Primal Heuristics for Mixed Integer Programs

Diplomarbeit
bei Prof. Dr. Dr. h.c. M. Grötschel

vorgelegt von Timo Berthold [1]
Fachbereich Mathematik der
Technischen Universität Berlin

Berlin, 11. September 2006

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

Contents

Acknowledgments

First of all I wish to thank my parents and Friederike C. Häckl for all the love and support they gave me. I thank Annegret Dix, Annette Mura, and Tim Januschowski for the best office atmosphere I could imagine.

I wish to thank Benjamin Hiller, Kati Wolter, Axel Metzler, and Marieluise Häckl for reading parts of the preliminary version and showing me how many mistakes one person can make on 123 pages. Especially, I thank Tobias Achterberg for all the advice and encouragement he gave me. It was a pleasure to work with you.

Furthermore, I wish to thank Prof. Dr. Martin Grötschel for supervising this thesis.

Zusammenfassung

Die vorliegende Arbeit beschäftigt sich mit Primalheuristiken für Gemischt-Ganzzahlige Programme (MIPs).

Diverse praktische Problemstellungen lassen sich mit Methoden der Kombinatorischen Optimierung modellieren. Kombinatorische Optimierungsprobleme können häufig effektiv als MIP formuliert werden (siehe [54]). Das Lösen eines MIPs ist ein $\mathcal{NP}$-vollständiges Optimierungsproblem. Eine Standardmethode zum Lösen von MIPs ist das Branch-And-Cut-Verfahren, welches eine Kombination der exakten Methoden des Branch-And-Bound und der Schnittebenenverfahren ist.

Moderne MIP-Löser wie Cplex oder das in dieser Arbeit verwendete SCIP bedienen sich zusätzlich einer Reihe von Methoden, welche in der Praxis häufig zu einer deutlichen Beschleunigung des Lösungsverfahrens führen. Primalheuristiken dienen dabei dem frühzeitigen Auffinden zulässiger Lösungen (Startheuristiken) oder der Konstruktion besserer Lösungen aus bereits vorhandenen (Verbesserungsheuristiken).

Wir geben zunächst einen Literaturüberblick und geben dann eine kurze Einführung in das MIP-Solving-Framework SCIP, in welches die in dieser Arbeit beschriebenen Heuristiken implementiert wurden.

Anschliessend stellen wir verschiedene Heuristiken ausführlicher vor. Zum einen konzentrieren wir uns dabei auf solche, die in der Literatur der letzten 10 Jahre beschrieben wurden: Die Feasibility Pump, Octane, Local Branching und Rins. Im Zuge dessen stellen wir eine verbesserte Version der Feasibility Pump vor, die wir auch schon in [3] beschrieben haben, sowie eine leichte modifizierte Version von Octane.

Zum anderen präsentieren wir zwei neue Heuristiken: Rens, eine Startheuristik, die Rundemethoden mit Nachbarschaftssuche kombiniert sowie Crossover, eine Verbesserungsheuristik, die Ähnlichkeiten zwischen verschiedenen bereits gefundenen Lösungen ausnutzt. Desweiteren stellen wir verschiedene konkrete Implementierungen der allgemeinen heuristischen Ideen des Rundens und des Diving vor.

In jedem Unterkapitel werden dabei neben den Ideen und der ausführlichen Beschreibung der jeweiligen Heuristik eine algorithmische Darstellung, Ergebnisse aus Testrechnungen sowie meist eine Illustration gegeben.

Die Arbeit schließt mit einem Vergleich der 15 Heuristiken, die standardmäßig in SCIP aktiviert sind. Wir untersuchen dabei den Einfluss der einzelnen Heuristiken auf den Gesamtverlauf von SCIP.

Chapter 1

Introduction

Mixed Integer Programming is a powerful tool to model and solve hard combinatorial optimization problems. Chip design and verification, network design, line planning, frequency assignment, various types of scheduling and many other combinatorial optimization problems (see e.g., Heipcke [38]) have been successfully modelled as Mixed Integer Programs, shortly MIPs. Since MIP-solving is $\mathcal{NP}$-hard [54], heuristic methods for it are of high interest.

Although a lot of heuristic ideas have been published in the recent years, few attempts have been made to summarize these and compare their performance to each other. This thesis gives an overview of a couple of MIP heuristics that have been developed during the last ten years. We introduce a new start heuristic which we named Rens and a new improvement heuristic called Crossover. The latter one has been independently developed by Rothberg [52]. Furthermore, we will present an extension of the Feasibility Pump heuristic by Bertacco, Fischetti, and Lodi [14] and some general diving and rounding heuristics.

We implemented all these heuristics into a MIP-solving framework called SCIP, see Achterberg [1], and ran extensive tests in order to evaluate their performance.

This thesis is organized as follows. In the remainder of this chapter we give basic definitions and a brief introduction into primal heuristics. In the second chapter, we describe the framework into which we integrated our implementations and the set of instances on which we ran our tests. The third and fourth chapter present several heuristics. The former concentrates on start heuristics, the latter on improvement heuristics.

For each heuristic, we begin with a short description of the main idea, mostly supplemented by an illustration and an algorithmic outline. Next, we fill in the implementation details, before subsuming the procedure into an algorithm. Finally, we present and discuss computational results.

The fifth chapter demonstrates the impact of the heuristics when we integrated them into SCIP.

1.1 Definitions

This section provides definitions of the most important terms that we use in this thesis. For a detailed introduction into linear and integer programming and combinatorial optimization see for example [20, 35, 54].

Let $\hat{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$.

Definition 1.1 *Let $m, n \in \mathbb{N}$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $c \in \mathbb{R}^n$, $l, u \in \hat{\mathbb{R}}^n$, and $I \subseteq N = \{1, \ldots, n\}$.*

$$\begin{array}{lrl} \min & c^T x & \\ \textit{such that} & Ax & \leq b \\ & l \leq x & \leq u \\ & x_j & \in \mathbb{Z} \quad \textit{for all } j \in I \end{array} \tag{1.1}$$

is called a mixed integer program (MIP).

We call $c^T x$ the *objective function*, shortly the objective, of the MIP. l and u are called the *lower* and *upper bounds* of the *variables* x. We call $l \leq x \leq u$ the *bounding constraints*, $Ax \leq b$ the *linear constraints*, and $x_j \in \mathbb{Z}$ for all $j \in I$ the *integrality constraints* of the MIP. A row $A_{i\cdot}$ of the matrix A is often identified with the linear constraint $A_{i\cdot} x \leq b_i$.

Let $B := \{j \in I \mid l_j = 0, u_j = 1\}$. We call $\{x_j \mid j \in I\}$ the set of *integer variables*, $\{x_j \mid j \in B\}$ the set of *binary variables*, $\{x_j \mid j \in I \setminus B\}$ the set of *general integer variables*, $\{x_j \mid j \in N \setminus I\}$ the set of *continuous variables*.

Naturally, if an integer variable has a fractional upper bound u_j, the bound can be strengthened to $\lfloor u_j \rfloor$. The same holds for fractional l_j and $\lceil l_j \rceil$. Due to this, we assume that $l_j, u_j \in \mathbb{Z}$, for all $j \in I$.

Definition 1.2 *A MIP given in the form* (1.1) *is called*

$$\begin{array}{rll} \textit{a} \text{ linear program (LP)} & \textit{if} & I = \emptyset, \\ \textit{an} \text{ integer program (IP)} & \textit{if} & I = N, \\ \textit{a} \text{ binary program (BP)} & \textit{if} & B = I = N, \textit{ and} \\ \textit{a} \text{ mixed binary program (MBP)} & \textit{if} & B = I. \end{array}$$

Definition 1.3 *Let $\hat{x} \in \mathbb{R}^n$. We call $\hat{x}$*

$$\begin{array}{rll} \text{LP-feasible } \textit{for } (1.1) & \textit{if} & A\hat{x} \leq b \textit{ and } l \leq \hat{x} \leq u, \\ \text{integer feasible } \textit{for } (1.1) & \textit{if} & \hat{x}_j \in \mathbb{Z} \quad \textit{for all } j \in I, \\ \textit{a} \text{ feasible solution } \textit{for } (1.1) & \textit{if} & \hat{x} \textit{ is LP-feasible and integer feasible, and} \\ \textit{an} \text{ optimal solution } \textit{for } (1.1) & \textit{if} & \hat{x} \textit{ is a feasible solution and } c^T\hat{x} \leq c^T x \\ & & \textit{for all other feasible solutions } x. \end{array} \tag{1.2}$$

The terms *LP-infeasible, integer infeasible,* and *infeasible vector* are defined analogously. If a MIP is given, the LP which arises by omitting the integrality constraints ($x_j \in \mathbb{Z}$ for all $j \in I$) is called the *LP-relaxation* of the MIP. The set $P(A, b, l, u) := \{x \in \mathbb{R}^n \mid Ax \leq b, l \leq x \leq u\}$ is called the *LP-polyhedron* of (1.1).

Definition 1.4 *Let $x \in \mathbb{R}^n$ and I the index set of integer variables of a given MIP. We call $f(x_j) := |x_j - \lfloor x_j + 0.5 \rfloor|$ the* fractionality *of the variable x_j, $j \in I$ and*

$$f(x) := \sum_{j \in I} f(x_j)$$

the fractionality *of the vector x. Let $A_{i\cdot}$ be a row of the matrix A. We call $\beta_i(x) := A_{i\cdot}x$ the* activity *of the row $A_{i\cdot}$, and*

$$v_i(x) := \begin{cases} 0 & \text{if } \beta_i(x) \leq b_i \\ \beta_i(x) - b_i & \text{if } \beta_i(x) > b_i \end{cases}$$

the violation *of the row $A_{i\cdot}$.*

Obviously, a vector x is integer feasible if and only if $f(x) = 0$, and LP-feasible, if and only if $l \leq x \leq u$ and $v_i(x) = 0$ for all $i \in \{1, \ldots, m\}$. A variable x_j, $j \in I$, with $f(x_j) \neq 0$ is called *fractional*.

Especially for rounding and diving heuristics it is of interest, whether a fractional variable can be rounded up or down without violating any constraint or how many constraints could possibly be violated. This motivates the following definitions:

Definition 1.5 *Let a MIP in the form 1.1 be given.*

1. *A variable x_j is called* trivially down-roundable*, if all coefficients a_{ij} of the corresponding column of the matrix A are nonnegative, hence $A_{\cdot j} \geq 0$.*

2. *A variable x_j is called* trivially up-roundable*, if all coefficients a_{ij} of the corresponding column of the matrix A are nonpositive, hence $A_{\cdot j} \leq 0$.*

3. *A variable is called* trivially roundable*, if it is trivially down-roundable or trivially up-roundable.*

4. *The number of negative coefficients a_{ij} of a column $A_{\cdot j}$ is called the number of* down-locks *of the variable x_j.*

5. *The number of positive coefficients a_{ij} of a column $A_{\cdot j}$ is called the number of* up-locks *of the variable x_j.*

6. *The minimum of the number of up-locks and the number of down-locks of x_j is called the number of* locks *of x_j.*

Sometimes, we will talk about the minima of more general functions or sets than those given in Definition 1.1.

Definition 1.6 *Let $X \subseteq \mathbb{R}^n$ be some closed set and $g : X \to \mathbb{R}$ a real-valued function on X.*

$$x' = argmin\{g(x)\} :\Leftrightarrow x' \in X \wedge g(x') = \min\{g(x) \mid x \in X\} \tag{1.3}$$

If we evaluate the quality of a feasible solution with respect to the objective function, we will consider the relative *gap* to the objective value of an optimal solution. One could also consider the relative gap to the proven dual bound, if the MIP-solving process has not been finished yet.

Definition 1.7 *Let x be a feasible solution of a given MIP, $x^\star$ an optimal or best known solution of this MIP, and $b^T y^\star$ the current dual bound during a MIP solving process.*

$$\gamma_P(x) := \begin{cases} 0 & \text{if } c^T x = c^T x^\star = 0 \\ \infty & \text{if } c^T x > c^T x^\star = 0 \\ 100 \cdot \frac{(c^T x - c^T x^\star)}{|c^T x^\star|} & \text{else} \end{cases}$$
$$\gamma_{PD}(x) := \begin{cases} 0 & \text{if } c^T x = b^T y^\star = 0 \\ \infty & \text{if } c^T x > b^T y^\star = 0 \vee c^T x \cdot b^T y^\star < 0 \\ 100 \cdot \frac{(c^T x - b^T y^\star)}{|b^T y^\star|} & \text{else} \end{cases} \tag{1.4}$$

$\gamma_P(x)$ is called the primal gap *of the solution x, $\gamma_{PD}(x)$ is called the* primal-dual gap *of x.*

If we want to compare different heuristics or settings, we also want to state how good they are "on average". Taking the arithmetic mean seems not practical to us, since the values we want to compare differ heavily in their magnitude. For example, the number of solving nodes lies in a range of 1 node to 50 million nodes. The arithmetic mean would only depend on the "huge" numbers. Therefore, we decided to use the geometric mean, but in a slightly modified form:

Definition 1.8 *Let $x^1, \ldots, x^n$ be a sequence of nonnegative real numbers.*

$$\mu := \sqrt[n]{\prod_{i=1}^{n} \max(x^i, 1)} \tag{1.5}$$

is called the geometric mean of $x^1, \ldots, x^n$.

We take the maximum of a number and one, in order to avoid difficulties caused by measuring errors of values near to zero. For example, whether the measured running time is 0.01 or 0.02 seconds could just be due to an imprecise measuring. Nevertheless, this would have the same impact as if the running time would be 1 hour or 2 hours. The above definition avoids this trouble or reduces it to an acceptable size, respectively.

1.2 A Brief History of MIP Heuristics

The word heuristic is derived from the Greek word $\epsilon\upsilon\rho\iota\sigma\kappa\epsilon\iota\nu$, which means 'to find something'. Heuristics are procedures which try to find good solutions in a short time by orientating themselves on some information about the problem which seems helpful to lead to the desired result.

In the sense of this very general and vague definition, we all apply heuristic methods quite frequently. For example, if we want to buy a new PC, we decide on the model to chose and the shop to buy it, after having considered only a small part of the available information. Collecting and evaluating all offers would probably take too much time, such that our PC is already out of fashion, until we are sure, which one is the optimal for us. Instead of, we orientate ourselves on some criteria which seem reasonable to us: maybe we read some test reports, we ask a neighbor who is interested in computers and bought a new one just a month ago. We could also follow the salesman's advice at our favorite computer shop who hopefully also wants to maximize our benefit and not merely his own profit.

The salesman willing to optimize his profit brings us back to mathematics. The *Traveling Salesman Problem*, briefly called *TSP*, was one of the first problems in combinatorial optimization, to which heuristics were efficiently applied and often succeeded in finding an optimal solution in a time quite small compared to the time it needs to prove the optimality of this solution. Among others, Grötschel and Padberg [36] give a short introduction into TSPs and present some simple heuristic ideas, an extensive discussion of the TSP was published by Gutin and Punnen [37]. The Nearest Neighbor heuristic and the Christofides Algorithm [19] are well known start heuristics for the TSP.

The k-OPT-algorithm is an improvement heuristic which was originally designed for the TSP, but variants of this are used for several other combinatorial optimization problems. It also formed the basis for the Lin-Kernighan-Heuristic [43] which is one of the most common algorithms used to find high quality solutions for TSPs in practice. Both, the Christofides and the k-OPT heuristic were developed in the 1970s.

Some years before, in 1958, Gomory [34] claimed the basic ideas of integer programming.

In 1969, Hillier [39] gave an algorithm for IPs without equality constraints which searches for feasible solutions along a line connecting an LP-feasible

and an integer feasible solution. He also included a k-OPT improvement heuristic into it.

