3765.6	4430.6	7200.0
iis-100-0-cov	$\Gamma^*(T)$	29247.4	30216.5	35017.4	36085.1	39767.6	45359.1
	n	102734.0	105711.0	91875.0	95989.0	85533.0	89706.0
	$c^t\hat{y}$	29.0	29.0	29.0	29.0	29.0	29.0
	$\Gamma(T)$	623.8	651.6	644.5	649.4	658.5	782.8
	t (sec)	1663.9	1722.3	1367.7	1420.9	1305.2	1358.8
iis-bupa-cov	$\Gamma^*(T)$	65009.9	67794.8	86873.1	93269.3	104947.8	114341.6
	n	182534.0	179812.0	170742.0	168904.0	172416.0	182329.0
	$c^t\hat{y}$	36.0	36.0	36.0	36.0	36.0	36.0
	$\Gamma(T)$	1397.0	1155.4	1110.9	1115.1	1115.4	2337.0
	t (sec)	6142.9	6512.3	5453.0	5314.9	5434.2	5552.2
iis-pima-cov	$\Gamma^*(T)$	19588.8	19722.4	19784.8	13113.0	19735.1	10262.7
	n	20364.0	20278.0	20278.0	12761.0	20296.0	7935.0
	$c^t\hat{y}$	33.0	33.0	33.0	33.0	33.0	33.0
	$\Gamma(T)$	4259.5	4391.8	4395.6	2157.3	4382.1	1156.2
	t (sec)	1383.2	1388.3	1392.8	870.4	1388.6	610.8
khb05250	$\Gamma^*(T)$	0.1	1.1	0.1	0.1	1.1	1.1
	n	3.0	3.0	3.0	3.0	2.0	3.0
	$c^t\hat{y}$	106940226.0	106940226.0	106940226.0	106940226.0	106940226.0	106940226.0
	$\Gamma(T)$	2.5	3.2	1.9	2.8	3.2	3.2
	t (sec)	0.5	0.6	0.3	0.6	1.0	0.6
l152lav	$\Gamma^*(T)$	12.9	13.5	13.2	23.9	13.8	23.7
	n	49.0	92.0	90.0	92.0	92.0	92.0
	$c^t\hat{y}$	4722.0	4722.0	4722.0	4722.0	4722.0	4722.0
	$\Gamma(T)$	91.0	57.6	31.9	62.0	62.0	68.6
	t (sec)	2.5	3.0	2.8	3.4	3.2	3.3
lectsched-4-obj	$\Gamma^*(T)$	237.1	249.8	229.6	227.5	237.5	237.5
	n	24222.0	8296.0	10926.0	11513.0	9683.0	9683.0
	$c^t\hat{y}$	4.0	4.0	4.0	4.0	4.0	4.0
	$\Gamma(T)$	21706.8	8547.6	11878.6	11413.7	15065.8	15061.2
	t (sec)	399.0	109.6	161.5	153.9	200.4	200.1
lseu	$\Gamma^*(T)$	3.5	5.4	3.4	3.6	5.5	5.0
	n	336.0	606.0	606.0	338.0	602.0	379.0
	$c^t\hat{y}$	1120.0	1120.0	1120.0	1120.0	1120.0	1120.0
	$\Gamma(T)$	2.3	8.5	2.7	3.0	8.5	8.2
	t (sec)	0.6	0.8	0.6	0.6	0.9	0.8
m100n500k4r1	$\Gamma^*(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	n	7184542.0	6987522.0	2006636.0	2056079.0	6993419.0	2046170.0
	$c^t\hat{y}$	-24.0	-24.0	-24.0	-24.0	-24.0	-24.0
	$\Gamma(T)$	28957.2	28896.8	28888.8	28890.0	28900.4	29028.8
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
macrophage	$\Gamma^*(T)$	146518.1	146136.8	215064.5	202953.9	146093.5	146068.1
	n	1251604.0	1233931.0	3329468.0	3632070.0	1236445.0	1239753.0
	$c^t\hat{y}$	375.0	376.0	389.0	381.0	376.0	376.0
	$\Gamma(T)$	10371.0	5015.6	28461.9	14051.9	5011.6	5012.4
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
manna81	$\Gamma^*(T)$	30.4	10.2	20.3	30.4	30.4	10.3
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	-13164.0	-13164.0	-13164.0	-13164.0	-13164.0	-13164.0
	$\Gamma(T)$	14.7	10.0	12.4	14.7	14.7	10.0
	t (sec)	0.8	0.4	0.6	0.8	0.8	0.6
map18	$\Gamma^*(T)$	3287.1	3879.2	3865.0	3855.5	3494.7	3934.9
	n	393.0	315.0	315.0	315.0	333.0	305.0
	$c^t\hat{y}$	-847.0	-847.0	-847.0	-847.0	-847.0	-847.0
	$\Gamma(T)$	3828.6	4000.9	3990.4	3976.2	4020.4	4000.4
	t (sec)	424.3	433.5	432.8	431.5	382.6	438.6

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings Problem		default	estim	log-n	log-it	oracle	rank-1
map20	$\Gamma^*(T)$	2763.1	2743.1	2782.0	2768.7	2746.8	2707.6
	n	299.0	299.0	299.0	299.0	319.0	315.0
	$c^t\hat{y}$	-922.0	-922.0	-922.0	-922.0	-922.0	-922.0
	$\Gamma(T)$	2792.9	2785.2	2836.7	2798.1	2770.5	2780.5
	t (sec)	335.0	333.3	338.0	334.4	333.9	327.6
markshare1	$\Gamma^*(T)$	720000.0	720000.0	720000.0	720000.0	720000.0	720000.0
	n	73327325.0	76824489.0	48123708.0	41054569.0	75830406.0	43086938.0
	$c^t\hat{y}$	5.0	4.0	7.0	7.0	4.0	3.0
	$\Gamma(T)$	602368.8	559538.0	622207.6	621340.7	559815.9	503482.0
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
markshare2	$\Gamma^*(T)$	720000.0	720000.0	720000.0	720000.0	720000.0	720000.0
	n	60920471.0	60781960.0	24897879.0	29294857.0	60892446.0	28720432.0
	$c^t\hat{y}$	13.0	12.0	10.0	13.0	12.0	13.0
	$\Gamma(T)$	675988.0	661153.5	654382.4	666935.4	661142.1	668024.1
	t (sec)	7200.1	7200.1	7200.0	7200.0	7200.0	7200.0
mas74	$\Gamma^*(T)$	2161.4	2268.0	6227.7	5913.8	2805.4	3517.6
	n	2834519.0	2767121.0	3954369.0	3630223.0	2760117.0	2594501.0