In 1980, Balas and Martin [11] presented a heuristic called Pivot-and-Complement which was developed for BPs. It is based on the observation that, in the nomenclature of the simplex algorithm, an LP-feasible solution of which all basic variables are slack variables is also integer feasible. It performs pivot operations which drive the integer variables out of the basis and the slacks into the basis. Six years later, the same authors [12] gave an extension of this heuristic called Pivot-and-Shift which could be applied to general MIPs. In 2004, Balas, Schmieta, and Wallace [13] published an improved version of Pivot-and-Shift. It contains more pivot types and new rules for selecting them, an extension of the shifting procedure, and a neighborhood search related to Local Branching (see [25] and Chapter 4.1).

In 1992, Saltzman and Hillier [53] presented a so-called Heuristic Ceiling Point Algorithm which was restricted to IPs without equality constraints. It enumerates integer feasible points which are near or on the boundary of $P(A, b, l, u)$ and close to the optimum of the LP-relaxation.

Scatter Search with Star Paths is a diversification heuristic which is based on the work of Glover and Laguna [29, 30] from 1997 and was further improved by Glover, Løkketangen, and Woodruff [32] in 2000. It creates a couple of points which are then linked by paths along which feasible solutions are searched. The basic ideas of path relinking were introduced by Ibaraki, Ohashi, and Mine [40] in the 1970s. The main goal of Scatter Search is to diversify the set of solutions and not improving the incumbent.

In the year of 2001, Balas, Ceria, Dawande, Margot, and Pataki [10] published OCTANE, in full words OCTAhedral Neighborhood Search. This heuristic for BPs will be discussed in Chapter 3.3. It is based on a ray shooting algorithm starting at the LP-optimum and hitting facets of the octahedron dual to the unit hypercube.

In the same year, Eckstein, and Nediak presented Pivot-Cut-and-Dive, a heuristic for MBPs. Its main procedure is a rounding method which is based on simplex pivot operations, supplemented by explicit probing techniques, cut generation and a diving heuristic. As well as the other pivot based heuristics, this one was not implemented into SCIP and will therefore not be treated in this thesis. This is due to the fact that the underlying LP-solver of SCIP is seen as a black-box and hence, features like pivot selection are not supported.

During the last years, a couple of Large Neighborhood Search heuristics (see Chapter 4) have been presented. One of the first was Local Branching which was published by Fischetti and Lodi [25] in 2003. One year later, Danna, Rothberg, and Le Pape [21] presented RINS, also known as Relaxation Induced Neighborhood Search. The evolutionary algorithm of Rothberg [52] is also combined of two Large Neighborhood Search heuristics which we will call Crossover and Mutation. We will present an alternative variant of the Crossover heuristic in Chapter 4.3, and another Large Neighborhood Search heuristic, called RENS in Chapter 3.2.1. With the exception of RENS, all

Large Neighborhood Search heuristics try to improve an incumbent solution of a MIP (or an MBP in the case of Local Branching) by solving a sufficiently smaller sub-MIP which promises to contain feasible solutions of high quality.

The Feasibility Pump is one of the most recently investigated heuristics. It creates two sequences of points, which hopefully converge to a common point. One of the sequences is LP-feasible, the other one is integer feasible, The Feasibility Pump was introduced by Fischetti, Glover, and Lodi [24] in 2005 and further improved by Bertacco, Fischetti, and Lodi [14] and by Achterberg, and Berthold [3]. We will present these improvements in Chapter 3.1.2.

In addition, a couple of meta-heuristics which could also be specified and applied to MIP-solving were described in literature. Amongst others, these are Tabu Search [28, 31], Local Search [57], Simulated Annealing [42, 56], and Evolutionary Algorithms [8, 47]. For example, Løkketangen [45] gives an introduction to meta-heuristics for MBP-solving.

Chapter 2

Integration Into SCIP

In the following chapters, we will present several primal heuristics. We will always describe the theoretical ideas, followed by implementation details and computational results. In order to evaluate the performance of the heuristics, we integrated implementations of them into the MIP-solving framework SCIP, which is to be shortly explained in this chapter. Furthermore, we will present the set of MIP instances on which we performed our tests.

2.1 SCIP – a MIP-Solver

SCIP [1] is a framework created to solve *Constraint Integer Programs*, shortly called *CIPs,* which denotes an integration of Constraint Programming (see for example [7]) and Mixed Integer Programming. An exact definition and a discussion of CIPs is given in [2]. Since MIPs are a sub-category of CIPs, and since the current distribution of SCIP primarily contains algorithms for MIP-solving, we want to regard SCIP as a MIP-solver.

SCIP was developed by Achterberg et al. [4]. It is conceived as a framework into which the majority of the algorithms that are needed to solve a MIP must be included as external plugins. The current distribution of SCIP already contains a bundle of MIP-solving plugins, e.g., presolvers, cut separators, and all primal heuristics described in this thesis. SCIP is implemented in C. For the computations presented in this thesis we used version SCIP 0.82b, which was compiled with `gcc 4.1.0`. Some of the primal heuristic plugins were implemented by Achterberg, while the others were implemented by the author of this thesis.

We used SCIP as an LP-based Branch-And-Cut MIP-solver, meaning that the problem instances are recursively subdivided into smaller subproblems, thereby generating a Branch-And-Bound-tree. At each node of this tree the LP-relaxation is solved to provide a lower bound of the objective value. In the root node the problem is strengthened by various types of cutting planes (see [58]). SCIP with default plugins is a state-of-the-art MIP-solver which is competitive (note Mittelmann's "Benchmarks for Optimization Soft-

ware" [48]) with other free solvers like CBC [27], GLPK [33], MINTO [49], and SYMPHONY [50] and also with commercial solvers like CPLEX [41], and XPRESS [22].

Except for the Objective Feasibility Pump from Section 3.1.2 where we gained access to the source code of the authors who described the original version, all implementations used in this thesis were made as SCIP plugins. CPLEX 10.0 [41] was always used as underlying LP-solver for SCIP.

2.2 Our Test Set

The computations of the Sections 2.2, 3.1, 3.2.2, and the first part of Section 3.2.1 were made on a 2.20 GHz AMD Opteron with 1024 KB cache and 32 GB RAM. The computations of the second part of Section 3.2.1, Sections 3.3 and 4.5 and of Chapter 5 were made on a 3.80 GHz Intel Pentium 4 with 1024 KB cache and 2 GB RAM.

We decided to use a test set consisting of 129 instances taken from:

- the MIPLIB 3.0 [16],
- the MIPLIB 2003 [6], and
- the MIP collection of Mittelmann [48].

Our test set contains all instances of these three collections except for the following: `gen`, `mod010`, `p0033`, `vpm1`, `manna81`, `neos4`, `neos8`, for which the optimum of the LP-relaxation using SCIP default settings is already integer feasible, `momentum3`, `stp3d`, whose root node LPs could not be solved by SCIP within a time limit of half an hour, and `markshare1_1`, `harp2`, which caused numerical troubles when running SCIP with default settings.

We subdivided the remaining 129 instances into two groups:

- the *easy test set*, all 97 instances which could be solved to optimality by SCIP with default settings on the 2.20 GHz AMD Opteron within a time limit of one hour, and
- the *hard test set*, all 32 instances which could not be solved to optimality by SCIP with default settings on the 2.20 GHz AMD Opteron within a time limit of one hour.

Tables 2.1 and 2.2 show the results of running SCIP with default settings and a time limit of one hour on all instances. They are already subdivided into both sets, and listed alphabetically. In Table 2.2, all instances of which the optimal objective is unknown are written in italics as well as their best known objective value.

The first column shows the name of the instance, the second one shows the type (for definitions see Section 1.1), the third one the number of constraints, and the fourth one the number of variables. Note that the data of Column 2 to 4 refers to the problem after SCIP presolving.

In Table 2.1, Column 5 shows the objective value of an optimal solution, Column 6 and 7 show the number of nodes and the time that SCIP, with default settings, needs to solve the instance to optimality.

Note that in contrast to the seven instances which were excluded, the optimal solutions of instances in Table 2.1, which were solved at the root node could only be found via heuristics.

In Table 2.2, Column 5 and 6 show the dual and primal bounds achieved by SCIP within the time limit of one hour. Column 7 shows the objective value of an optimal or best known solution of the MIP, Column 8 and 9 show the primal gap $\gamma_P(\tilde{x})$ and the primal-dual gap $\gamma_{PD}(\tilde{x})$ of the incumbent solution $\tilde{x}$ from Column 6. Column 10 gives the number of nodes SCIP processed until the time limit was reached.

For the remainder of this thesis, the tables with the detailed results of the test runs are always to be looked up in Appendix B. A survey of the main results is always given in the according section.

Due to technical reasons, the SCIP parameter "maxrestarts" was set to 0 for all tests in this thesis which process only the root node, since otherwise some heuristics would not have been applied. For all tests on the hard test set which were performed on the 3.80 GHz Intel Pentium 4 a memory limit of 1.5 GB was set.

Name	Type	Constraints	Variables	Primal Bound	Nodes	Time
10teams	BP	210	1600	924	1795	61.3
30:70:4_5:0_5:100	BP	12035	10768	9	183	485.9
30:70:4_5:0_95:98	BP	12459	10978	12	130	354.3
30:70:4_5:0_95:100	BP	12525	10975	3	121	457.0
acc-0	BP	1737	1620	0	1	13.2
acc-1	BP	2286	1620	0	1	29.6
acc-2	BP	2520	1620	0	1	33.0
acc-3	BP	3246	1570	0	78	203.9
acc-4	BP	3280	1570	0	1600	1230.5
acc-5	BP	2534	1017	0	4087	1041.0
acc-6	BP	2540	1018	0	274	150.8
aflow30a	MBP	479	842	1158	8751	60.6
air03	BP	81	10639	340160	2	38.4
air04	BP	601	7370	56137	269	236.2
air05	BP	343	6120	26374	215	94.8
bc1	MBP	1620	1002	3.33836255	16046	820.7
bell3a	MIP	97	110	878430.316	47669	44.8
bell5	MIP	75	94	8966406.49	1231	0.8
bienst1	MBP	576	505	46.75	9600	47.5
bienst2	MBP	576	505	54.6	94824	553.3
blend2	MIP	169	319	7.598985	869	3.1
cap6000	BP	1342	4109	-2451377	2938	63.9
dano3_3	MBP	3187	13873	576.344633	19	320.9
dano3_4	MBP	3187	13873	576.435225	41	277.3
dano3_5	MBP	3187	13873	576.924916	193	588.3
disctom	BP	394	9991	-5000	1	113.6
dcmulti	MBP	269	545	188182	151	5.1
dsbmip	MBP	1012	1662	-305.198175	1	0.5
egout	MBP	40	52	568.1007	2	0.0
eilD76	BP	75	1823	885.411847	1325	101.3
enigma	BP	21	100	0	731	0.5
fast0507	BP	474	62999	174	1747	2907.9
fiber	BP	279	930	405935.18	31	2.2
fixnet6	MBP	477	877	3983	13	1.5
flugpl	MIP	15	15	1201500	474	0.3
gesa2-o	MIP	1248	1224	25779856.4	1402	13.0
gesa2	MIP	1392	1224	25779856.4	153	6.4
gesa3	MIP	1344	1128	27991042.6	28	3.5
gesa3_o	MIP	1200	1128	27991042.6	647	10.9
gt2	IP	28	173	21166	60	0.1
irp	BP	39	19941	12159.4928	480	151.3
khb05250	MBP	100	1299	106940226	4	0.6
l152lav	BP	97	1989	4722	63	8.6
lseu	BP	27	88	1120	304	0.3
mas74	MBP	13	150	11801.1857	4578439	1241.0
mas76	MBP	12	150	40005.0541	347300	89.9
mas284	MBP	68	150	91405.7237	17213	29.7
misc03	BP	95	138	3360	38	1.4
misc06	MBP	630	1352	12850.8607	16	0.5
misc07	BP	223	232	2810	34475	44.3
mitre	BP	1115	3662	115155	27	93.6
mod008	BP	6	319	307	212	1.0
mod011	MBP	2215	6764	-54558535	2159	172.7
modglob	MBP	287	387	20740508.1	6842	11.2
mzzv11	IP	8633	8878	-21718	3003	1130.5

continue next page

Name	Type	Constraints	Variables	Primal Bound	Nodes	Time
mzzv42z	IP	9604	10390	-20540	2095	563.0
neos1	BP	1738	1728	19	1	5.0
neos2	MBP	813	1525	454.864697	71178	268.1
neos3	MBP	1190	2245	368.842751	655725	3219.9
neos5	MBP	33434	18985	-4.86034408e+10	3	344.9
neos6	MBP	868	8563	83	5177	549.5
neos7	MIP	1987	1538	721934	9061	87.4
neos10	IP	14621	793	-1135	5	306.6
neos11	MBP	2566	1130	9	21385	1551.0
neos13	MBP	14698	1827	-95.4748066	11191	1077.7
neos21	BP	1081	599	7	2107	39.5
neos22	MBP	4300	2786	779715	31583	410.4
neos632659	MBP	180	300	-94	1405657	601.1
noswot	MIP	171	120	-41	9209331	2569.5
nug08	BP	912	1632	214	5	103.8
nw04	BP	34	76309	16862	6	230.6
p0201	BP	107	195	7615	61	1.9
p0282	BP	253	189	258411	35	0.5
p0548	BP	288	445	8691	43	0.8
p2756	BP	1866	2635	3124	71	6.0
pk1	MBP	45	86	11	330702	145.2
pp08a	MBP	136	240	7350	1761	3.7
pp08aCUTS	MBP	246	240	7350	355	2.6
prod1	MBP	171	213	-56	65869	53.3
qap10	BP	1820	4150	340	5	519.1
qiu	MBP	1192	840	-132.873137	9947	156.2
qnet1	IP	364	1417	16029.6927	112	5.6
qnet1_o	IP	332	1417	16029.6927	40	4.1
ran8x32	MBP	296	512	5247	14497	30.7
ran10x26	MBP	296	520	4270	24747	54.7
ran12x21	MBP	285	504	3664	147324	243.2
ran13x13	MBP	195	338	3252	71961	75.3
rentacar	MBP	1350	3133	30356761	6	4.6
rgn	MBP	24	175	82.1999991	2125	0.8
rout	MIP	290	555	1077.56	80420	134.7
set1ch	MBP	446	666	54537.75	22	1.7
seymour1	MBP	4827	1255	410.763701	4680	985.6
stein27	BP	118	27	18	4063	3.4
stein45	BP	331	45	30	53074	51.1
swath1	MBP	482	6320	379.071296	11225	277.1
swath2	MBP	482	6320	385.199693	63456	894.0
vpm2	MBP	128	181	13.75	9815	7.3
Total (97)					17509205	28945.4
Geom. Mean					686	41.9

Table 2.1. Easy test set: all instances were solved to optimality within one hour

Name	Type	Conss	Vars	Dual Bound	Primal Bound	Optimum / *Best known*	P-D Gap	P Gap	Nodes	Time
a1c1s1	MBP	3240	3493	8625.9465	12505.9439	*11596.1364*	45.0	7.8	23175	
aflow40b	MBP	1442	2728	1145.17278	1215	1168	6.1	4.0	247369	
arki001	MIP	767	960	7580525.42	7584819.57	7580928.38	0.1	0.1	618932	
atlanta-ip	MIP	19412	17261	82.9973629	–	*95.0095497*	–	–	375	
binkar10_1	MBP	826	1444	6735.06954	6743.24002	6742.20002	0.1	0.0	677473	
dano3mip	MBP	3187	13873	576.909847	743.464286	*697.857143*	28.9	6.5	802	
danoint	MBP	664	521	63.671765	66.25	65.67	4.0	0.9	236660	
ds	BP	625	67076	58.3411586	399.275161	*283.4425*	584.4	40.9	433	
glass4	MBP	392	317	800087789	1.7800148e+09	1.2000126e+09	122.5	48.3	3967642	
liu	MBP	2178	1154	560	1926	*1172*	243.9	64.3	488269	
markshare1	BP	6	50	0	5	1	–	400.0	41678324	
markshare2	BP	7	60	0	11	1	–	1000.0	35738239	
mkc	MBP	3212	5198	-565.03	-546.412	-563.846	3.3	3.1	251796	
mkc1	MBP	3274	5314	-607.21	-606.71	-607.207	0.1	0.1	450353	
momentum1	MBP	14859	2785	96411.7673	–	109143.493	–	–	2091	
momentum2	MIP	15440	2785	10710.8783	–	12314.2196	–	–	1895	
msc98-ip	MIP	15159	12790	19702877	–	*23271298*	–	–	821	
neos616206	MBP	534	440	889.561042	937.866667	937.6	5.4	0.0	859053	
net12	MBP	12783	12531	137.910831	–	214	–	–	1560	
nsrand-ipx	MBP	536	6601	50651.6299	55040	51200	8.7	7.5	58256	
opt1217	MBP	64	759	-19.2240572	-16.000007	-16	16.8	0.0	4251254	
protfold	BP	2112	1835	-36.3333333	–	*-30*	–	–	862	
rd-rplusc-21	MBP	17905	484	100	179794.684	*171182*	Large	5.0	38747	
roll3000	MIP	1478	881	12722.6185	12936	12929	1.7	0.1	194565	
seymour	BP	4827	1255	413.675105	425	423	2.7	0.5	5469	
sp97ar	BP	1638	14100	655957310	700910273	*664565104*	6.9	5.5	7107	
swath	MBP	482	6320	397.576488	504.002447	*477.341*	26.8	5.6	190274	
swath3	MBP	482	6320	371.816774	397.761344	397.761344	7.0	0.0	297024	
t1717	BP	551	67716	135535.1	208442	*193221*	53.8	7.9	225	
timtab1	MIP	168	201	642081.421	911718	764772	42.0	19.2	3892397	
timtab2	MIP	291	341	567341.879	1548284	*1216284*	172.9	27.3	1728071	
tr12-30	MBP	730	1048	130485.754	130683	130596	0.2	0.1	374439	

Table 2.2. Hard test set: No instance was solved to optimality within one hour

Chapter 3

Start Heuristics

Pruning suboptimal branches is an important part of Branch-And-Bound-algorithms. This helps keeping the Branch-And-Bound-tree small as well as the number of computing steps and hence, the solving time and the required memory. A branch can be pruned if the objective value of its LP-optimum is not smaller than the one of the incumbent solution.