	$c^t\hat{y}$	11801.2	11801.2	11801.2	11801.2	11801.2	11801.2
	$\Gamma(T)$	79.7	332.6	779.0	584.0	331.8	32.5
	t (sec)	565.4	583.8	650.1	595.9	516.2	440.1
mas76	$\Gamma^*(T)$	82.0	96.0	93.3	207.7	278.7	133.5
	n	404939.0	471714.0	471714.0	695047.0	848147.0	436756.0
	$c^t\hat{y}$	40005.1	40005.1	40005.1	40005.1	40005.1	40005.1
	$\Gamma(T)$	8.5	9.1	8.6	9.1	9.1	7.6
	t (sec)	66.7	79.2	77.5	86.2	118.7	56.2
mcsched	$\Gamma^*(T)$	1240.3	1116.8	1272.4	1240.1	1103.0	1259.6
	n	19507.0	15565.0	13982.0	14980.0	15471.0	14275.0
	$c^t\hat{y}$	211913.0	211913.0	211913.0	211913.0	211913.0	211913.0
	$\Gamma(T)$	230.5	246.4	236.6	255.3	237.8	234.9
	t (sec)	211.9	172.9	155.6	164.7	173.4	159.9
mik-250-1-100-1	$\Gamma^*(T)$	744.6	738.2	742.5	999.9	944.9	1006.2
	n	943440.0	943440.0	943440.0	617742.0	595707.0	683166.0
	$c^t\hat{y}$	-66729.0	-66729.0	-66729.0	-66729.0	-66729.0	-66729.0
	$\Gamma(T)$	10.2	0.2	0.3	10.2	10.2	10.2
	t (sec)	365.1	362.7	364.3	245.6	236.8	278.7
mine-166-5	$\Gamma^*(T)$	431.7	453.0	436.1	446.0	427.0	425.8
	n	2045.0	2045.0	1996.0	2045.0	1997.0	2045.0
	$c^t\hat{y}$	-566395707.9	-566395707.9	-566395707.9	-566395707.9	-566395707.9	-566395707.9
	$\Gamma(T)$	1654.3	1626.2	1625.3	1627.4	1618.3	1608.6
	t (sec)	31.0	30.7	30.6	30.9	30.7	31.0
mine-90-10	$\Gamma^*(T)$	409.8	360.6	543.4	361.0	382.0	354.7
	n	77784.0	68094.0	92258.0	68094.0	67313.0	57851.0
	$c^t\hat{y}$	-784302337.6	-784302337.6	-784302337.6	-784302337.6	-784302337.6	-784302337.6
	$\Gamma(T)$	1944.8	1813.7	1834.8	1813.7	1815.5	1789.9
	t (sec)	256.3	228.4	270.8	229.0	232.2	199.4
misc03	$\Gamma^*(T)$	42.4	38.5	38.4	55.0	35.0	41.9
	n	139.0	123.0	123.0	123.0	137.0	170.0
	$c^t\hat{y}$	3360.0	3360.0	3360.0	3360.0	3360.0	3360.0
	$\Gamma(T)$	49.0	31.5	20.8	40.8	19.1	30.8
	t (sec)	1.2	1.1	1.1	1.4	1.0	1.2
misc06	$\Gamma^*(T)$	10.0	10.0	10.0	10.0	10.0	10.0
	n	4.0	4.0	4.0	4.0	6.0	4.0
	$c^t\hat{y}$	12850.9	12850.9	12850.9	12850.9	12850.9	12850.9
	$\Gamma(T)$	6.4	5.5	5.5	5.5	5.5	5.5
	t (sec)	0.7	0.5	0.7	0.7	0.8	0.5
misc07	$\Gamma^*(T)$	562.5	510.5	509.1	618.7	557.1	596.4
	n	21721.0	20003.0	14450.0	17854.0	15439.0	17292.0
	$c^t\hat{y}$	2810.0	2810.0	2810.0	2810.0	2810.0	2810.0
	$\Gamma(T)$	63.6	34.6	61.6	27.3	52.9	34.6
	t (sec)	14.4	13.1	10.4	12.8	11.2	12.3
mitre	$\Gamma^*(T)$	590.0	580.0	580.0	600.0	660.0	590.0
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	115155.0	115155.0	115155.0	115155.0	115155.0	115155.0
	$\Gamma(T)$	582.4	580.2	580.2	600.2	652.4	582.4

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings / Problem		default	estim	log-n	log-it	oracle	rank-1
	t (sec)	6.0	5.9	5.9	6.1	6.7	6.0
mkc	$\Gamma^*(T)$	1306.8	1296.7	3772.1	1327.7	1307.1	1340.1
	n	2524672.0	2989875.0	4028953.0	3786593.0	2985313.0	3871184.0
	$c^t\hat{y}$	-563.7	-563.6	-555.1	-557.4	-563.6	-559.6
	$\Gamma(T)$	2537.0	2067.8	11417.7	8522.4	2080.1	5789.1
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
mod008	$\Gamma^*(T)$	3.2	2.3	3.6	3.0	3.6	3.6
	n	7.0	7.0	7.0	7.0	4.0	7.0
	$c^t\hat{y}$	307.0	307.0	307.0	307.0	307.0	307.0
	$\Gamma(T)$	11.6	8.7	14.5	11.6	13.1	13.1
	t (sec)	0.9	0.7	1.1	0.9	1.2	1.1
mod010	$\Gamma^*(T)$	20.1	20.1	10.1	10.1	20.1	20.1
	n	2.0	7.0	7.0	7.0	7.0	7.0
	$c^t\hat{y}$	6548.0	6548.0	6548.0	6548.0	6548.0	6548.0
	$\Gamma(T)$	68.0	60.2	40.2	45.1	60.0	60.0
	t (sec)	0.8	0.7	0.6	0.6	0.7	0.7
mod011	$\Gamma^*(T)$	491.4	517.1	522.4	472.8	476.4	464.9
	n	1229.0	1229.0	1229.0	1045.0	1021.0	1068.0
	$c^t\hat{y}$	-54558535.0	-54558535.0	-54558535.0	-54558535.0	-54558535.0	-54558535.0
	$\Gamma(T)$	1540.6	1570.5	1562.3	1531.3	1559.4	1571.1
	t (sec)	176.8	180.3	178.1	156.9	152.0	145.8
modglob	$\Gamma^*(T)$	0.3	0.4	0.5	0.4	0.5	0.4
	n	905.0	905.0	905.0	820.0	739.0	820.0
	$c^t\hat{y}$	20740508.1	20740508.1	20740508.1	20740508.1	20740508.1	20740508.1
	$\Gamma(T)$	0.2	0.2	0.3	0.2	0.3	0.2
	t (sec)	1.4	1.5	1.7	1.7	1.5	1.3
momentum1	$\Gamma^*(T)$	66647.1	60364.6	53666.4	60183.5	60205.1	60150.5
	n	44070.0	15148.0	178458.0	19203.0	15305.0	15331.0
	$c^t\hat{y}$	115610.8	160511.2	154120.5	160511.2	160511.2	160511.2