Therefore, it is of high importance to discover feasible solutions as early as possible in order to achieve a good performance of the Branch-And-Bound-process. Start heuristics aim at finding a feasible solution early in the Branch-And-Bound-process.

In practice, most of them are already applied at the root node, some can even be applied while or before the root-LP is solved to optimality.

3.1 Diving Heuristics

SCIP, like most state-of-the-art MIP solvers, uses the techniques of LP-based Branch-And-Bound. The branching process hereby focuses on two different aims: on the one hand fractional variables should be driven towards integrality in order to find feasible solutions, on the other hand the dual bound should be raised in order to prove the optimality (see [5]). For the process of finding feasible solutions, it would be desirable to completely concentrate on bounding variables such that the number of fractional variables decreases while staying LP-feasible without caring for the dual.

This is the motivation of the following heuristic idea: one bounds or fixes variables of a fractional LP-solution to promising values and resolves the LP iteratively. Thereby, the exploration of a possible root-leaf path of the Branch-And-Bound-tree is simulated. The behavior that these heuristics "quickly go down" the Branch-And-Bound-tree gave the reason to name this class *diving heuristics*.

Another approach is to change the objective function such that the corresponding LP-solution is driven towards integer feasibility. The heuristics presented in Section 3.1.1 are of the first type, whereas the one presented in

Section 3.1.2 is of the second type.

The implementation of the heuristics of Section 3.1.1 has been done by Achterberg, the integration into the original source code and the SCIP-implementation of the heuristic described in Section 3.1.2 has been done by the author of this thesis.

3.1.1 Some Simple Divers

Most of the diving heuristics of SCIP bound a single fractional variable in each iteration, like most MIP branching rules do (see [5]).

Definition 3.1 *Let $\bar{x}$ be an LP-feasible solution of a given MIP, l,u the lower and upper bound vectors. Let $j \in I$. We call the process of replacing l_j by $\lceil \bar{x}_j \rceil$* bounding up *the variable x_j, replacing l_j by $\lfloor \bar{x}_j \rfloor$* bounding down *the variable x_j.*

Note that for fractional binary variables bounding and fixing are equivalent actions, whereas for general integers bounding in general is a weaker action than fixing the variable.

The Idea

Bounding variables – especially binary ones – helps to simplify the LP, and additionally, the solution of the modified LP has less fractional variables. The LP with the changed bound is resolved in order to regain linear feasibility. The resolving is normally done with the dual simplex algorithm, since the dual LP-solution stays feasible if changing a bound. The dual simplex algorithm usually needs only few LP iterations to reoptimize the modified problem. If this process of bounding and resolving is iterated, it simulates a depth-first-search of a promising root-leaf path of the Branch-And-Bound-tree and might lead to a feasible solution.

The diving process terminates as soon as one of the following conditions holds:

- the LP gets infeasible,
- the optimum of the LP is worse than the incumbent solution of the MIP in terms of the objective function,
- we reach some iteration limit or some limit on the LP solving process.

If one of the first two conditions holds, the diving process cannot produce a better solution and should hence be aborted. The limits mentioned in the third condition are made to keep control of the time the heuristic needs. A limit on the LP solving process could be a time limit or a limit on the LP-iterations, say $nlp \in \mathbb{N}$.

The principial process can be seen in Algorithm 1.

Input: $\bar{x}$ LP-feasible point

1 $\bar{I} \leftarrow$ index set of all fractional variables of $\bar{x}$;

2 $c_{\max} \leftarrow \begin{cases} \infty & \text{if no feasible solution found so far} \\ c^T\tilde{x} & \text{else, with } \tilde{x} \text{ being the incumbent solution} \end{cases}$;

3 $nlp \leftarrow 0, i \leftarrow 0, nfr \leftarrow |\bar{I}|$;

4 **while** $\bar{I} \neq \emptyset$ **and** $c^T\bar{x} < c_{\max}$ **and** *(* $(nlp < nlp_{\max}$ **and** $i < i_{\max})$ **or** $i < i_{\min}$ **or** $|\bar{I}| \leq nfr - \frac{i}{2})$ **do**

5 $\quad i \leftarrow i + 1$;

6 $\quad$ Select some $j \in \bar{I}$;

7 $\quad$ Bound x_j: either $l_j \leftarrow \lceil \bar{x}_j \rceil$ or $u_j \leftarrow \lfloor \bar{x}_j \rfloor$;

8 $\quad$ **if** *modified LP has a feasible solution* **then**

9 $\quad\quad \bar{x} \leftarrow$ optimal solution of modified LP ;

10 $\quad$ **else**

11 $\quad\quad$ **stop!**

12 $\quad \bar{I} \leftarrow$ index set of all fractional variables of $\bar{x}$, $nfr \leftarrow |\bar{I}|$;

13 $\quad$ **if** *all* x_j *with* $j \in \bar{I}$ *are trivially roundable* **then**

14 $\quad\quad$ Round $\bar{x}$; `/* yields feasible solution */`

15 $\quad nlp \leftarrow nlp +$ number of LP-iterations needed to solve LP in Step 9;

Algorithm 1: A General Diving heuristic

The last part of the loop-condition (Step 4) is introduced, because there are two cases for which we want to continue diving, even if we exceeded one of the iteration limits. The first one is that we just have started our dive ($i < i_{\min}$). The second one is that we are in a promising dive, meaning that the number of fractional variables decreased rapidly enough ($|\bar{I}| \leq nfr - \frac{i}{2}$, where $nfr \in \mathbb{N}$ denotes the number of fractional variables at the beginning of the diving).

Implementation Details

There are a couple of ideas how to select the variable which should be bounded. Some of them are presented here, each one is implemented in the named SCIP diving heuristic. Let $\bar{x}$ be the optimum of the LP relaxation of the current Branch-And-Bound node. The variable x_j which shall be bounded is always chosen among all integer variables x_j for which $\bar{x}_j$ is fractional.

- *Fractional Diving* : A variable x_j with lowest fractionality $f(\bar{x}_j)$ is bounded to the nearest integer.
- *Coefficient Diving* : A variable x_j with minimal positive number of locks is bounded to a direction where this minimum is achieved. If there is more than one variable where the minimal number of locks is reached, one chooses a variable with minimal fractionality $f(\bar{x}_j)$. The

number of locks is the number of constraints which could possibly be violated if rounding a variable to a certain direction (see Definition 1.5).

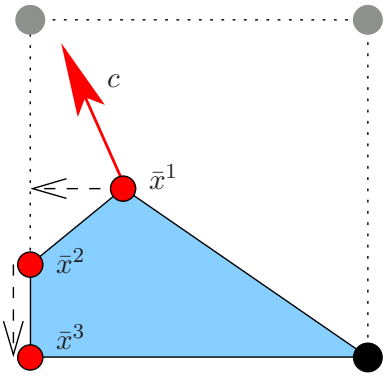
- *Linesearch Diving* : Let $\mathring{x}$ be the optimum of the LP-relaxation of the root node and let s be the line connecting $\mathring{x}$ with $\bar{x}$. A fractional variable x_j for which s first intersects the coordinate-hyperplane $x_j = k$, $k \in \{\lfloor \bar{x}_j \rfloor, \lceil \bar{x}_j \rceil\}$ is bounded to the value k.

- *Guided Diving* : Like Fractional Diving, Guided Diving tends to bound variables with low fractionality. Let $\tilde{x}$ be the incumbent solution of the MIP. Fractional Diving selects a fractional variable with lowest value $|\tilde{x}_j - \bar{x}_j|$ and bounds it to $\tilde{x}_j$. Note that due to the fact that Guided Diving needs a feasible solution of the MIP, it is actually an improvement heuristic, not a start heuristic.

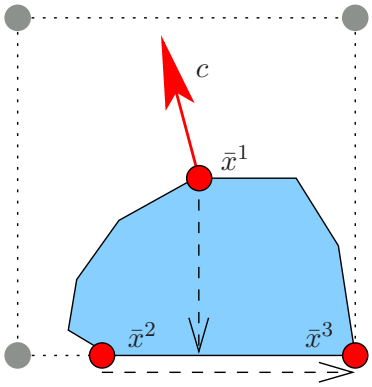
- *Pseudocost Diving* : The *pseudocosts* of a variable are a measure for the average change of the objective function per unit change of the variable, taken over all Branch-And-Bound-nodes, where this variable has been chosen to branch on. An exact definition and discussion of pseudocosts and their applications is given in [5, 46]. The Pseudocost Diving heuristic integrated into SCIP combines some ideas of other diving heuristics. First of all, one makes the decision whether a variable x_j would be bounded up or down, if it was chosen for bounding.

  If $|\bar{x}_j - \mathring{x}_j| > 0.4$, x_j would be bounded to the direction, the variable has developed, like it is done in Linesearch Diving: if $\bar{x}_j > \mathring{x}_j$ the variable would be bounded up, otherwise it would be bounded down. If $f(x_j) < 0.3$, x_j would be bounded to the nearest integer, like it is done in Fractional Diving. Otherwise the variable would be bounded to the direction with the smaller pseudocosts. Afterwards, a score is calculated for each variable: the fractionality $f(\bar{x})$ times the quotient between the pseudocosts of the variable to the direction where it should not be bounded, and the pseudocosts of the variable to the direction where it should be bounded. A low score indicates a direction with high decrease in the objective function, a high score a direction with small decrease. A variable with highest score will be chosen for bounding.

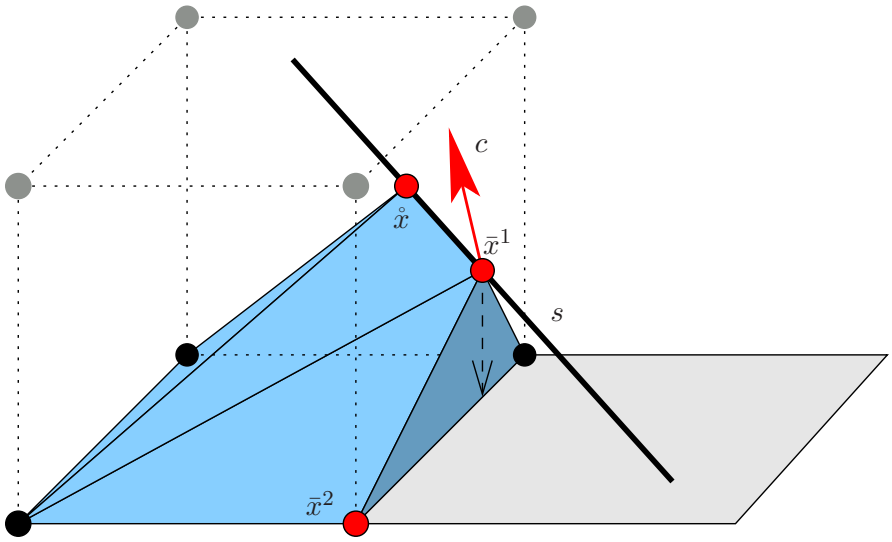
- *Vectorlength Diving* : The idea of this diving heuristic is taken from regarding set partitioning constraints (see [18]). 27 out of 60 MIPLIB 2003 instances (see [6]) contain such constraints. If one of the variables of a set partitioning constraint is fractional, there is at least one other fractional variable in this constraint. Hence, fixing a variable and resolving the LP influences the other variables and hopefully drives them to integrality, too. Especially fixing a variable to 1 fixes all others to 0 in the solution of the resolved LP. Vectorlength Diving bounds a variable, for which the quotient between $([\bar{x}_j] - \bar{x}_j) * c_j$ and $|\{a_{ij} \mid a_{ij} \neq 0\}|$ is minimal. In this case, $[\cdot]$ denotes rounding to the direction where



Fractional Diving



Coefficient Diving



Linesearch Diving

Figure 3.1. The fundamental procedure of some selected diving heuristics

the objective function potentially gets worse, hence up, if $c_j \geq 0$ and down otherwise. This ratio provides the change in the objective function per row which is effected by the rounding. Naturally, little loss in the objective and a high number of "repaired" rows is desirable.

Figure 3.1 illustrates the fundamental procedure of some diving heuristics. $\bar{x}^i$ denotes the optimum of the LP-relaxation in the i-th step of the diving process, $\mathring{x}$ the optimum of the LP-relaxation of the Branch-And-Bound root node. A dashed arrow shows which variable is bounded and to which direction. The objective function is only sketched at the first LP-solution.

However, there are some properties which the variables chosen as bounding candidate should not have. The first one is that the chosen variable should not be trivially roundable (see Definition 1.5), since in Step 14 fixing them will be tried either way, as soon as there are only trivially roundable, fractional variables left. Hence, trivially roundable variables should only be chosen if there are only such variables left. If a trivially roundable variable is chosen to be bounded, it should always be bounded to the direction where rounding is not trivial. Otherwise, the developing of $\bar{x}$ would be similar to the one of a rounding heuristic but with much more computational effort.

Secondly, binary variables should be favored over general integers, since bounding binary variables in the described way is equivalent to fixing them. On the one hand this results in easier LPs, on the other hand one avoids the risk of bounding the same integer variable again and again. Another reason for preferring binary variables is that in many real world motivated problems the binary variables present the critical decisions and thereby induce the main structure of the problem.

Computational Results

We analyzed the performance of Coefficient Diving, Fractional Diving, and Vectorlength Diving applied to the optimum of the root node's LP-relaxation. These three diving heuristics only require some starting point and the general structure of the MIP itself in order to get started. In contrast to that the other three diving heuristics described in this chapter rely on some information which is got during the MIP-solving process: Guided Diving needs an incumbent solution, Linesearch Diving the LP-optimum of a deeper node of the Branch-And-Bound-tree, and Pseudocost Diving needs pseudocosts of at least some variables.

The results are to be seen in Table B.1. Table 3.1 summarizes the main results of Table B.1.

Coefficient Diving found at least one feasible solution for 71 out of the 129 instances, Fractional Diving succeeded 68 times, and Vectorlength Diving was able to find a solution for 93 instances. There were 54 instances for which all three diving heuristics found a feasible solution. Coefficient Diving found a strictly better solution than the other two 10 times, Fractional Diving was superior 7 times, and Vectorlength Diving defeated the others 29 times.

Criterion	Coefficient Diving	Fractional Diving	Vectorlength Diving
Solution Found	71	68	**93**
Best Solution	10	7	**29**
Only Solution	1	2	**12**
Optimal Solution	8	4	**14**
Total Time	834.6	**700.2**	2370.7
Time Geometric Mean	2.3	**2.1**	3.9

Table 3.1. Summarized results of applying different diving heuristics to the optimum of the LP-relaxation

There was one instance for which Coefficient Diving was the only diving heuristic which found a feasible solution, two instances where this holds for Fractional Diving, and 12 for Vectorlength Diving. Coefficient Diving was able to find an optimal solution for 8 instances, 4 times Fractional Diving found an optimum. However, Vectorlength Diving found optimal solutions for 14 instances. Note that due to rounding the primal gap, a value of 0.0% in the table does not necessarily mean that the solution is optimal. For example, all three heuristics found solutions with a pairwise different objective value and a primal gap of nearly 0 % for instance `mitre`, but only the solution found by Coefficient Diving is optimal.

Hence, Vectorlength Diving is superior to the other two variants in terms of the number and the quality of the solutions it found, but it looses in terms of the computation time. Coefficient Diving needed 834.6 seconds or 2.3 seconds in the geometric mean, Fractional Diving 700.2 or 2.1 seconds, but Vectorlength Diving took 2370.7 seconds overall, and 3.9 in the geometric mean.

All three divers proved to be reasonable start heuristics as they all found solutions for more than half of the problems, and often these solutions were of good quality. Due to the necessity of resolving LPs they consume much more time than, for example, all rounding heuristics from Chapter 3.2.2 do and should hence be called less frequently during the Branch-And-Bound-process.

3.1.2 Feasibility Pump

The *Feasibility Pump,* as proposed by Fischetti, Glover, and Lodi [24], is a primal heuristic for MBPs. It was restated for MIPs with general integer variables by Bertacco, Fischetti, and Lodi [14]. Achterberg, and Berthold [3] described a modified version which takes the objective function of the MIP into account and is therefore called *Objective Feasibility Pump.*

The Idea

The fundamental idea of the Feasibility Pump is to construct two sequences of points which hopefully converge to a feasible solution of the given MIP. One sequence consists of LP-feasible, but possibly integer infeasible points, the other one of integer feasible, but possibly LP-infeasible points. These

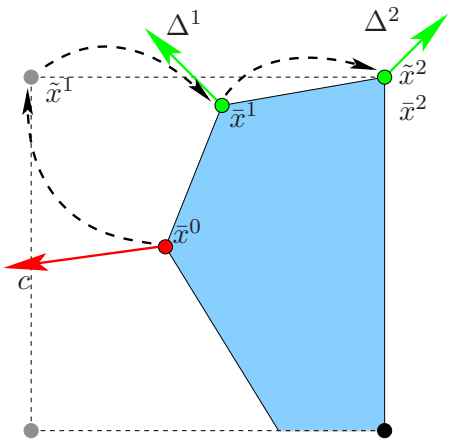


Figure 3.2. Fundamental procedure of the Feasibility Pump: two sequences of LP-feasible points $\bar{x}^t$ and integer feasible points $\tilde{x}^t$ are build up until they converge

are produced by alternately rounding an LP-feasible point $\bar{x} \in P(A, b, l, u)$ to an integer feasible point $\tilde{x}$ and finding a point in $P(A, b, l, u)$ closest to $\tilde{x}$ (w. r. t. the Manhattan distance), which is then used as the new $\bar{x}$ in the next iteration step. This procedure is illustrated in Figure 3.2.

Definition 3.2 *Let $S \subseteq N$. The rounding of a vector $x \in \mathbb{R}^n$ with respect to S is defined by components:*

$$[x]_j^S := \begin{cases} \lfloor x_j + 0.5 \rfloor & \text{if } j \in S \\ x_j & \text{if } j \notin S. \end{cases}$$

Similarly, the L_1-distance of two vectors $x, y \in \mathbb{R}^n$ with respect to S is defined as:

$$\Delta^S(x, y) := \sum_{j \in S} |x_j - y_j|$$

and the fractionality of a vector $x \in \mathbb{R}^n$ with respect to S is defined as:

$$f^S(x) := \sum_{j \in S} f(x_j) \quad \text{with} \quad f(x_j) := |x_j - \lfloor x_j + 0.5 \rfloor| .$$

For the rest of this section, the terms rounding, distance, and fractionality will always refer to these definitions unless otherwise noted.

The Feasibility Pump as described in [14] proceeds as follows: first of all the LP-relaxation is solved, taking an optimal LP-solution as starting point $\bar{x}$. Then, $\bar{x}$ is rounded to a vector $\tilde{x} = [x]^S$ (with $S = B$ or $S = I$). If $\tilde{x}$ is a feasible solution, the procedure stops. If not, an additional LP is solved in order to find a new LP-feasible point $\bar{x}$ in the polyhedron $P(A, b, l, u)$ which is a closest to $\tilde{x}$, with respect to $\Delta^S(x, \tilde{x})$ as the distance function. The procedure is iterated using this point as new solution $\bar{x} \in P$. The algorithm

terminates, if a feasible solution is found or if a predefined iteration limit is reached.

Algorithm 2 gives us an outline of the fundamental proceeding of the Feasibility Pump.

Input: $\bar{x}$ LP-feasible point
1 **repeat**
2 | Get an integer feasible point $\tilde{x}$ by rounding $\tilde{x} \leftarrow [\bar{x}]^S$;
3 | Get a new LP-feasible point $\bar{x}$ by solving
| $\bar{x} \leftarrow \operatorname{argmin}\{\Delta^S(x, \tilde{x}) \mid x \in P(A, b, l, u)\}$;
4 **until** $\bar{x} = \tilde{x}$ **or** *some limit reached* ;

Algorithm 2: Outline of the Feasibility Pump

Implementation Details

In order to determine a point

$$\bar{x} := \operatorname{argmin}\{\Delta^S(x, \tilde{x}) \mid x \in P(A, b, l, u)\} \tag{3.1}$$

which is closest to $\tilde{x}$ in $P(A, b, l, u)$, one solves the LP

$$\begin{aligned} \min \quad & \sum_{j \in S: \tilde{x}_j = l_j} (x_j - l_j) + \sum_{j \in S: \tilde{x}_j = u_j} (u_j - x_j) + \sum_{j \in S: l_j < \tilde{x}_j < u_j} d_j \\ \text{such that} \quad & Ax \leq b \\ & d \geq x - \tilde{x} \\ & d \geq \tilde{x} - x \\ & l \leq x \leq u. \end{aligned} \tag{3.2}$$

The variables d_j are introduced to model the nonlinear function $|x_j - \tilde{x}_j|$ for general integer variables x_j that are not equal to one of their bounds in the rounded solution $\tilde{x}$. Since for an optimal solution $\bar{x}$ of (3.2) every d_j strictly fulfills one of its two inequalities with equality, (3.2) is a correct linearization of (3.1). Note that for $S = B$ the variables d_j are not needed.

In many practical applications the binary variables represent the most important decisions that define most of the structure. Therefore, Bertacco, Fischetti, and Lodi [14] suggested to subdivide the Feasibility Pump into three stages.

During the first stage, the algorithm tries to find an LP-feasible solution that is integral in the binary variables by setting $S = B$, hence relaxing the integrality constraints for all general integer variables. Stage 1 is interrupted as soon as such a solution is found or an iteration limit is reached. This could be a limit on the absolute number of iterations or on the number of consecutive *stallings*. In the following, we will call an iteration a stalling, if the fractionality measure $f^S(\bar{x})$ could not be decreased by at least a factor of

$p \in [0,1]$ (e.g., $p = 0.9$ for 10 % reduction). By setting a limit on the number of consecutive stallings, one wants to avoid wasting time on performing a huge number of iterations with only slight advances in the main goal of reducing the fractionality of the LP-feasible points, as it can be seen in [14].

The motivation for dividing the algorithm into stages is that in a solution from Stage 1 hopefully only a few general integer variables have a fractional value and require the introduction of the auxiliary variables d_j in the second stage of the algorithm in which $S = I$ is chosen. It starts from a rounded point $\tilde{x}$ of Stage 1 with minimal distance from $P(A, b, l, u)$. Thus, we try to benefit from the insights we won in Stage 1. Stage 2 terminates as soon as a feasible solution of the given MIP is found or some iteration limit as above is reached. Note that the algorithm will skip Stage 2 for MBPs.

Stage 3 consists of a Large Neighborhood Search (see Chapter 4) which examines the neighborhood of a point $\tilde{x}$ from Stage 2 with minimal distance to $P(A, b, l, u)$. It is the hope that such a point is nearly feasible and that it is likely to find feasible solutions in the vicinity of this point. Therefore, one tries to solve the MIP

$$\begin{aligned} \min \quad & \Delta^I(x, \tilde{x}) \\ \text{such that} \quad & Ax \leq b \\ & l \leq x \leq u \\ & d \geq x - \tilde{x} \\ & d \geq \tilde{x} - x \\ & x_j \in \mathbb{Z} \text{ for all } j \in I \end{aligned} \tag{3.3}$$

by a MIP-solver which stops after the first feasible solution was found.

The Objective Feasibility Pump

Computational results show that the Feasibility Pump is quite successful in finding feasible solutions of a given MIP [24, 14, 3]. Unfortunately, the quality of the solutions in terms of the primal gap is sometimes quite poor. This can be explained from the observation that the original objective function c of the given MIP is only regarded once, namely, when the first LP-feasible point $\bar{x}$ is determined as an optimal solution of the LP-relaxation. Achterberg and Berthold [3] provide a modified version of the Feasibility Pump, called the Objective Feasibility Pump, which takes the objective function much more into account than the original Feasibility Pump does.

The Objective Feasibility Pump does not totally disregard the objective function c after the first iteration, but decreases its influence in every iteration step. Therefore, it tends to go more towards feasibility and less towards optimality the longer it runs. If no proper original objective function exists, i.e., $c = 0$, one will just use the original version of the Feasibility Pump. To avoid some technical difficulties, we assume that $c \neq 0$ for the remainder of this chapter.

The Objective Feasibility Pump is gained from the original Feasibility Pump by replacing the distance function $\Delta^S(\cdot, \tilde{x})$ by a convex combination of $\Delta^S(\cdot, \tilde{x})$ and c, the original objective function vector.

Definition 3.3 *Let* $\tilde{x} \in \mathbb{R}^n$, $S \subseteq N$, $c \in \mathbb{R}^n \backslash \{0\}$, *and* $\alpha \in [0, 1]$.

$$\Delta^S_\alpha(x, \tilde{x}) := (1 - \alpha)\Delta^S(x, \tilde{x}) + \alpha \frac{\sqrt{|S|}}{\|c\|} c^T x \tag{3.4}$$

Here, $\|\cdot\|$ denotes the Euclidean norm of a vector. Note that $\sqrt{|S|}$ is the Euclidean norm of the objective function vector of (3.2).

The Objective Feasibility Pump computes the LP-feasible points $\bar{x}$ using Δ^S_α instead of Δ^S in (3.1). Therefore, the objective function of (3.2) is appropriately modified, namely by replacing it with a convex combination of itself and c. In each iteration step α is geometrically decreased by a fixed factor $\varphi \in [0, 1)$, i.e., $\alpha_{t+1} = \varphi\, \alpha_t$ and $\alpha_0 \in [0, 1]$. Note that if we choose $\alpha_0 = 0$ we will get the original version of the heuristic.

Dealing with Cycles

One main problem arises in both variants of the Feasibility Pump: what to do if one gets back to an integer feasible point $\tilde{x}$ which was already visited in a prior iteration? The urgency to deal with this manner differs in both versions. For the original Feasibility Pump, turning back to a point that has been already visited means getting caught in a *cycle*. The Feasibility Pump would find exactly the same closest point $\bar{x}$, would round it to the same integer feasible point like in the prior iteration and hence, get the whole sequence of points over and over again. Therefore, Fischetti, Glover and Lodi [24] suggest to perform a *restart* operation.

A restart conducts a random perturbation which shifts some of the variables' values of the last LP-feasible solution $\bar{x}$ randomly up or down, instead of rounding them as usual. If there is a cycle of the length 1, which means that you immediately turn back to the rounded point $\tilde{x}$ of the preceding iteration, you will skip the random choice, but just round a certain number of the say T most fractional variables to the other side.

The Objective Feasibility Pump uses different parameters α_t and $\alpha_{t'}$ in different iteration steps t and t'. Because of the different objective functions Δ_{α_t} and $\Delta_{\alpha_{t'}}$ it is possible to get to a new LP-feasible point $\bar{x}$ and not to run into a cycle even if $\tilde{x}$ was already visited. The probability of this event clearly depends on how much the two functions Δ_{α_t} and $\Delta_{\alpha_{t'}}$ differ, which itself depends on how much α_t and $\alpha_{t'}$ differ. The Objective Feasibility Pump performs a restart at iteration t only if the point $\tilde{x}$ was already visited at iteration $t' < t$ with $\alpha_{t'} - \alpha_t \leq \delta_\alpha$, where $\delta_\alpha \in (0, 1]$ is a fixed parameter value.

Altogether this provides the start heuristic, which is depicted in Algorithm 3.

Input: $\bar{x}$ LP-feasible point

Parameters: $maxIter_{\mathrm{ST1}}$ maximum number of iterations in stage 1
$maxIter_{\mathrm{ST2}}$ maximum number of iterations in stage 2
$maxStalls_{\mathrm{ST1}}$ maximum number of stallings in stage 1
$maxStalls_{\mathrm{ST2}}$ maximum number of stallings in stage 2
T number of variables to change if 1-cycle occurs
p minimum improvement during *maxStalls* iterations
α_0 initial value for α
φ factor to reduce α in each iteration
δ_α equality tolerance for α

`Stage 1`

1 Initialize $\bar{x}^0 \leftarrow \operatorname{argmin}\{c^T x \mid x \in P(A,b,l,u)\}$, $S \leftarrow B$, $\tilde{x}^0 \leftarrow [\bar{x}^0]^S$, $maxIter \leftarrow maxIter_{\mathrm{ST1}}$, $maxStalls \leftarrow maxStalls_{\mathrm{ST1}}$, $t \leftarrow 0$;
2 Initialize list of visited points $L \leftarrow \emptyset$;
3 **loop**
4 **if** $\tilde{x}^t = \tilde{x}^{t-1}$ **then**
5 Round the T most fractional variables $\bar{x}^t_j$, $j \in S$, to the other side compared to $\tilde{x}^{t-1}_j$;
6 **while** $\exists\, (\tilde{x}^{t'}, \alpha_{t'}) \in L : \tilde{x}^{t'} = \tilde{x}^t \wedge \alpha_{t'} - \alpha_t \leq \delta_\alpha$ **do**
7 Perform a random perturbation on $\tilde{x}^t$;
8 **if** $\tilde{x}^t$ is feasible for (MIP) **then stop!**
9 $L \leftarrow L \cup \{(\tilde{x}^t, \alpha_t)\}$;
10 $t \leftarrow t+1$, $\alpha_t \leftarrow \varphi \cdot \alpha_{t-1}$;
11 **if** $t \geq maxIter$ **then goto** next stage;
12 Solve $\bar{x}^t \leftarrow \operatorname{argmin}\{\Delta^S_{\alpha_t}(x, \tilde{x}^{t-1}) \mid x \in P(A,b,l,u)\}$;
13 **if** $\bar{x}^t = \tilde{x}^{t-1}$ **then goto** next stage;
14 **if** $t \geq maxStalls \wedge f^S(\bar{x}^t) \geq p \cdot f^S(\bar{x}^{t-maxStalls})$ **then**
15 **goto** next stage;
16 $\tilde{x}^t \leftarrow [\bar{x}^t]^S$;

`Stage 2`

1 Initialize $\tilde{x}^0$ to be an integer point of Stage 1 with minimal $\Delta^B(x, \tilde{x}^i)$, $x \in P(A,b,l,u)$;
2 Set $t \leftarrow 0$, $S \leftarrow I$, $maxIter \leftarrow maxIter_{\mathrm{ST2}}$, $maxStalls \leftarrow maxStalls_{\mathrm{ST2}}$, $L \leftarrow \emptyset$;
3 Perform Steps 3 to 16 of Stage 1, but **if** Step 7 is performed more than 100 times **then goto** Stage 3, and **if** $\bar{x}^t = \tilde{x}^{t-1}$in Step 13 **then stop!**

`Stage 3`

1 Solve MIP (3.3) with $\tilde{x}$ being an integer point of Stage 2 with minimal $\Delta^I(x, \tilde{x}^i)$, $x \in P(A,b,l,u)$;
2 After the first feasible solution has been found → **stop!**

Algorithm 3: Objective Feasibility Pump

Computational Results

Bertacco, Fischetti, and Lodi sent us their original source code. Thus, we were able to directly integrate our ideas for the Objective Feasibility Pump into it. We thank them for making this possible.