	$\Gamma(T)$	113930.9	260453.6	255624.2	260091.0	260146.2	260052.7
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
momentum2	$\Gamma^*(T)$	69616.4	94193.3	95091.2	95147.0	80807.5	94353.1
	n	90508.0	99580.0	74591.0	85292.0	86526.0	74211.0
	$c^t\hat{y}$	12314.4	12315.1	13813.4	13813.5	12314.6	13813.8
	$\Gamma(T)$	85974.5	83308.9	112742.8	112815.2	82711.0	112704.8
	t (sec)	7200.0	7200.0	7200.1	7200.0	7200.0	7200.0
msc98-ip	$\Gamma^*(T)$	5430.7	5420.6	5410.7	5410.6	5410.6	5410.8
	n	3391.0	18438.0	10163.0	12229.0	10164.0	10162.0
	$c^t\hat{y}$	21655010.0	22273180.0	22273180.0	22273180.0	22273180.0	22273180.0
	$\Gamma(T)$	353627.0	147592.2	147430.4	147485.6	147355.8	147440.3
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
mspp16	$\Gamma^*(T)$	61662.4	63439.4	63198.2	63230.9	78986.1	63230.3
	n	51.0	57.0	57.0	57.0	47.0	57.0
	$c^t\hat{y}$	363.0	363.0	363.0	363.0	363.0	363.0
	$\Gamma(T)$	49036.0	49190.0	49042.0	49084.0	49030.0	49060.0
	t (sec)	2579.3	2841.5	2832.7	2838.1	5437.7	2838.0
mzzv11	$\Gamma^*(T)$	4336.8	4216.2	4373.7	4247.8	4278.5	4223.1
	n	1999.0	1999.0	4989.0	1999.0	1908.0	1975.0
	$c^t\hat{y}$	-21718.0	-21718.0	-21718.0	-21718.0	-21718.0	-21718.0
	$\Gamma(T)$	12450.4	12445.9	12652.1	12449.1	12449.1	12349.4
	t (sec)	267.9	266.0	357.4	268.0	258.1	262.7
mzzv42z	$\Gamma^*(T)$	3052.0	3019.6	3092.6	3051.0	3050.0	3037.6
	n	534.0	534.0	3281.0	534.0	536.0	1012.0
	$c^t\hat{y}$	-20540.0	-20540.0	-20540.0	-20540.0	-20540.0	-20540.0
	$\Gamma(T)$	12112.4	12014.9	12055.7	12110.9	12018.5	11988.8
	t (sec)	340.3	337.7	391.1	339.5	338.4	316.6
n3div36	$\Gamma^*(T)$	43102.2	43060.7	43099.5	50187.3	45913.6	43935.3
	n	250934.0	260655.0	260209.0	457915.0	372168.0	345004.0
	$c^t\hat{y}$	131000.0	130800.0	130800.0	135000.0	130800.0	130800.0
	$\Gamma(T)$	5793.9	5732.8	5759.8	25013.9	5758.1	5974.6
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
n3seq24	$\Gamma^*(T)$	7360.9	7520.3	7360.9	7410.7	7430.7	7420.7
	n	393.0	392.0	393.0	393.0	393.0	393.0
	$c^t\hat{y}$	61600.0	61600.0	61600.0	61600.0	61600.0	61600.0

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings / Problem		default	estim	log-n	log-it	oracle	rank-1
	$\Gamma(T)$	172367.6	174014.4	173777.5	173782.8	173563.3	173691.5
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
n4-3	$\Gamma^*(T)$	3286.1	3343.7	61688.7	3477.8	3975.2	5487.5
	n	32231.0	33154.0	806766.0	29912.0	29104.0	44646.0
	$c^t\hat{y}$	8993.0	8993.0	8993.0	8993.0	8993.0	8993.0
	$\Gamma(T)$	1409.3	1439.8	55145.9	1425.9	1443.4	1754.8
	t (sec)	542.3	564.0	6003.9	504.3	472.5	633.3
neos-1109824	$\Gamma^*(T)$	1397.6	1409.5	1399.3	1524.4	1231.8	1462.6
	n	21927.0	22678.0	14347.0	16305.0	10652.0	14781.0
	$c^t\hat{y}$	378.0	378.0	378.0	378.0	378.0	378.0
	$\Gamma(T)$	773.5	994.5	978.4	1004.9	981.8	984.9
	t (sec)	156.1	154.0	117.6	134.3	103.4	123.3
neos-1337307	$\Gamma^*(T)$	1127.4	2995.5	2685.5	2512.8	2506.2	1025.0
	n	370421.0	553458.0	514252.0	491428.0	519383.0	391710.0
	$c^t\hat{y}$	-202319.0	-202319.0	-202319.0	-202319.0	-202319.0	-202319.0
	$\Gamma(T)$	8200.6	5426.7	5289.8	5291.7	5280.8	5281.7
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
neos-1396125	$\Gamma^*(T)$	11732.0	15076.5	23658.8	29774.6	15160.0	27068.1
	n	61200.0	69115.0	68530.0	66297.0	59721.0	70372.0
	$c^t\hat{y}$	3000.0	3000.0	3000.0	3000.0	3000.0	3000.0
	$\Gamma(T)$	4047.4	5430.3	5405.8	5431.0	5570.4	5463.6
	t (sec)	766.1	925.5	911.6	899.7	856.8	778.5
neos-1601936	$\Gamma^*(T)$	1433.3	1433.3	1456.7	1446.7	1530.0	1423.3
	n	6755.0	1615.0	1609.0	1615.0	1606.0	1615.0
	$c^t\hat{y}$	4.0	6.0	6.0	6.0	6.0	6.0
	$\Gamma(T)$	251274.8	686858.2	688358.0	686858.2	688908.0	686958.2
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
neos-476283	$\Gamma^*(T)$	7894.7	8034.8	8004.8	7984.8	8124.8	7884.8
	n	685.0	685.0	685.0	685.0	667.0	855.0
	$c^t\hat{y}$	406.4	406.4	406.4	406.4	406.4	406.4
	$\Gamma(T)$	9009.9	9194.0	9167.6	9148.2	9298.1	9030.1
	t (sec)	275.9	282.0	281.1	281.3	279.6	279.8
neos-686190	$\Gamma^*(T)$	1453.1	1825.5	1819.9	1848.6	1770.5	1757.6
	n	7264.0	10378.0	9805.0	10400.0	9405.0	9445.0
	$c^t\hat{y}$	6730.0	6730.0	6730.0	6730.0	6730.0	6730.0
	$\Gamma(T)$	3954.3	1825.0	1828.5	1829.9	1866.9	1848.6
	t (sec)	93.8	118.6	114.9	119.6	110.5	109.8
neos-849702	$\Gamma^*(T)$	180.0	160.0	160.0	170.0	170.0	160.0
	n	6115.0	48917.0	48917.0	48917.0	48917.0	48917.0
	$c^t\hat{y}$	0.0	0.0	0.0	0.0	0.0	0.0
	$\Gamma(T)$	17471.0	55887.0	56100.0	55900.0	55967.0	55999.0
	t (sec)	174.7	558.8	561.2	559.2	559.6	559.9
neos-916792	$\Gamma^*(T)$	3591.9	3537.8	3572.6	114174.4	3480.3	9305.6
	n	106472.0	123066.0	123066.0	1726425.0	124088.0	210792.0
	$c^t\hat{y}$	31.9	31.9	31.9	32.3	31.9	31.9
	$\Gamma(T)$	938.9	923.6	920.9	15027.4	904.0	1075.7
	t (sec)	406.2	454.3	460.3	7200.0	399.2	593.9
neos-934278	$\Gamma^*(T)$	3221.1	3221.1	3221.1	3221.1	3251.0	3251.0
	n	889.0	1133.0	1145.0	81077.0	992.0	1095.0
	$c^t\hat{y}$	275.0	271.0	271.0	1283.0	271.0	271.0
	$\Gamma(T)$	133833.6	333188.8	332304.4	579814.8	341132.6	335158.6
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
neos13	$\Gamma^*(T)$	33785.0	41656.5	35058.8	41716.4	41652.2	42219.1
	n	4422.0	3230.0	4701.0	3230.0	3230.0	3209.0
	$c^t\hat{y}$	-95.5	-95.5	-95.5	-95.5	-95.5	-95.5
	$\Gamma(T)$	40072.4	44630.5	30221.1	44684.1	44623.2	45119.0
	t (sec)	1514.8	1727.5	1433.3	1730.0	1725.1	1738.6
neos18	$\Gamma^*(T)$	509.0	538.1	515.0	512.6	567.4	545.6
	n	6778.0	5601.0	5174.0	4841.0	5179.0	6249.0
	$c^t\hat{y}$	16.0	16.0	16.0	16.0	16.0	16.0
	$\Gamma(T)$	390.2	373.1	351.8	374.8	363.1	338.7
	t (sec)	32.1	29.8	29.2	28.5	31.9	31.6
net12	$\Gamma^*(T)$	135566.4	143794.4	144311.6	143694.8	155271.1	198889.8
	n	3864.0	4985.0	4985.0	4985.0	5016.0	4605.0

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings Problem		default	estim	log-n	log-it	oracle	rank-1
	$c^t\hat{y}$	214.0	214.0	214.0	214.0	214.0	214.0
	$\Gamma(T)$	91859.7	31752.4	31904.3	31717.4	31764.8	16073.1
	t (sec)	2532.6	2746.2	2756.3	2744.4	3018.6	3473.0
netdiversion	$\Gamma^*(T)$	88934.9	86721.9	87024.5	86789.2	86057.6	86149.9
	n	72.0	119.0	119.0	119.0	113.0	113.0
	$c^t\hat{y}$	251.0	242.0	242.0	242.0	242.0	242.0
	$\Gamma(T)$	358622.1	554572.7	554672.7	553972.7	554572.7	554572.7
	t (sec)	7200.0	6630.7	6634.4	6622.8	6581.5	6580.6
newdano	$\Gamma^*(T)$	83214.8	83697.5	132577.2	154842.9	112165.5	153952.0
	n	2083404.0	2083404.0	1804254.0	2176878.0	1993781.0	2002198.0
	$c^t\hat{y}$	65.7	65.7	65.7	65.7	65.7	65.7
	$\Gamma(T)$	1835.3	1846.3	2438.3	2581.7	1842.6	1647.9
	t (sec)	3557.7	3573.5	2763.5	3548.9	3704.2	3242.0
noswot	$\Gamma^*(T)$	825.1	837.9	814.9	1779.5	814.4	805.1
	n	829543.0	829543.0	467476.0	1250880.0	436956.0	455271.0
	$c^t\hat{y}$	-41.0	-41.0	-41.0	-41.0	-41.0	-41.0
	$\Gamma(T)$	10.0	12.7	10.2	566.3	14.1	12.9
	t (sec)	177.5	180.3	175.2	382.6	175.1	173.1
ns1208400	$\Gamma^*(T)$	187020.0	127900.0	128940.0	129650.0	151980.0	106290.0
	n	3118.0	2777.0	2770.0	2770.0	2785.0	2772.0
	$c^t\hat{y}$	2.0	2.0	2.0	2.0	2.0	2.0
	$\Gamma(T)$	180863.0	26900.0	27000.0	26900.0	26900.0	27000.0
	t (sec)	1870.2	1279.0	1289.4	1296.5	1519.8	1062.9
ns1688347	$\Gamma^*(T)$	12504.0	6524.3	9963.7	7777.3	7857.2	7727.9
	n	6667.0	2609.0	9975.0	4351.0	3905.0	4330.0
	$c^t\hat{y}$	27.0	27.0	27.0	27.0	27.0	27.0
	$\Gamma(T)$	33631.8	10151.3	12012.2	10365.2	10354.4	10263.8
	t (sec)	738.9	275.2	487.9	384.9	388.7	380.6
ns1758913	$\Gamma^*(T)$	78175.5	78124.6	78642.7	78351.2	78053.3	79121.5
	n	2.0	2.0	2.0	2.0	2.0	2.0
	$c^t\hat{y}$	-236.8	-236.8	-236.8	-236.8	-236.8	-236.8
	$\Gamma(T)$	613682.7	613682.2	613764.1	613715.3	613665.3	613842.1
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
ns1766074	$\Gamma^*(T)$	82440.0	84260.0	84730.0	84210.0	84420.0	84170.0
	n	942544.0	893992.0	893992.0	893992.0	893992.0	893992.0
	$\Gamma(T)$	82440.0	84260.0	84730.0	84210.0	84420.0	84170.0
	t (sec)	824.4	842.6	847.3	842.1	844.2	841.7
ns1830653	$\Gamma^*(T)$	17278.1	17348.0	15214.1	15278.2	15647.9	18374.2
	n	41218.0	46638.0	36733.0	38717.0	36114.0	46887.0