Since we wanted to facilitate the comparison of both Feasibility Pump variants, we decided not to perform the tests on our usual test set. Instead of, we decided to use a huge test set, which besides others includes all instances on which Bertacco, Fischetti, and Lodi [14] tested their implementation. It consists of 121 instances taken from

- Miplib 2003 [6],
- the MIP collection of Mittelmann [48], and
- Danna, Rothberg, and LePape [21].

Like the authors of the original source code, we used Cplex 9.03 as underlying LP-solver. In all runs we used the parameter settings for the Feasibility Pump as suggested in [14]. The maximum numbers of total iterations for Stages 1 and 2 were set to $maxIter_{\mathrm{ST1}} = 10000$ and $maxIter_{\mathrm{ST2}} = 2000$. The maximum numbers of iterations without at least $p = 10\%$ improvement in a single iteration were set to $maxStalls_{\mathrm{ST1}} = 70$ and $maxStalls_{\mathrm{ST2}} = 600$.

The shifting described in Step 5 of algorithm 3 is applied on a random number of $T \in [10, 30]$ variables, whereby only variables x_j with current fractionality $f(\bar{x}_j) > 0.02$ are regarded as shifting candidates. For the unmodified Feasibility Pump we set $\alpha_0 = 0$ to deactivate all modifications. For the Objective Feasibility Pump we set $\alpha_0 = 1$, $\varphi = 0.9$, and $\delta_\alpha = 0.005$.

In contrast to the results presented in [3], we deactivated the third stage of the Feasibility Pump, since we were about to examine the diving process, which is only performed in the first two stages.

We applied the MIP preprocessing of Cplex before running the Feasibility Pump, since we aimed to avoid difficulties with the calculation of Δ_α^S. Such cases may appear if the objective function is highly degenerated, e.g., it consists of only one single artificial variable which is defined as a linear combination of several other variables.

Table B.2 shows the results of running the Objective Feasibility Pump with different settings for the parameter φ, namely $\varphi = 0$ (and $\delta_\alpha = 0$) in order to simulate the original Feasibility Pump and $\delta_\alpha = 0.8$, $\delta_\alpha = 0.9$, $\delta_\alpha = 0.95$, $\delta_\alpha = 0.99$, $\delta_\alpha = 0.999$. For the latter five, we set $\alpha_0 = 1$ and $\delta_\alpha = 0.005$. For reasons of simplification we will abbreviate the different versions with FP00, FP08, etc.

Table 3.2 summarizes the results we got from Table B.2.

Our first observation is that all versions, except for FP0999, are able to produce feasible solutions for a large number of instances. FP00 found a solution for 90 out of 121 instances, FP08 for 93, FP09 for 92, FP095 for 92, FP099 for 80, but FP0999 only for 25 instances.

Criterion	FP00	FP08	FP09	FP095	FP099	FP0999
Solution Found	90	**93**	92	92	80	25
Compared to FP00 (better:draw:worse)	–	68:36:17	68:35:18	**70:33:18**	64:35:22	21:35:65
Best Solution	12	23	27	**37**	30	12
Unique Best Solution	6	11	14	**25**	22	8
Only Solution	0	1	1	**2**	1	0
Time Geometric Mean	**4.7**	5.1	5.3	5.6	7.4	9.1
Primal Gap Geometric Mean (of 75)	106.7	47.0	43.1	**37.8**	39.8	–

Table 3.2. Summarized comparison of Feasibility Pump versions which differ in the factor $\varphi \in [0, 1)$ to reduce the convex combination parameter α

Every Objective Feasibility Pump version except FP0999 clearly dominates the original Feasibility Pump in terms of the quality of the solutions. FP08 produced a better solution for 68 instances, a worse for 17 instances with 36 draws. FP09 was superior in 68 cases, inferior in 18 cases, 35 times both variants got the same result. FP095 defeated the original Feasibility Pump in 70 cases, succumbed in 18 cases, 33 draws. FP099 still was better for 64 instances, worse for only 22 and 35 ties. Only for FP0999 the original Feasibility Pump scored better 65 times, the modified version 21 times, 35 times both got the same result.

Comparing all six variants, FP00 found a solution best among the six 12 times, FP08 23 times, FP09 27 times, FP095 37 times, FP099 30 times, FP0999 12 times. FP00 was the unique winner in 6 cases, in the sense that all solutions found by other variants were worse, FP08 in 11 cases, FP09 in 14 cases, FP095 in 25 cases, FP099 in 22 cases, FP0999 in 8 cases.

For one instance, FP08 was the only variant, which found a solution, FP09 was the only successful variant for one instance, too, FP095 for two instances, FP099 for one.

The geometric mean of the primal gap is 106.7% for FP00, 47.0% for FP08, 43.1% for FP09, 37.8% for FP095, and 39.8% for FP099.

The geometric mean of the running time is 4.7 seconds for FP00, 5.1 for FP08, 5.3 for FP09, 5.6 for FP095, 7.4 for FP099, 9.1 for FP0999. We see, that all Objective Feasibility Pump versions are slower than the original Feasibility Pump.

To put all these data together, using the Objective Feasibility Pump and setting $\varphi = 0.95$ seems to be the best choice. FP095 scored first for the geometric mean of the primal gap, for the number of best solutions found, for the number of unique best solutions, and for the number of instances where it was the only variant which found any solution. FP095 finished second for the number of instances where it found a solution with only one less than FP08, whose solutions have a clearly worse quality.

In comparison to the original Feasibility Pump, using FP095 causes a slowdown of 19% in the geometric mean, but reduces the (geometric mean of the) primal gap to nearly a third.

Other Approaches

There were some other approaches than the one of the Objective Feasibility Pump to deal with the annoying circumstance of a poor solution quality. One was proposed by Fischetti, Glover, and Lodi [24], who suggested to add an objective cutoff and iterate the whole solving process. Let $\mathring{x}$ be an optimal solution of the LP-relaxation of the MIP and $\tilde{x}$ a feasible point found by the Feasibility Pump. Fischetti, Glover, and Lodi suggested to iterate the Feasibility Pump with an additional constraint $c^T x \leq \beta c^T \mathring{x} + (1-\beta) c^T \tilde{x}$ for some fixed parameter $\beta \in (0,1)$. Bertacco, Fischetti, and Lodi proposed to use some Large Neighborhood Search improvement heuristics, namely Local Branching (see Section 4.1) or Rins (see Section 4.2) to improve the quality of the solution found by the Feasibility Pump.

3.2 Rounding Methods

Whenever an LP is solved in the Branch-And-Cut-process, at least one LP-feasible, but probably fractional solution is produced. It seems natural that one tries to round the fractional variables in order to get a feasible solution.

The possible number of roundings of an LP-feasible point doubles with every fractional variable and it is not obvious to which direction variables should be rounded. A fractional value of 0.9 can be interpreted as "nearly 1" as well as "could not be set to 1".

Let $\bar{x}$ be an LP-feasible point. Classical ideas are to round a variable x_j down, if $c_j > 0$ and up, if $c_j < 0$ or to round every variable up with a probability of $\bar{x}_j - \lfloor \bar{x}_j \rfloor$ and down otherwise.

In order to end up at a feasible solution, it seems reasonable that one takes care of the LP-feasibility, when iteratively rounding variables. The approach of the rounding heuristics presented in Section 3.2.2 is to find rounding directions for each variable, which maintain or retrieve the LP-feasibility of a partially rounded point.

Section 3.2.1 describes a heuristic which regards the problem of finding a best rounding as a MIP and is able to enumerate the whole subspace of all possible roundings.

The implementation of the heuristics of Section 3.2.2 has been done by Achterberg, whereas the implementation of the Rens heuristic presented in Section 3.2.1 has been done by the author of the thesis.

3.2.1 RENS

Pure rounding heuristics all work in a common fashion: they round every integer variable which takes a fractional value in an LP-feasible vector to an integer value. Vice versa, this means that every integer variable which is already integral in this LP-feasible vector stays unchanged.

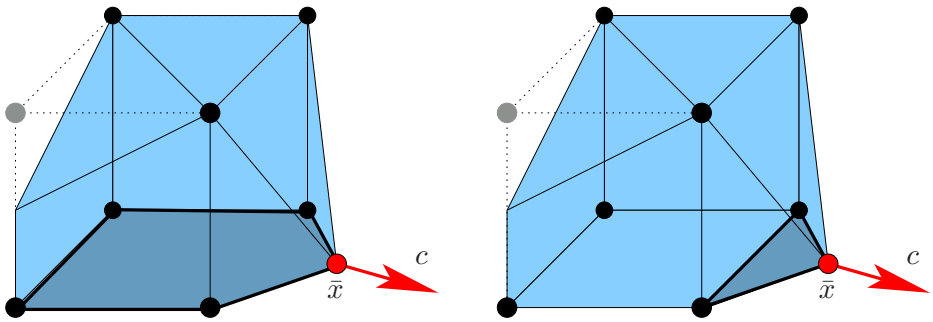


Figure 3.3. Sub-MIPs received by fixing integer variables of $\bar{x}$ (left hand) and additionally changing the variables' bounds (right hand)

This section will describe a heuristic working in the way of a Large Neighborhood Search. The main difference, acoording to the large neighborhood search strategies which will be described in Chapter 4, is that it does not require an existing feasible solution to work. Therefore, it can be used as a start heuristic.

The Idea

Let $\bar{x}$ be an optimum of the LP-relaxation at some node (normally the root) of the Branch-And-Bound-tree. The idea is to create a sub-MIP of the original MIP by changing the bounds of all integer variables to $l_j = \lfloor \bar{x}_j \rfloor$ and $u_j = \lceil \bar{x}_j \rceil$ as it can be seen in Figure 3.3. Note that every integer variable which takes an integral value in $\bar{x}$ gets fixed. As the overall performance of the heuristic strongly depends on $\bar{x}$ we named it *relaxation enforced neighborhood search*, or shortly RENS.

We decided to regard RENS as a rounding heuristic since every feasible solution of the described sub-MIP is a possible rounding of $\bar{x}$.

Because of this, RENS is of special interest for the analysis of pure rounding heuristics. If the sub-MIP created by RENS is proven to be infeasible, no pure rounding heuristic exists which is able to generate a feasible solution out of the fractional LP optimum. If the sub-MIP created by RENS is completely solved, its optimal solutions are the best roundings any pure rounding heuristic can generate.

The outline of RENS is quite simple. We create a sub-MIP

$$\begin{aligned} \min \quad & c^T x \\ \text{such that} \quad & Ax \leq b \\ & l_j \leq x_j \leq u_j \text{ for all } j \in N \backslash I \\ & \lfloor \bar{x}_j \rfloor \leq x_j \leq \lceil \bar{x}_j \rceil \text{ for all } j \in I \\ & x_j \in \mathbb{Z} \text{ for all } j \in I \end{aligned} \tag{3.5}$$

of the original MIP that fixes all integer variables for which the optimum of the LP-relaxation $\bar{x}$ takes an integral value. MIP (3.5) can be easily transformed to an MBP by substituting $y_j = x_j - \lfloor \bar{x}_j \rfloor$. We will identify these two programs and, with a slight abuse of notation, also call (3.5) an MBP or a sub-MBP of the original MIP, respectively.

Implementation Details

For the practical use as a start heuristic integrated into a Branch-And-Bound-framework, one should only call Rens, if the resulting sub-MBP seems to be substantially easier than the original one. Which means that at least a specific ratio of all integer variables, say $r_I \in (0, 1)$, or a specific ratio of all variables including the continuous ones, say $r_N \in (0, 1)$, should be fixed. The first criterion keeps control of the difficulty of the sub-MBP itself, the second one of the LPs that will have to be solved during the solving process. For example, think of a MIP which consists of 20 integer and 10'000 continuous variables. Even if one fixes 50% of the integer variables, Rens would be a time-consuming heuristic since solving the LPs of the sub-MBP would be nearly as expensive as solving the ones of the original MIP. Another way to avoid spending too much time in solving sub-MBPs is to add some limit to the solving process of the sub-MBP. This could be a time limit or a limit on the solving nodes.

We decided to limit the number of solving nodes and the number of stalling nodes of the sub-MBP. The solving node limit is a hard limit on the maximum number of Branch-And-Bound-nodes the MIP-solver should at most process. The stalling node limit indicates how many nodes the MIP-solver should at most process without an improvement in the incumbent solution of the sub-MBP. On the one hand one wants to keep control of the overall running time of the heuristic and hence, use a solving node limit. On the other hand one does not want to abort the procedure too early if the objective value of the incumbent solution decreases continuously and hence use a stalling node limit. Therefore, we decided to use both node limitation strategies simultaneously.

As the idea is rather simple, it can be subsumed to the rather short Algorithm 4.

Computational Results

As mentioned earlier, Rens can be used to determine whether there are solutions of the MIP which are roundings of the optimum of the LP-relaxation and if so, which objective value is the best possible for such solutions. To investigate this, we set the solving node limit of the original MIP to 1, the time limit to 1 hour, the solving node limit and the stalling node limit of the Rens sub-MBP to infinity, r_I and r_N to 0.0. Some expensive presolving strategies and heuristics were deactivated when solving the sub-MBP since using Rens as a start heuristic should avoid such time wasting features.

Input: $\bar{x}$ LP-feasible point
Parameters: nl node limit for the sub-MBP
r_I minimum ratio of integer variables to fix
r_N minimum ratio of all variables to fix

```
1 x̄ ← optimum of LP-relaxation;
2 Create MBP (3.5);
3 if at least r_I · |I| integer variables fixed and at least r_N · n variables
  fixed then
4     Set node limit nl for MIP-solver;
5     Solve MBP (3.5);
```

Algorithm 4: RENS

Table B.3 shows the results of applying RENS with the described settings for our standard test set.

RENS found a feasible solution for 82 out of 129 MIPs, for 47 instances it failed. 23 times it could find an optimal solution of the original MIP. Note that even if $\gamma_P(\tilde{x}) = 0.0\%$ there could be a little gap which is omitted due to rounding the gap (which for example happens for instance `gesa2`). 16 times the RENS sub-MBP could not be solved within the time limit of one hour, for 13 of these 16 instances RENS could find at least one feasible solution.

Which means that for 69 instances the solution found by RENS is the best possible rounding of the LP-relaxation's optimum and for 44 instances there does not exist any feasible rounding.

For the three instances `a1c1s1`, `momentum1`, and `momentum2` it is unknown, whether a feasible rounding of the root-LP optimum exists. RENS did not find any feasible solution within the time limit, but it was not able to prove the infeasibility of the problem.

Sometimes solving the RENS sub-MBP takes more time than solving the original MIP. This is due to the fact that some SCIP parameters which were set for solving the sub-MBP differ from defaults (e.g., the presolving techniques and the cut separation).