	$c^t\hat{y}$	20622.0	20622.0	20622.0	20622.0	20622.0	20622.0
	$\Gamma(T)$	8839.0	7288.8	6136.6	6143.6	6224.8	7889.9
	t (sec)	440.3	371.5	333.7	387.5	382.9	394.2
nsrand-ipx	$\Gamma^*(T)$	5735.0	5194.8	5199.9	8511.7	5216.1	5206.5
	n	1599798.0	1763180.0	1746790.0	2890013.0	1758600.0	1730377.0
	$c^t\hat{y}$	51840.0	51360.0	51360.0	51200.0	51360.0	51360.0
	$\Gamma(T)$	11873.2	5928.3	5954.1	5634.1	5970.0	5998.2
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
nw04	$\Gamma^*(T)$	967.3	986.4	976.8	967.2	980.6	997.7
	n	11.0	11.0	11.0	11.0	6.0	11.0
	$c^t\hat{y}$	16862.0	16862.0	16862.0	16862.0	16862.0	16862.0
	$\Gamma(T)$	1144.7	1159.7	1151.6	1146.5	1151.2	1174.7
	t (sec)	24.5	24.4	24.4	24.5	25.3	24.9
opm2-z7-s2	$\Gamma^*(T)$	11760.0	11726.0	37761.8	11828.8	11834.2	18626.1
	n	2092.0	2092.0	30810.0	2092.0	2094.0	15231.0
	$c^t\hat{y}$	-10280.0	-10280.0	-10280.0	-10280.0	-10280.0	-10280.0
	$\Gamma(T)$	8068.2	8138.1	9132.7	8118.2	8277.5	8477.0
	t (sec)	794.6	789.9	2017.4	796.7	794.6	1220.6
opt1217	$\Gamma^*(T)$	2.8	5.4	5.4	5.4	7.7	5.4
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	-16.0	-16.0	-16.0	-16.0	-16.0	-16.0
	$\Gamma(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	t (sec)	0.3	0.4	0.4	0.4	0.4	0.4
p0033	$\Gamma^*(T)$	0.3	0.4	0.4	0.4	0.3	0.3
	n	1.0	1.0	1.0	1.0	1.0	1.0

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings / Problem		default	estim	log-n	log-it	oracle	rank-1
	$c^t\hat{y}$	3089.0	3089.0	3089.0	3089.0	3089.0	3089.0
	$\Gamma(T)$	0.8	0.8	0.8	0.8	0.8	0.8
	t (sec)	0.1	0.1	0.1	0.1	0.1	0.1
p0201	$\Gamma^*(T)$	14.6	14.3	5.0	14.6	14.7	4.7
	n	67.0	67.0	67.0	67.0	59.0	65.0
	$c^t\hat{y}$	7615.0	7615.0	7615.0	7615.0	7615.0	7615.0
	$\Gamma(T)$	23.9	21.1	18.5	23.4	24.0	14.9
	t (sec)	1.8	1.7	1.8	1.8	1.8	1.7
p0282	$\Gamma^*(T)$	0.3	0.8	0.5	0.5	0.8	0.8
	n	3.0	3.0	3.0	2.0	1.0	3.0
	$c^t\hat{y}$	258411.0	258411.0	258411.0	258411.0	258411.0	258411.0
	$\Gamma(T)$	1.0	2.3	1.2	2.3	2.4	2.3
	t (sec)	0.3	0.6	0.3	0.8	0.8	0.6
p0548	$\Gamma^*(T)$	0.8	10.2	10.2	0.8	10.2	10.2
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	8691.0	8691.0	8691.0	8691.0	8691.0	8691.0
	$\Gamma(T)$	10.1	20.1	20.1	10.1	20.1	20.1
	t (sec)	0.1	0.3	0.3	0.2	0.3	0.3
p2756	$\Gamma^*(T)$	21.5	11.4	31.9	31.9	31.9	11.4
	n	137.0	9.0	9.0	9.0	9.0	9.0
	$c^t\hat{y}$	3124.0	3124.0	3124.0	3124.0	3124.0	3124.0
	$\Gamma(T)$	26.1	16.2	32.2	32.1	31.9	16.2
	t (sec)	1.6	1.2	1.6	1.5	1.4	1.2
pg5_34	$\Gamma^*(T)$	156.6	175.1	254.3	316.9	175.4	252.0
	n	291242.0	291323.0	301183.0	362704.0	273355.0	305210.0
	$c^t\hat{y}$	-14339.4	-14339.4	-14339.4	-14339.4	-14339.4	-14339.4
	$\Gamma(T)$	163.2	183.8	175.1	202.2	183.8	190.0
	t (sec)	1287.1	1338.1	1412.7	1689.3	1297.1	1400.7
pigeon-10	$\Gamma^*(T)$	72009.0	72009.0	72000.0	72000.0	72009.0	72009.0
	n	17116573.0	17457654.0	11244389.0	11397342.0	10565366.0	17502182.0
	$c^t\hat{y}$	-9000.0	-9000.0	-9000.0	-9000.0	-9000.0	-9000.0
	$\Gamma(T)$	520.0	500.0	520.0	490.0	500.0	520.0
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
pk1	$\Gamma^*(T)$	3399.4	3453.3	3371.8	7991.4	4329.0	4349.3
	n	284323.0	284323.0	284323.0	481791.0	281331.0	281341.0
	$c^t\hat{y}$	11.0	11.0	11.0	11.0	11.0	11.0
	$\Gamma(T)$	311.4	328.9	309.8	1191.2	343.5	347.9
	t (sec)	64.2	65.2	63.9	80.2	53.8	55.0
pp08a	$\Gamma^*(T)$	5.5	3.0	6.0	5.5	5.7	3.7
	n	221.0	225.0	225.0	225.0	231.0	253.0
	$c^t\hat{y}$	7350.0	7350.0	7350.0	7350.0	7350.0	7350.0
	$\Gamma(T)$	27.2	15.8	29.7	27.2	29.7	22.8
	t (sec)	1.3	1.1	1.6	1.3	1.4	1.3
pp08aCUTS	$\Gamma^*(T)$	2.8	3.4	4.8	3.8	5.5	3.2
	n	194.0	165.0	165.0	165.0	149.0	153.0
	$c^t\hat{y}$	7350.0	7350.0	7350.0	7350.0	7350.0	7350.0
	$\Gamma(T)$	19.8	30.1	36.1	32.4	32.8	22.9
	t (sec)	1.1	1.4	1.5	1.4	1.4	1.3
protfold	$\Gamma^*(T)$	146786.0	151335.2	151309.8	151426.1	151308.6	151293.3
	n	10226.0	11588.0	11587.0	11577.0	11587.0	11595.0
	$c^t\hat{y}$	-23.0	-20.0	-20.0	-20.0	-20.0	-20.0
	$\Gamma(T)$	251536.1	300859.4	300774.8	301088.4	300804.5	300826.7
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