There are a bundle of instances for which RENS with this settings consumes too much time, e.g., `qiu`, where SCIP needs 2.7 seconds for processing the root node, but RENS does not finish within an hour. For most of these instances this can be explained by the high number of Branch-And-Bound-nodes processed in order to solve the RENS sub-MBP. Therefore, we suggest to set some limit on the number of nodes, as it is described above. However, there are also some instances such as `dano3_5` for which RENS needs much time, but not so many Branch-And-Bound-nodes. One can see, that the number of fixed variables is relatively small for such instances, e.g., 15.4% of all variables in the mentioned case. Therefore, we recommend to set the parameters r_N and r_I to a sufficiently large value.

Next, we would like to examine the integration of RENS into an LP-based

Criterion	SCIP_{NoR}	$\text{SCIP}_{\text{RRoot}}$	$\text{SCIP}_{\text{RFreq10}}$
Time Geometric Mean	36.7	**35.9**	36.3
Nodes Geometric Mean	704	**614**	615
Fewest Nodes	11	**24**	21
Fastest (of 19)	8	**10**	5

Table 3.3. Summarized results of different integrations of Rens into SCIP, easy instances

Branch-And-Bound-framework. Therefore, we applied SCIP without Rens, SCIP with Rens used in the root node, and SCIP with Rens used at every tenth depth of the Branch-And-Bound-tree on both standard test sets.

We set the stalling node limit to 500, the absolute node limit to 10'000, $r_I = 0.5$, and $r_N = 0.25$. The results are shown in Tables B.4 and B.5.

To keep the following paragraphs short, we abbreviate the three versions as follows: SCIP with Rens only used at the root node should be called $\text{SCIP}_{\text{RRoot}}$, SCIP with Rens completely deactivated should be called SCIP_{NoR}, and SCIP with Rens used every tenth depth should be called $\text{SCIP}_{\text{RFreq10}}$.

For our easy test set, SCIP_{NoR} needs 36.7 seconds in the geometric mean to find and prove an optimal solution, $\text{SCIP}_{\text{RRoot}}$ needs 35.9 seconds, $\text{SCIP}_{\text{RFreq10}}$ 36.3 seconds. SCIP_{NoR} needs 704 nodes in the geometric mean to finish, $\text{SCIP}_{\text{RRoot}}$ needs 614, and $\text{SCIP}_{\text{RFreq10}}$ 615 nodes.

There are 32 instances for which the number of solving nodes differs among the three versions. SCIP_{NoR} needs fewest solving nodes 11 times, $\text{SCIP}_{\text{RRoot}}$ 24 times, and $\text{SCIP}_{\text{RFreq10}}$ 21 times.

In many cases, the differences in the solving time are only marginal. However, there are 19 instances, for which the longest solving time and the shortest solving time differ by more than 10%. Among these, SCIP_{NoR} is fastest 8 times, $\text{SCIP}_{\text{RRoot}}$ 10 times, and $\text{SCIP}_{\text{RFreq10}}$ 5 times.

We also notice that, on our hard test set, SCIP with Rens performs better than SCIP without Rens. At the moment we reach the time or memory limit, $\text{SCIP}_{\text{RRoot}}$ has a smaller primal bound 6 times compared to SCIP_{NoR} which is superior 4 times.

$\text{SCIP}_{\text{RRoot}}$ defeats $\text{SCIP}_{\text{RFreq10}}$ 5 times, for 1 instance the latter one is superior.

The results of calling Rens at the root node with a stalling node limit of 500, an absolute node limit of 10'000, $r_I = 0.5$, and $r_N = 0.25$ can be seen in Table B.6. Rens still finds feasible solutions for 66 instances (of 82 without these settings). It achieves an optimal or best known rounding in 45 cases, and finds an optimal solution 9 times.

After all, Rens seems to be a reasonable root node heuristic.

3.2.2 Some Other Rounding Methods

There are some other rounding methods integrated into SCIP. All of them are easy to describe and hence, they are subsumed to one section.

Simple Rounding

Simple Rounding iterates over the set of fractional variables of some LP-feasible point and, if possible, rounds them to an integral value while staying LP-feasible.

We can state the following about trivially roundable variables (see Definition 1.5): if a trivially down-roundable variable x_j is fractional in some solution of the LP, rounding down this variable yields another solution of the LP, since the activities of all rows get smaller or stay equal. The number of down-locks of a variable x_j equals the number of rows whose values grow, if rounding down x_j, and which could possibly be violated. Obviously, a variable is trivially down-roundable, if its number of down-locks is 0. Everything holds vice versa for up-roundable variables and up-locks.

Let $\bar{x}$ be a solution of the LP-relaxation of the current Branch-And-Bound-node. Let $\bar{I} \subseteq I$ be the index set of all integer variables which are fractional in $\bar{x}$. Simple Rounding iteratively checks for each variable x_i with $i \in \bar{I}$ whether it is trivially down- or up-roundable. If this holds it rounds variable x_i, concluding one iteration step, if not, the procedure stops.

If Simple Rounding is able to round all fractional variables, it will result in a feasible solution of the MIP after $|\bar{I}|$ iteration steps.. Algorithm 5 summarizes the procedure.

```
   Input:        x̄ LP-feasible point
1  Ī ← index set of all fractional variables of x̄ ;
2  for j ∈ Ī do
3      if x_j is trivially down-roundable then
4          x̄ ← x̄ with x̄_j replaced by ⌊x̄_j⌋;
5      else if x_j is trivially up-roundable then
6          x̄ ← x̄ with x̄_j replaced by ⌈x̄_j⌉;
7      else
8          stop!
```

Algorithm 5: Simple Rounding

Rounding

In difference to Simple Rounding, Rounding also performs roundings which may lead to LP-infeasible points and then, tries to recover from this infeasibility by further roundings as it can be seen in Figure 3.4.

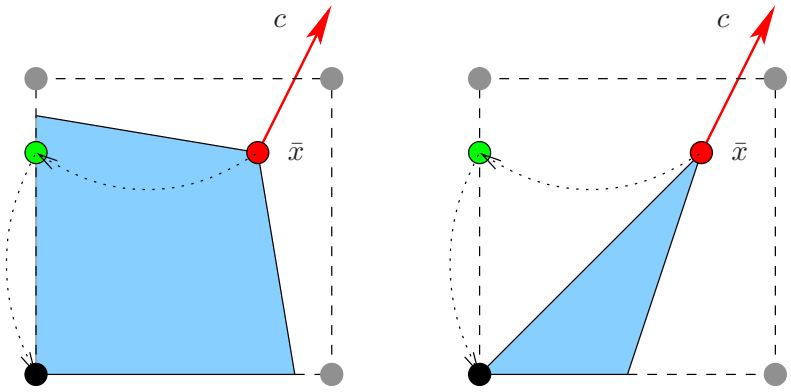


Figure 3.4. Simple Rounding compared to Rounding: Simple Rounding (left) always stays LP-feasible, Rounding is able to recover from an LP-infeasible point

Input: $\bar{x}$ LP-feasible point
1 $x' \leftarrow \bar{x}$;
2 $\bar{I} \leftarrow$ index set of all fractional variables of x' ;
3 **while** $\bar{I} \neq \emptyset$ **do**
4 **if** *there is a violated row* **then**
5 Look for fractional variable x_j, whose rounding decreases violation of this row and minimizes the number of locks;
6 **else**
7 Look for fractional variable x_j, whose rounding to the other side maximizes the number of locks;
8 $x' \leftarrow x'$ with x'_j replaced by x'_j rounded to according direction;
9 $\bar{I} \leftarrow \bar{I} \backslash x_j$;

Algorithm 6: Outline of Rounding

Rounding reduces the number of fractional variables by 1 in each step, just like Simple Rounding does, but follows a selection strategy in order to find a variable which is locally most crucial to round, see Algorithm 7. Let x' be an integer infeasible (and potentially also linear infeasible) point. The choice, which variable of x' to round, is done by the following criteria:

- If there is a violated linear constraint, hence $v_i(x') > 0$, only fractional variables x_j with non-zero coefficients a_{ij} in the row $A_{i\cdot}$ are regarded as candidates for the next rounding.

  Obviously, if $a_{ij} > 0$, the variable x_j has to be rounded down in order to reduce the violation $v_i(x')$ of the constraint, and if $a_{ij} < 0$, the variable x_j has to be rounded up. Let ξ_j be the number of down-locks for variables x_j with $a_{ij} > 0$, and the number of up-locks for variables

x_j with $a_{ij} < 0$.

If there are fractional variables with non-zero coefficients in the row $A_{i\cdot}$, one with minimal ξ_j will be rounded, otherwise the heuristic will abort. If the minimum of ξ_j is not unique, a variable x_j with minimal increase $\Delta c_j = c_j([x_j] - x_j)$ of the objective function is chosen. Here, $[\cdot]$ denotes the rounding to the direction where the violation of the row is reduced.

- If there is no violated linear constraint, hence x' is LP-feasible, one looks for a rounding which seems most crucial in order to maintain LP-feasibility.

  For every variable let ξ_j be the maximum of the number of down-locks and the number of up-locks. The variable with maximal ξ_j is chosen to be rounded into the opposite direction of where the maximum is achieved. As it was shown in the first case, if the maximum of ξ_j is not unique, a variable x_j with minimal increase $\Delta c_j = c_j([x_j] - x_j)$ in the objective function is chosen.

The first criterion is made to recover from an infeasibility by rounding some variable in a direction, such that the infeasibility of the selected violated constraint decreases or vanishes with minimal potential violation of other rows.

The second criterion is made to perform the essential steps as early as possible. Since every variable has to be rounded, one tries to avoid the most "awkward" roundings as soon as possible.

Shifting

The computational results of Section 3.2.1 showed that there often is no feasible pure rounding of an LP-solution. Shifting is a heuristic which does not only round fractional variables, but it is also able to change continuous or integral variables, if this seems necessary. This can be seen in Figure 3.5. Therefore, it is not a pure rounding heuristic like the ones we have described earlier in this subsection. Nevertheless, we will treat it as a rounding heuristic due to its high resemblance to Rounding.

Let $x' \in \mathbb{R}^n$. A *shift* x''_j of a variable x_j denotes:

- a rounding $x''_j = \lfloor x'_j \rfloor$ or $x''_j = \lceil x'_j \rceil$ for a fractional variable x_j, $j \in I$, $x'_j \notin \mathbb{Z}$,
- a variation within the bounds $x''_j \in [l_j, u_j]$ for a continuous variable x_j, $j \in N \backslash I$,
- a variation within the bounds $x''_j \in [l_j, u_j]$ preserving the integrality $x''_j \in \mathbb{Z}$ for an integer variable x_j, $j \in I$, $x'_j \in \mathbb{Z}$.

Like Rounding, Shifting follows different variable selection strategies depending on the existence of a violated row.

Input: $\bar{x}$ LP-feasible point

$x' \leftarrow \bar{x}$;
$\bar{I} \leftarrow$ index set of all fractional variables of x' ;
while $\bar{I} \neq \emptyset$ **do**
  if $\exists i \in \{1, \ldots, m\} : A_i.x' > b_i$ **then**
    $\xi_{\min} \leftarrow \infty$, $\Delta c_{\min} \leftarrow \infty$, $j_{\min} \leftarrow \infty$;
    for $j \in \bar{I}$ **do**
      if $a_{ij} > 0$ **then**
        $\xi_j \leftarrow$ number of down-locks of x_j;
        if $\xi_j < \xi_{\min} \vee \xi_j = \xi_{\min} \wedge c_j(\lfloor x'_j \rfloor - x'_j) < \Delta c_{\min}$ **then**
          $\xi_{\min} \leftarrow \xi_j$, $\Delta c_{\min} \leftarrow c_j(\lfloor x'_j \rfloor - x'_j)$, $j_{\min} \leftarrow j$;
      else if $a_{ij} < 0$ **then**
        $\xi_j \leftarrow$ number of up-locks of x_j;
        if $\xi_j < \xi_{\min} \vee \xi_j = \xi_{\min} \wedge c_j(\lceil x'_j \rceil - x'_j) < \Delta c_{\min}$ **then**
          $\xi_{\min} \leftarrow \xi_j$, $\Delta c_{\min} \leftarrow c_j(\lceil x'_j \rceil - x'_j)$, $j_{\min} \leftarrow j$;
    if $\xi_{\min} = \infty$ **then**
      stop!
    else
      if $a_{ij_{\min}} > 0$ **then**
        $x' \leftarrow x'$ with $x'_{j_{\min}}$ replaced by $\lfloor x'_{j_{\min}} \rfloor$;
      else
        $x' \leftarrow x'$ with $x'_{j_{\min}}$ replaced by $\lceil x'_{j_{\min}} \rceil$;
      $\bar{I} \leftarrow \bar{I} \backslash j_{\min}$;
  else
    $\xi_{\max} \leftarrow -1$, $\Delta c_{\min} \leftarrow \infty$, $j_{\min} \leftarrow \infty$, $\sigma \leftarrow 0$;
    for $j \in \bar{I}$ **do**
      $\xi_j \leftarrow$ number of up-locks of x_j;
      if $\xi_j > \xi_{\max} \vee \xi_j = \xi_{\max} \wedge c_j(\lfloor x'_j \rfloor - x'_j) < \Delta c_{\min}$ **then**
        $\xi_{\max} \leftarrow \xi_j$, $\Delta c_{\min} \leftarrow c_j(\lfloor x'_j \rfloor - x'_j)$, $j_{\min} \leftarrow j$, $\sigma \leftarrow -1$;
      $\xi_j \leftarrow$ number of down-locks of x_j;
      if $\xi_j > \xi_{\max} \vee \xi_j = \xi_{\max} \wedge c_j(\lceil x'_j \rceil - x'_j) < \Delta c_{\min}$ **then**
        $\xi_{\max} \leftarrow \xi_j$, $\Delta c_{\min} \leftarrow c_j(\lceil x'_j \rceil - x'_j)$, $j_{\min} \leftarrow j$, $\sigma \leftarrow +1$;
    if $\sigma = +1$ **then**
      $x' \leftarrow x'$ with $x'_{j_{\min}}$ replaced by $\lceil x'_{j_{\min}} \rceil$;
    else
      $x' \leftarrow x'$ with $x'_{j_{\min}}$ replaced by $\lfloor x'_{j_{\min}} \rfloor$;
    $\bar{I} \leftarrow \bar{I} \backslash j_{\min}$;
Check x' for linear feasibility;

Algorithm 7: Rounding

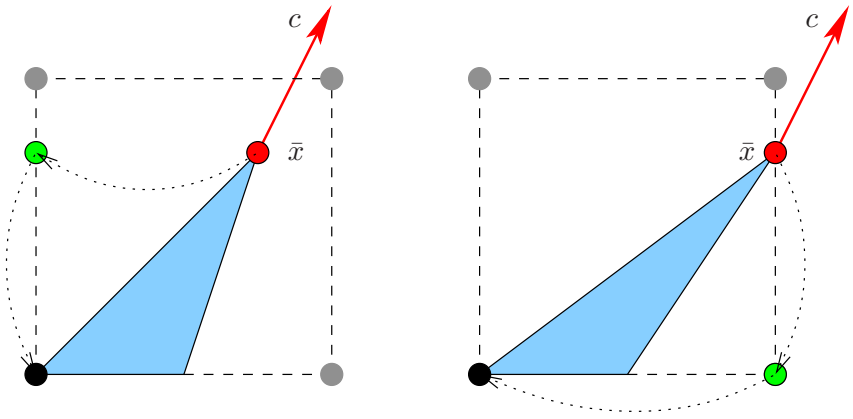


Figure 3.5. Rounding compared to Shifting: Rounding (left) only changes fractional variables, Shifting is able to vary integral variables

- If a violated row exists, Shifting first selects a row whose violation should be vanished or at least decreased. If there are violated rows with fractional variables, Shifting selects one of these and rounds a fractional variable with minimal number of locks.
- If a violated row exists, but none which has fractional variables, a violated row is chosen randomly. Every variable that has a non-zero coefficient in this row will get a score which indicates the urgency to shift this variable. A variable with lowest score will be shifted. The score mainly depends on how often and how many steps before the variable has been shifted to the other side. Additionally, integer variables get an extra "penalty score" in contrast to continuous variables.
- If no violated row exists, Shifting proceeds just like Rounding does: It rounds a fractional variable with maximal number ξ_j of up- or down-locks into the opposite direction. If this maximum is not unique, a variable x_j with a minimal increase $\Delta c_j = c_j([x_j] - x_j)$ in the objective function is chosen.

The procedure terminates, as soon as a feasible solution is found, or after a certain number of *non-improving shifts* have been applied. A non-improving shift is a shift, where neither the number of fractional variables nor the number of violated rows can be reduced.

Taking all these considerations together yields Algorithm 8.

Computational Results

First of all, we will compare the performance of the described rounding heuristics and of the RENS heuristic from Section 3.2.1, if they are applied to a single LP solution, namely the optimum of the root node's LP relaxation.

Input: $\bar{x}$ LP-feasible point

1 $x' \leftarrow \bar{x}, t \leftarrow 0$;
2 $\bar{I} \leftarrow$ index set of all fractional variables of x' ;
3 **while** $\bar{I} \neq \emptyset$ **or** $\exists i \in \{1, \ldots, m\} : A_i.x' > b_i$ **do**
4 $t \leftarrow t + 1$;
5 **if** $\exists i' \in \{1, \ldots, m\} : A_{i'}.x' > b_{i'}$ **then**
6 **if** $\exists i' \in \{1, \ldots, m\} : A_{i'}.x' > b_{i'}$ **and** $\exists j \in \bar{I} \mid a_{i'j} \neq 0$ **then**
7 $i \leftarrow i'$;
8 **else** Select i randomly under all violated rows;
9 $\varsigma_{\min} \leftarrow \infty$, $j_{\min} \leftarrow \infty$;
10 **for** $j \in N \mid a_{ij} \neq 0$ **do**
11 **if** $a_{ij} < 0$ **then**
12 **if** $j \in \bar{I}$ **then**
13 $\xi_j \leftarrow$ number of down-locks of x_j;
14 $\varsigma_j \leftarrow -1 + \frac{1}{\xi_j + 1}$;
15 **else**
16 $S \leftarrow$ {iterations when x_j was shifted down};
17 $\varsigma_j \leftarrow \sum_{s \in S} 1.1^{s-t}$;
18 **if** $j \in I$ **then** $\varsigma_j \leftarrow \varsigma_j + 1$;
19 **if** $\varsigma_j < \varsigma_{\min}$ **then**
20 $\varsigma_{\min} \leftarrow \varsigma_j$,$j_{\min} \leftarrow j$;
21 **else** analog to Lines 12 to 20 with down replaced by up
22 **if** $a_{ij_{\min}} > 0$ **then**
23 $x' \leftarrow x'$ with $x'_{j_{\min}}$ replaced by $\lfloor x'_{j_{\min}} \rfloor$;
24 **else**
25 $x' \leftarrow x'$ with $x'_{j_{\min}}$ replaced by $\lceil x'_{j_{\min}} \rceil$;
26 **if** $j_{\min} \in \bar{I}$ **then** $\bar{I} \leftarrow \bar{I} \backslash j_{\min}$;
27 **else**
28 $\xi_{\max} \leftarrow -1$, $\Delta c_{\min} \leftarrow \infty$, $j_{\min} \leftarrow \infty$, $\sigma \leftarrow 0$;
29 **for** $j \in \bar{I}$ **do**
30 $\xi_j \leftarrow$ number of up-locks of x_j;
31 **if** $\xi_j > \xi_{\max} \vee (\xi_j = \xi_{\max} \wedge c_j(\lfloor x'_j \rfloor - x'_j) < \Delta c_{\min})$ **then**
32 $\xi_{\max} \leftarrow \xi_j$, $\Delta c_{\min} \leftarrow c_j(\lfloor x'_j \rfloor - x'_j)$, $j_{\min} \leftarrow j$, $\sigma \leftarrow -1$;
33 $\xi_j \leftarrow$ number of down-locks of x_j;
34 **if** $\xi_j > \xi_{\max} \vee (\xi_j = \xi_{\max} \wedge c_j(\lceil x'_j \rceil - x'_j) < \Delta c_{\min})$ **then**
35 $\xi_{\max} \leftarrow \xi_j$, $\Delta c_{\min} \leftarrow c_j(\lceil x'_j \rceil - x'_j)$, $j_{\min} \leftarrow j$, $\sigma \leftarrow +1$;
36 **if** $\sigma = +1$ **then**
37 $x' \leftarrow x'$ with $x'_{j_{\min}}$ replaced by $\lceil x'_{j_{\min}} \rceil$;
38 **else**
39 $x' \leftarrow x'$ with $x'_{j_{\min}}$ replaced by $\lfloor x'_{j_{\min}} \rfloor$;
40 $\bar{I} \leftarrow \bar{I} \backslash j_{\min}$;
41 Check x' for linear feasibility;

Algorithm 8: Shifting

Criterion	Simple Rounding	Rounding	Shifting	RENS
Solution Found	26	37	55	**66**
Feasible Rounding Found in % (of 92)	28	40	–	**72**
Optimal Rounding	0	0	–	**45**
Total Time	**0.1**	3.4	8.2	932.6

Table 3.4. Summarized results of different rounding heuristics applied to the optimum of the LP-relaxation

For this test, we modified the SCIP implementations of Simple Rounding, Rounding, and Shifting such that they are only applied to the optimum of the LP-relaxation, not to every LP-solution during the cutting plane separation loop.

For RENS we used the settings as they are described in the Computational Results of Section 3.2.1: the stalling node limit was set to 500, the absolute node limit to 10'000, the minimal fixing rates $r_I = 0.5$, and $r_N = 0.25$.

Table B.6 shows the results of this test and compares the rounding heuristics from this chapter to the best known rounding which can be seen in Table B.3. The values of the last two columns are taken from Table B.3. Note that since Shifting is able to unfix integer variables, it could find solutions, which are not a pure rounding of the LP-optimum. Table 3.4 summarizes the main results taken from Table B.6. In Columns "Optimal Rounding" and "Feasibile Rounding Found", Shifting gets a bar "–" since it can produce solutions which are not a rounding of the starting point.

Since Rounding is an extension of Simple Rounding, and Shifting is an extension of Rounding, we would expect that Shifting dominates Rounding and Rounding dominates Shifting in terms of the number of found solutions. We would further expect that Simple Rounding dominates Rounding and Rounding dominates Shifting in terms of the running time.

Simple Rounding succeeds in finding a feasible solution 26 times, Rounding 37 times, and Shifting 55 times. For all instances where Simple Rounding found a solution, the other two variants found the same one, respectively a solution of the same objective value. Analogously, for all instances for which Rounding found a solution, Shifting found one of the same objective value. This substantiates the statement that these heuristics are extensions of each other.

A feasible rounding is known for 89 instances, and for 3 instances we do not know whether a feasible rounding exists (see computational results of Section 3.2.1). Simple Rounding finds some feasible rounding for 28% of these 92 instances, Rounding succeeds for 40%, RENS for 72%. Note, that neither Simple Rounding nor Rounding succeeded in finding an optimal or best known rounding in any case. In some cases, Shifting gets even better solutions than an optimal rounding. This is due to the fact that shifting does not only consider pure roundings.

The computation time is small for all three variants, but increases with the complexity of the method. The overall computation time for all 129

Criterion	Simple Rounding	Rounding	Shifting	Rens
Solution Found (of 129)	27	39	61	**66**
Better Than Single Call	14	29	41	**72**
Total Time	**0.6**	23.6	54.5	932.6
Time Geometric Mean	**1.0**	**1.0**	1.1	2.0
Only Solution (Shifting vs. Rens)	–	–	9	**13**
Better Solution(Shifting vs. Rens)	–	–	4	**47**

Table 3.5. Summarized results of different rounding heuristics applied to every LP-feasible point of the LP-solving loop

instances is 0.1 seconds for Simple Rounding, 3.4 seconds for Rounding, and 8.2 seconds for Shifting.

Next we want to compare Shifting, the best of the three heuristics we described in this section, to Rens, the heuristic presented in Section 3.2.1. 7 times Shifting found a solution, when Rens failed. There were 48 instances for which a feasible solution was found by Rens as well as by Shifting. In 2 of these 48 cases, the one found by Shifting had a better objective value, 46 times the one found by Rens was superior.

All three heuristics are not very time consuming. The computation time of the slowest heuristic summed up over all instances was 8.2 seconds, whereas the overall time which SCIP needed to solve the root LPs was about 7200 seconds. This is their advantage if comparing them to Rens, a quite time consuming heuristic, which needed 932.6 seconds summed up over all instances. It seems reasonable to apply the three heuristics presented in this section not just to the LP-optimum, but to all LP-feasible points the underlying LP-solver creates during its solving loop.

Therefore, we performed another test run on our two standard test sets, using the SCIP default settings of applying the three rounding heuristics to every LP-solution during the cut separation loop at the root node.

The results can be seen in Table B.7. The best known rounding is missing in this table, since it only referred to rounding the LP-optimum after having finished the cut separation loop. Table 3.5 is a summary of the results from Table B.7.

Altogether there were 27 instances (versus 26 in the previous test) for which Simple Rounding found a solution, 39 (vs. 37) for which Rounding succeeded, and 61 (vs. 55) instances for which Shifting was successful.

As before, the solutions found by Rounding are at least as good as the ones found by Simple Rounding. There were 9 instances where both heuristics found a feasible solution, but the one found by Rounding was strictly better with respect to the objective. There were 5 instances for which the solution found by Shifting was superior to the one found by Rounding.

Of course, the best solution of an instance found in this test run was as least as good as the one found in the previous run. Simple Rounding called for every LP-solution yielded a strictly better solution than Simple Rounding called only for the LP-optimum 14 times, Rounding called for every LP-

solution was superior to the single call case 29 times and Shifting improved its quality 41 times.

Comparing Shifting and RENS again, we see that there were 52 instances for which both found a solution, 9 instances for which only Shifting succeeded, and 13 instances for which RENS found a solution, but Shifting failed. For 4 out of these 52 instances, the solution found by Shifting was superior, 47 times RENS found a better solution, once both achieved the same objective value.

The overall computation time needed by each of the heuristics grew roughly by a factor between 6 and 7. The multi call variant of Simple Rounding needed 0.6 seconds in sum, Rounding 23.6 seconds, Shifting 54.5 seconds. These values are still small compared to the overall computation time of 2 hours SCIP needed for presolving and solving the root LPs or to the 933 seconds RENS needed.

Since the number of instances for which solutions were found, and especially the quality of the solutions increased substantially, it seems reasonable to call Shifting for every LP solution, at least at the root node. The overall computation time of Simple Rounding seems small enough, that this heuristic could be called at every node of the Branch-And-Bound-tree.

In the majority of the cases, RENS is superior to the three other heuristics in terms of the objective, but they have two advantages. First, they need few time and can hence be applied more frequently. Second, they even find better solutions in some cases. The extensive computations of RENS and the straight-forward-strategy of Simple Rounding, Rounding, and Shifting seem to complement each other in a useful way.

3.3 Octane

Balas, Ceria, Dawande, Margot, and Pataki [10] provided an extensive description of a heuristic for pure binary programs which in its strategy is very different from all other classes of start heuristics. They called it OCTANE, an abbreviation for OCTAhedral Neighborhood Search. It is based on a highly geometrical idea.

The Idea – Geometric Background

Clearly, every point $\tilde{x} \in \tilde{X} := \{0,1\}^n$ is integer feasible for any BP. One can observe that there is a one-to-one correspondence between these 2^n points and the 2^n facets of an n-dimensional octahedron.

The basic idea of OCTANE is that finding facets that are near an LP-feasible point $\bar{x}$ is equivalent to finding integer feasible points $\tilde{x}$ that are near $\bar{x}$ and therefore potentially LP-feasible themselves. Furthermore, if we choose the optimum of the LP-relaxation as our initial point $\bar{x}$, they potentially will be of high quality in terms of the objective function. These facets will be determined as k facets of an n-dimensional octahedron first hit by a ray

starting at $\bar{x}$. Obviously, choosing the direction of the ray is the crucial part of this ray shooting algorithm.

First of all, we specify the relationship between the elements of $\tilde{X}$ and the facets of an octahedron.

Definition 3.4 *Let $e := \{1\}^n$ be the vector of all ones.*

$$K := \{x \in \mathbb{R}^n \mid -\frac{e}{2} \leq x \leq \frac{e}{2}\}$$

is called the unit hypercube, centered at the origin.

$$\begin{aligned} K^\star &:= \{x \in \mathbb{R}^n \mid \|x\|_1 \leq \frac{n}{2}\} \\ &= \{x \in \mathbb{R}^n \mid \delta^T x \leq \frac{n}{2} \text{ for all } \delta \in \{\pm 1\}^n\} \end{aligned}$$

is called its circumscribing octahedron.

For simplification, we will identify the vector δ with the hyperplane $\delta^T x = \frac{n}{2}$ it creates and which contains exactly one facet of the octahedron $K^\star$. We call both, the vector and the corresponding hyperplane, just facet.

Note that every facet δ of the octahedron $K^\star$ contains exactly one vertex ν of the hypercube K, namely the one with $\nu_j = \frac{1}{2}$ if $\delta_j = 1$ and $\nu_j = -\frac{1}{2}$ if $\delta_j = -1$. Since both sets have the same cardinality, every vertex of the hypercube is as well contained in exactly one facet of the octahedron.

This correspondence is kept if translating K and $K^\star$ by the same vector. Consider the hypercube $K + \frac{e}{2} = \{x \in \mathbb{R}^n \mid 0 \leq x_j \leq 1\}$. Its vertex set is exactly $\tilde{X}$, the set of all integer feasible points, and its circumscribing octahedron is $K^\star + \frac{e}{2}$. Hence we have a one-to-one correspondence between $\tilde{X}$ and the facets of $K^\star + \frac{e}{2}$.

Nevertheless, we will describe the whole procedure on K and $K^\star$ and not on the translated version, since we would like to take advantage of the origin symmetry of K and $K^\star$ which results in a much easier notation.

Therefore, we will at first transform our initial point $\bar{x}$, normally the optimum of the LP-relaxation, to its corresponding point $\bar{\chi} := \bar{x} - \frac{e}{2}$. After finishing the ray shooting algorithm, every vertex ν^i of K it determined, will be transformed to an integer feasible point $\tilde{x}^i = \nu^i + \frac{e}{2}$.

A Ray Shooting Algorithm

The ray shooting algorithm should find k facets $\{\delta^1, \ldots, \delta^k\}$ which are first hit by the half-line

$$\begin{aligned} r \colon \mathbb{R}^{>0} &\to \mathbb{R}^n \\ \lambda &\mapsto \bar{\chi} + \lambda a \end{aligned} \tag{3.6}$$

with $a \neq 0$ being the direction of the ray. These k facets correspond to k vertices $\{\nu^1, \ldots, \nu^k\} \subset K$ and hence, to k integer feasible points $\{\tilde{x}^1, \ldots, \tilde{x}^k\}$.

Definition 3.5 *We call a facet $\delta \in \{\pm 1\}^n$ reachable, if there exists a $\lambda > 0$ for which $\delta^T r(\lambda) = \frac{n}{2}$.*

A facet is called first-reachable, if the parameter λ is minimal among all reachable facets.

The set of reachable facets contains exactly all facets which get hit by the ray r defined in (3.6). Note that the first-reachable facet will not be unique, if the ray hits the facets just at a point where two or more facets intersect each other.

The outline of the ray shooting algorithm is the following: at first, some facet is generated, which is definitely reachable. Secondly, by changing components of this facet a first-reachable facet is determined. At last, one performs a reverse search in order to find $k-1$ further facets.