pw-myciel4	$\Gamma^*(T)$	110490.5	111043.5	113480.4	130187.2	132077.2	157876.3
	n	712713.0	712713.0	291077.0	375464.0	368355.0	433819.0
	$c^t\hat{y}$	10.0	10.0	10.0	10.0	10.0	10.0
	$\Gamma(T)$	1816.4	1887.4	1789.5	1811.3	1878.4	1856.1
	t (sec)	3542.8	3550.0	1889.3	3041.3	2199.1	2629.2
qiu	$\Gamma^*(T)$	5951.6	6199.1	6113.8	6128.7	6272.9	6260.0
	n	12604.0	12618.0	12624.0	12559.0	12616.0	12629.0
	$c^t\hat{y}$	-132.9	-132.9	-132.9	-132.9	-132.9	-132.9
	$\Gamma(T)$	2123.9	2134.9	2123.9	1602.1	2153.9	2083.5
	t (sec)	79.9	81.6	79.0	74.1	77.8	77.2
qnet1	$\Gamma^*(T)$	40.1	22.5	35.4	34.2	35.2	22.7

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings Problem		default	estim	log-n	log-it	oracle	rank-1
	n	36.0	7.0	7.0	7.0	7.0	7.0
	$c^t\hat{y}$	16029.7	16029.7	16029.7	16029.7	16029.7	16029.7
	$\Gamma(T)$	97.6	54.6	72.8	66.7	70.7	54.6
	t (sec)	8.3	4.5	5.5	4.9	5.3	4.5
qnet1_o	$\Gamma^*(T)$	11.0	9.8	18.5	19.2	18.9	19.1
	n	16.0	6.0	6.0	6.0	6.0	6.0
	$c^t\hat{y}$	16029.7	16029.7	16029.7	16029.7	16029.7	16029.7
	$\Gamma(T)$	62.0	51.8	59.8	63.8	61.8	62.8
	t (sec)	6.6	5.1	5.1	5.5	5.3	5.4
rail507	$\Gamma^*(T)$	1246.6	1244.4	1242.7	1394.6	1230.5	1234.1
	n	799.0	799.0	799.0	1584.0	855.0	865.0
	$c^t\hat{y}$	174.0	174.0	174.0	174.0	174.0	174.0
	$\Gamma(T)$	1380.5	1379.9	1386.6	1506.5	1372.0	1372.0
	t (sec)	242.4	239.9	238.3	324.5	235.7	239.6
ran16x16	$\Gamma^*(T)$	838.1	813.8	816.2	3854.3	857.8	1343.3
	n	368022.0	346094.0	346094.0	855239.0	265832.0	373581.0
	$c^t\hat{y}$	3823.0	3823.0	3823.0	3823.0	3823.0	3823.0
	$\Gamma(T)$	69.6	105.8	105.4	871.9	106.6	162.5
	t (sec)	291.3	283.9	285.0	593.0	231.0	276.4
rd-rplusc-21	$\Gamma^*(T)$	719567.3	719567.3	719567.3	719567.3	719567.3	719567.3
	n	77078.0	61764.0	61811.0	61764.0	60360.0	70998.0
	$c^t\hat{y}$	165935.9	166009.7	166009.7	166009.7	166009.7	177205.0
	$\Gamma(T)$	60818.1	118211.1	117774.9	117817.0	120667.1	131330.8
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
reblock67	$\Gamma^*(T)$	558.5	549.1	701.9	559.5	554.0	538.3
	n	109664.0	109664.0	118698.0	67830.0	57072.0	105846.0
	$c^t\hat{y}$	-34630648.4	-34630648.4	-34630648.4	-34630648.4	-34630648.4	-34630648.4
	$\Gamma(T)$	1629.4	1701.9	1733.6	1619.2	1627.8	1617.9
	t (sec)	253.3	251.3	263.1	189.4	172.6	246.5
rentacar	$\Gamma^*(T)$	114.2	104.2	104.2	113.9	118.4	124.0
	n	4.0	4.0	4.0	4.0	4.0	4.0
	$c^t\hat{y}$	30356761.0	30356761.0	30356761.0	30356761.0	30356761.0	30356761.0
	$\Gamma(T)$	130.0	120.0	120.0	120.0	126.0	136.0
	t (sec)	2.7	2.6	2.6	2.6	3.5	2.8
rgn	$\Gamma^*(T)$	1.5	4.2	4.2	3.8	4.2	4.2
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	82.2	82.2	82.2	82.2	82.2	82.2
	$\Gamma(T)$	8.2	24.5	24.5	24.5	24.5	24.5
	t (sec)	0.1	0.3	0.3	0.2	0.3	0.3
rmatr100-p10	$\Gamma^*(T)$	1808.9	1806.0	1791.4	1814.5	1778.7	1757.8
	n	851.0	851.0	851.0	851.0	909.0	909.0
	$c^t\hat{y}$	423.0	423.0	423.0	423.0	423.0	423.0
	$\Gamma(T)$	952.9	942.9	937.4	957.7	957.4	941.8
	t (sec)	135.1	135.8	134.5	134.9	131.0	130.7
rmatr100-p5	$\Gamma^*(T)$	5293.7	6929.7	6915.4	6315.1	6131.5	6391.4
	n	420.0	451.0	451.0	447.0	483.0	439.0
	$c^t\hat{y}$	976.0	976.0	976.0	976.0	976.0	976.0
	$\Gamma(T)$	1492.9	1399.0	1392.4	1386.1	1392.0	1376.4
	t (sec)	302.9	304.2	303.9	276.1	267.6	280.0
rmine6	$\Gamma^*(T)$	665.5	387.9	419.9	1269.0	395.2	385.2
	n	2004491.0	742664.0	742446.0	1434625.0	736822.0	738018.0
	$c^t\hat{y}$	-457.2	-457.2	-457.2	-457.2	-457.2	-457.2
	$\Gamma(T)$	367.9	343.8	342.0	416.6	331.7	331.4
	t (sec)	6096.9	2287.9	2265.3	3930.6	2246.4	2263.0
rocII-4-11	$\Gamma^*(T)$	12279.0	7037.1	8611.2	9936.4	10788.3	10042.0
	n	40477.0	11718.0	17604.0	21292.0	30330.0	17308.0
	$c^t\hat{y}$	-6.7	-6.7	-6.7	-6.7	-6.7	-6.7
	$\Gamma(T)$	11909.9	3637.1	5323.9	8665.8	10136.1	7544.9
	t (sec)	463.1	204.7	252.4	329.1	433.9	299.9
rococoC10-001000	$\Gamma^*(T)$	5034.0	4558.5	4549.5	23711.3	4816.8	11986.9
	n	203201.0	174936.0	161306.0	393185.0	135810.0	224776.0
	$c^t\hat{y}$	11460.0	11460.0	11460.0	11460.0	11460.0	11460.0
	$\Gamma(T)$	752.7	920.8	921.8	5329.4	941.2	2245.8
	t (sec)	1217.3	1011.1	966.6	2015.8	876.6	1274.5

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings Problem		default	estim	log-n	log-it	oracle	rank-1
roll3000	$\Gamma^*(T)$	6626.7	3522.8	8267.9	23828.8	2892.8	19509.0
	n	2781398.0	1063691.0	1089420.0	2488274.0	423972.0	2331855.0
	$c^t\hat{y}$	12899.0	12890.0	12890.0	12890.0	12890.0	12890.0
	$\Gamma(T)$	1534.3	1083.3	2656.1	7615.9	1054.9	4709.2
	t (sec)	7200.0	3082.7	2879.0	6007.8	1387.3	5522.8
rout	$\Gamma^*(T)$	224.3	215.0	360.8	781.9	204.9	207.2
	n	26664.0	20646.0	51961.0	101606.0	18547.0	18646.0
	$c^t\hat{y}$	1077.6	1077.6	1077.6	1077.6	1077.6	1077.6
	$\Gamma(T)$	162.8	190.5	225.9	466.6	184.3	190.1
	t (sec)	40.1	32.2	48.2	91.4	27.9	28.2
satellites1-25	$\Gamma^*(T)$	48757.5	74492.3	74623.6	74244.3	57389.7	68402.2
	n	3064.0	2648.0	2648.0	2138.0	1588.0	2212.0
	$c^t\hat{y}$	-5.0	-5.0	-5.0	-5.0	-5.0	-5.0
	$\Gamma(T)$	31264.0	32148.0	32104.0	32176.0	32108.0	45240.0
	t (sec)	660.2	995.5	997.1	992.0	765.3	913.5
set1ch	$\Gamma^*(T)$	2.5	4.6	4.5	4.3	5.3	4.0
	n	9.0	9.0	9.0	9.0	9.0	9.0
	$c^t\hat{y}$	54537.8	54537.8	54537.8	54537.8	54537.8	54537.8
	$\Gamma(T)$	13.9	16.7	15.7	18.2	21.1	13.5
	t (sec)	0.7	0.8	0.8	0.9	0.9	0.7
seymour	$\Gamma^*(T)$	15701.9	15933.6	25570.9	22115.1	15928.6	15953.0
	n	150798.0	146737.0	429173.0	430158.0	146590.0	146047.0
	$c^t\hat{y}$	423.0	424.0	431.0	431.0	424.0	424.0
	$\Gamma(T)$	4737.7	2925.2	13909.1	13946.0	2927.8	2946.7
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
sp97ar	$\Gamma^*(T)$	15504.2	47703.3	47997.5	47703.2	48193.8	47725.6
	n	6289.0	8065.0	8126.0	8147.0	8101.0	7619.0
	$c^t\hat{y}$	710063477.9	696853271.7	696853271.7	696853271.7	696853271.7	710898758.9
	$\Gamma(T)$	60048.1	72299.2	72315.0	72166.5	72495.0	72262.6
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
sp98ic	$\Gamma^*(T)$	3463.5	3263.9	3899.3	5771.1	3265.7	3399.2
	n	131473.0	137628.0	295850.0	300865.0	137322.0	185344.0
	$c^t\hat{y}$	452431468.0	450852689.3	452206394.1	450038459.8	450852689.3	450852689.3
	$\Gamma(T)$	18843.6	5858.7	7390.8	5964.0	5856.4	5888.1
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
sp98ir	$\Gamma^*(T)$	236.9	224.6	220.9	234.1	220.6	204.4
	n	8210.0	5379.0	5195.0	5379.0	5375.0	4630.0
	$c^t\hat{y}$	219676790.4	219676790.4	219676790.4	219676790.4	219676790.4	219676790.4
	$\Gamma(T)$	378.2	458.9	445.7	458.5	462.1	442.6
	t (sec)	106.6	86.3	82.2	85.9	81.3	69.2
stein27	$\Gamma^*(T)$	27.8	30.6	27.8	27.8	33.3	19.4
	n	3905.0	3905.0	3905.0	3905.0	3973.0	3607.0
	$c^t\hat{y}$	18.0	18.0	18.0	18.0	18.0	18.0
	$\Gamma(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	t (sec)	1.0	1.1	1.0	1.0	1.2	0.7
stein45	$\Gamma^*(T)$	207.5	225.0	214.2	232.2	268.3	283.9
	n	47352.0	47352.0	47352.0	48444.0	50336.0	46693.0
	$c^t\hat{y}$	30.0	30.0	30.0	30.0	30.0	30.0
	$\Gamma(T)$	4.5	6.6	5.1	5.1	5.1	5.4
	t (sec)	12.2	13.1	12.5	11.4	11.3	10.7
stp3d	$\Gamma^*(T)$	248584.7	249365.6	248584.7	249658.4	248779.9	248389.5
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$\Gamma(T)$	720000.0	720000.0	720000.0	720000.0	720000.0	720000.0
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
swath	$\Gamma^*(T)$	105447.4	96980.1	132728.6	126898.1	96898.0	110113.1
	n	1231280.0	1214072.0	1614555.0	1597955.0	1220123.0	1672194.0
	$c^t\hat{y}$	472.6	476.1	473.1	484.4	476.1	490.6
	$\Gamma(T)$	14764.9	19927.5	9564.9	26605.2	19873.7	34175.3
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
tanglegram1	$\Gamma^*(T)$	89159.2	87762.2	88550.5	88460.0	87629.7	88029.6
	n	37.0	37.0	37.0	37.0	37.0	37.0
	$c^t\hat{y}$	5182.0	5182.0	5182.0	5182.0	5182.0	5182.0
	$\Gamma(T)$	15880.2	15738.7	15847.9	15803.7	15706.3	15784.6
	t (sec)	1161.2	1138.7	1150.6	1149.9	1138.6	1142.8
tanglegram2	$\Gamma^*(T)$	1397.2	1415.6	1377.7	1397.3	1377.7	1377.7

continued on next page

Table H.9: Instancewise results of the experiment in Section 5.3. $\Gamma^*(T)$: dual integral, n: number of branch-and-bound nodes, $c^t\hat{y}$: primal bound at termination, $\Gamma(T)$: primal integral, t (sec): overall solving time.

Settings Problem		default	estim	log-n	log-it	oracle	rank-1
	n	5.0	5.0	5.0	5.0	5.0	5.0
	$c^t\hat{y}$	443.0	443.0	443.0	443.0	443.0	443.0
	$\Gamma(T)$	711.1	720.2	699.1	711.2	699.1	697.8
	t (sec)	14.9	15.1	14.7	14.9	14.7	14.7
timtab1	$\Gamma^*(T)$	5932.4	5559.8	10933.6	7761.4	7258.5	11280.7
	n	870361.0	868207.0	985756.0	963713.0	896679.0	965573.0
	$c^t\hat{y}$	764772.0	764772.0	764772.0	764772.0	764772.0	764772.0
	$\Gamma(T)$	385.4	412.6	601.8	492.1	380.8	356.2
	t (sec)	390.1	388.1	381.4	399.4	376.2	377.6
timtab2	$\Gamma^*(T)$	237001.7	326894.8	332626.7	331415.7	245532.2	326896.4
	n	9144342.0	14814841.0	14897005.0	15049120.0	9027482.0	14855771.0
	$c^t\hat{y}$	1136721.0	1208245.0	1255290.0	1188906.0	1138052.0	1208245.0
	$\Gamma(T)$	38386.2	67445.2	107488.7	66581.0	43432.2	67212.9
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
tr12-30	$\Gamma^*(T)$	191.9	173.2	398.4	531.3	196.8	294.2
	n	1471731.0	1186814.0	1788765.0	1832526.0	1162158.0	1306275.0
	$c^t\hat{y}$	130596.0	130596.0	130596.0	130596.0	130596.0	130596.0
	$\Gamma(T)$	84.8	93.4	101.9	130.2	101.3	102.3
	t (sec)	1814.6	1521.3	1927.2	2056.2	1505.3	1454.0
triptim1	$\Gamma^*(T)$	13000.0	12800.0	12900.0	12800.0	12700.0	12700.0
	n	47.0	4.0	4.0	4.0	4.0	4.0
	$c^t\hat{y}$	22.9	22.9	22.9	22.9	22.9	22.9
	$\Gamma(T)$	195940.0	98168.0	98677.0	98300.0	99763.0	98172.0
	t (sec)	2791.3	981.7	986.8	983.0	997.6	981.7
unitcal_7	$\Gamma^*(T)$	3015.6	2979.1	3020.8	2990.2	3036.7	3050.8
	n	23265.0	27125.0	27125.0	27125.0	19216.0	19861.0
	$c^t\hat{y}$	19635558.2	19635558.2	19635558.2	19635558.2	19635558.2	19635558.2
	$\Gamma(T)$	8237.2	8204.4	8320.9	8234.8	8238.8	8257.5
	t (sec)	1304.6	1469.5	1476.6	1477.2	1230.3	1271.4
vpm1	$\Gamma^*(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	n	1.0	1.0	1.0	1.0	1.0	1.0
	$c^t\hat{y}$	20.0	20.0	20.0	20.0	20.0	20.0
	$\Gamma(T)$	0.0	0.0	0.0	0.0	0.0	0.0
	t (sec)	0.0	0.0	0.0	0.0	0.0	0.0
vpm2	$\Gamma^*(T)$	8.4	6.6	7.2	16.6	10.5	9.0
	n	294.0	218.0	218.0	230.0	224.0	206.0
	$c^t\hat{y}$	13.8	13.8	13.8	13.8	13.8	13.8
	$\Gamma(T)$	20.5	17.5	17.5	24.9	26.2	20.5
	t (sec)	1.3	1.2	1.3	1.3	1.7	1.4
vpphard	$\Gamma^*(T)$	720000.0	720000.0	720000.0	720000.0	720000.0	720000.0
	n	7476.0	6321.0	233249.0	98462.0	6292.0	6321.0
	$c^t\hat{y}$	9.0	30.0	25.0	19.0	30.0	30.0
	$\Gamma(T)$	402227.3	605195.8	589146.0	586065.7	605180.1	605182.1
	t (sec)	7200.0	7200.0	7200.0	7200.0	7200.0	7200.0
zib54-UUE	$\Gamma^*(T)$	37084.8	49612.0	62041.5	61204.5	58244.4	58701.5
	n	539744.0	706521.0	314461.0	316760.0	296047.0	294878.0
	$c^t\hat{y}$	10334015.8	10334015.8	10334015.8	10334015.8	10334015.8	10334015.8
	$\Gamma(T)$	1071.3	1115.0	2127.0	2164.6	1110.9	2147.6
	t (sec)	3829.7	5375.5	2630.8	2735.5	2703.8	2625.2