Input: $\bar{x}$ LP-feasible point
a direction of the shooting ray
Parameters: k number of integer feasible points to create
1 Transform coordinates appropriately (see Figure 3.6);
2 Find a first-reachable facet $\delta^\star$;
3 Incrementally construct k facets nearest to $\bar{\chi}$;
4 Transform these facets to k integer feasible points $\{\tilde{x}^1, \ldots, \tilde{x}^k\}$ of the BP ;
5 Check the integer feasible points $\{\tilde{x}^1, \ldots, \tilde{x}^k\}$ for LP-feasibility;

Algorithm 9: Outline of OCTANE

Before the algorithm is able to start two further transformations are done which the authors stated as general assumption (3.14). Firstly, for all $j \in B$ for which $a_j < 0$, the signs of a_j and $\bar{\chi}_j$ are flipped. Secondly, the coordinates are sorted by nondescending values $\frac{x_j}{a_j}$. This leads to a problem which is completely symmetric to the original one, compare Figure 3.6. Of course, one has to flip the signs of the according ν_j^i and resort after finishing the algorithm.

Then the first step, namely finding any reachable facet, is as trivial as it can be. $\delta = e = \{+1\}^n$ fulfills the requirement, as $a_j \geq 0$ for all $j \in B$ and $a \neq 0$.

The following theorem deals with the problem of finding a first-reachable facet, i.e., one nearest to $\bar{\chi}$, if you have any reachable facet δ at hand:

Theorem 3.6 *1. If a facet δ is reachable, but not first-reachable, there exists an $i \in N$ for which the facet $\delta \diamond i$ defined by*

$$(\delta \diamond i)_j := \begin{cases} -\delta_j & \text{if } j = i \\ \delta_j & \text{otherwise} \end{cases}$$

is a facet which is hit before δ. Then i is called a decreasing flip.

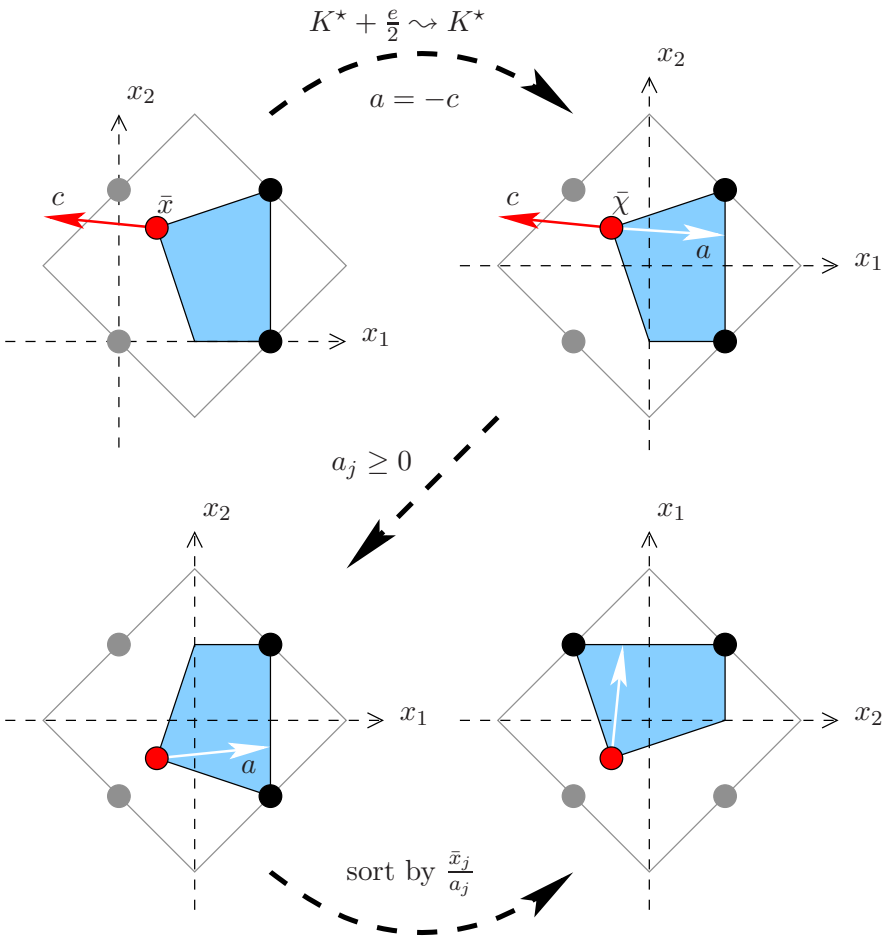


Figure 3.6. Three transformations are applied to $\bar{x}$ and a

2. *Starting with $\delta = e$ and iteratively flipping its components, if they yield a decreasing flip, is an algorithm which has a first-reachable facet as output and can be implemented with a running time of $O(n \log n)$.*

For detailed proofs of these statements, please see pages 5-10 of [10]. Note that the running time the authors gave in Lemma 3.7 differs from the one stated above, since we took also the time into account that is needed for sorting the components $j \in N$ by ascending values $\frac{\bar{x}_j}{a_j}$.

Now, we know how to get a facet which is first-reachable. Note that there could be more than one first-reachable facet. In practice this is rather the rule, if one chooses ray directions not randomly, but following some certain geometric structure [10].

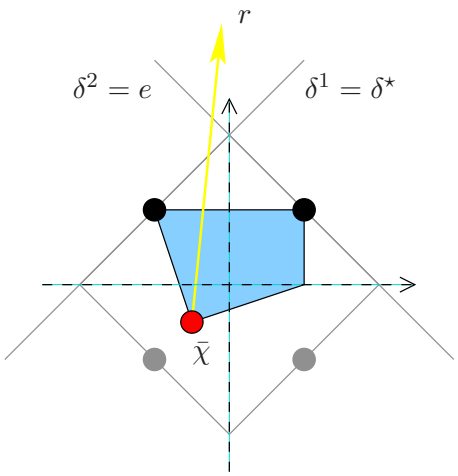


Figure 3.7. Shooting ray hits two facets, each associated with a feasible solution

Reverse Search

At next, we will focus our attention on the problem of finding the rest of the k facets we need. Therefore, one builds up an arborescence which has a first-reachable facet as root-node and reachable facets as vertices.

All arcs connect facets which can be obtained from each other by flipping the sign of a single component, but not every possible flip is represented by an arc.

We want to define a unique predecessor pred(δ) for each reachable facet δ, except for one first-reachable facet which will be the root-node of the arborescence.

Definition 3.7 *We call $i \in N$ a*

- *decreasing $+$ to $-$ flip for δ, if $\delta_i = +1$ and $\delta \diamond i$ is a facet hit before δ,*
- *nonincreasing $-$ to $+$ flip for δ, if $\delta_i = -1$ and $\delta \diamond i$ is a facet hit by r, but not after δ.*

The terms decreasing and nonincreasing hereby refer to the change of λ. If there exists at least one decreasing $+$ to $-$ flip for δ, let i be the one with the minimum index. Then, pred(δ) $:= \delta \diamond i$. If there exists no decreasing $+$ to $-$ flip for δ, but at least one nonincreasing $-$ to $+$ flip, let i be the one with the maximum index. Then, pred(δ) $:= \delta \diamond i$. Otherwise, pred(δ) stays undefined. One can prove that there is exactly one facet $\delta^\star$ for which pred($\delta^\star$) is undefined and obviously this is a first-reachable facet.

Consider the following weighted digraph:

$$\begin{aligned} V :=& \{\text{all reachable facets } \delta \in \{\pm 1\}^n\} \\ A :=& \{(\text{pred}(\delta), \delta) \text{ for all } \delta \in V \backslash \{\delta^\star\}\} \\ w(\text{pred}(\delta), \delta) :=& \text{ (distance between the two points,} \\ & \quad \text{where } r \text{ hits pred}(\delta) \text{ and } \delta) \text{ for all } (\text{pred}(\delta), \delta) \in A \end{aligned}$$

Balas, Ceria, Dawande, Margot, and Pataki [10] proved:

Theorem 3.8 *Let V, A, w as mentioned above.*

1. *$G = (V, A, w)$ is an arborescence rooted at $\delta^\star$.*
2. *There is an $O(kn \log k)$ algorithm which finds k vertices of G with minimum distance to $\delta^\star$.*
3. *These k vertices of G correspond to the k facets of $K^\star$ first hit by r.*

Selection of the Ray Direction

However, we still need to know in which direction the ray should be shot. The authors of [10] gave four different approaches:

- *the objective ray:* There is already one ray, one has, when solving a MIP: the direction of the objective function c. It seems reasonable to shoot a ray into the opposite direction $-c$, as this clearly leads into the inner of the polyhedron and promises to produce feasible solutions.
- *the difference ray:* If OCTANE is not called at the root-node of the Branch-And-Bound-tree, but somewhere deeper, the difference between the optimum of the LP-relaxation at the root-node and the current LP-optimum can indicate some useful information. It shows a part of the development of each variable from the value in an optimal LP-solution to the one in an integer feasible solution. Therefore, this difference vector seems to be a ray direction with a promising geometric interpretation. The same vector is used for Linesearch Diving, see Section 3.1.1.
- *the average ray:* The optimal basis the simplex algorithm found for the LP-relaxation at the current node defines a cone C which has the optimum the LP-solver found as apex. The extreme rays of C are defined by edges of the current LP-polyhedron. The average of the normalized extreme rays of C points into the inner of the polyhedron $P(A, b, l, u)$ and therefore, hopefully into the direction of some feasible solutions. Balas [9] described that one can easily get the extreme rays from the last simplex tableau.

- *the average weighted slack ray:* This ray is obtained from the one mentioned before by additionally assigning weights to the extreme rays. Every extreme ray corresponding to a non-basic slack variable with positive reduced costs gets the inverse of the reduced costs as weights, all others weights are assigned to 0.

We additionally tested another direction which seems to be promising:

- *the average normal ray:* The LP-optimum is a vertex of the polyhedron $P(A, b, l, u)$ and therefore, at least n of the linear and bounding constraints are fulfilled with equality. The normalized (inner) normal of a hyperplanes corresponding to some linear constraint gives a direction where all points are feasible for this constraint. It is the hope that the average of all these normals indicates a direction where one can find feasible points.

Balas, Ceria, Dawande, Margot, and Pataki [10] proposed to carry out the enumeration on the space of fractional variables and not on the whole space of variables. Their computational results ([10], Tables A.1 and A.2) show that this is only slightly inferior in terms of the objective function of the best solution, but noticeably superior in terms of the computation time. Furthermore, they suggested to abort the construction of facets if all of the say k_1 first facets violate a common linear constraint.

All these considerations are summed up in Algorithm 10.

Computational Results

As OCTANE is a heuristic designed for BPs, we tested it on the set of all BPs taken from our two standard test sets. The first test compares the four ray selection rules which can be applied to any point of an LP: the *objective ray*, the *average ray*, the *average weighted slack ray*, and the *average normal ray*. The *difference ray* needs some information about the Branch-And-Bound-process and is therefore regarded later, when we are looking at the integration of OCTANE into SCIP.

We took the optimum of the LP-relaxation attained by SCIP with standard settings as LP-feasible point $\bar{x}$ defining the origin $\bar{\chi}$ of the shooting ray. We applied OCTANE once with each of the four rules to each of the 45 problems. The node limit of SCIP was set to 1, all other heuristics were deactivated. Furthermore, we ran SCIP with the three rounding heuristics Simple Rounding, Rounding, Shifting and the integrated Feasibility Pump version. Thus, it is possible to compare the solutions OCTANE finds to the solutions of the other root-node heuristics of SCIP. RENS was deactivated, since it is a very investigative heuristic, and it seemed unfair to us to compare it to OCTANE. The implementation of OCTANE has been done by the author of this thesis.

Input: $\bar{x}$ LP-feasible point
a direction of the shooting ray
Parameters: k_1 number of integer feasible points to create first
k number of integer feasible points to create at all

1 Translate $\bar{\chi} \leftarrow \bar{x} - \frac{e}{2}$;
2 Flip coordinates x_j with $a_j < 0$;
3 Resort coordinates by nondescending values $\frac{\bar{x}_j}{a_j}$;
4 $\delta^\star \leftarrow e$;
5 **for** $j \leftarrow 1$ **to** n **do**
6 ⌊ **if** $\delta^\star \diamond j$ *is a decreasing flip* **then** $\delta^\star \leftarrow \delta^\star \diamond j$;
7 **for** $j \leftarrow n$ **down to** 1 **do**
8 ⌊ **if** $\delta^\star \diamond j$ *is a nonincreasing* $-$ *to* $+$ *flip* **then** $\delta^\star \leftarrow \delta^\star \diamond j$;
9 Initialize $\mathsf{L} \leftarrow \delta^\star$, a list of facets sorted by their distance to $\bar{\chi}$;
10 **for** $j \leftarrow 1$ **to** k_1 **do**
11 ⌊ $\delta \leftarrow$ j-th element of L; Determine all facets which can be obtained from δ by a single nondecreasing flip and insert them into L, if not already contained;
12 **if** *All integer feasible points associated with the first* k_1 *elements of* L *violate a common constraint* **then stop!**
13 **for** $j \leftarrow k_1 + 1$ **to** k **do**
14 ⌊ $\delta \leftarrow$ j-th element of L ;
15 ⌊ Determine all facets which can be obtained from δ by a single nondecreasing flip and insert them into L, if not already contained;
16 Apply inverse transformations of Steps 1 to 3 to the first k elements of $\mathsf{L} \rightsquigarrow \{\tilde{x}^1, \ldots, \tilde{x}^k\}$;
17 Check $\{\tilde{x}^1, \ldots, \tilde{x}^k\}$ for linear feasibility;

Algorithm 10: OCTANE

Criterion	Objective Ray	Average Ray	Avg Weighted Slack Ray	Avg Normal Ray
Solution Found (of 45)	15	8	6	15
Best Solution	7	4	4	11
Better Than SCIP heuristics	5	1	5	7

Table 3.6. Summarized results of of different ray directions

Table B.8 shows the results of this test. In order to keep it compact, we just show those problems for which at least one ray selection rule was able to find a feasible solution. Table 3.6 summarizes the results.

There were 17 out of 45 instances for which at least one rule bore a feasible solution. The objective ray and the average normal ray were the most fruitful directions. OCTANE could find feasible solutions for 15 instances either way. The variant which uses the average ray was successful in 8 cases, the one with the average weighted slack ray in 6 cases. OCTANE using the average normal ray found a solution, minimal among the four versions, in 11 cases, using the objective ray in 7 cases, using the average weighted slack ray in 4 cases, and finally using the average ray in 4 cases.

It is worth mentioning that for each rule there is at least one instance for which this rule was the only one which found a solution of this minimal value, e.g., `air03` for the average normal ray, `fiber` for the objective ray, `irp` for the average weighted slack ray, and `neos21` for the average ray. Although the average weighted slack ray has the fewest number of instances for which it found solutions, these solutions are of a remarkable quality within less than 1% gap to the optimal or best known solution in 3 out of 6 cases. Additionally, for two instances the average weighted slack ray was the only variant for which the minimal value could be achieved.

To sum it up, the average normal ray seems to be the best one, as it scored first in the number of instances for which it found feasible solutions as well as in the number of instances for which the objective value of the solution found by a certain rule was minimal. Hence, it is a good amendment to the ray rules suggested by Balas, Ceria, Dawande, Margot, and Pataki [10]. As all four rules performed well, we decided to integrate them all into our OCTANE implementation and to use them alternately in consecutive calls of OCTANE.

OCTANE was only able to find a feasible solution for 37.8% of the BPs in our test set, but the solutions often were of high quality, as Table B.8 shows. There were 10 out of 17 instances, where the best solution, delivered by OCTANE, was superior to the best solution which we obtained by the other root-node heuristics. The average ray could defeat SCIP once, the objective and the average weighted slack ray 5 times, and the average normal ray 7 times.

Next, we want to examine, how OCTANE changes the overall performance of SCIP and whether the difference ray is useful as fifth integrated ray selection rule. We ran SCIP without OCTANE, with OCTANE and only four ray selection rules activated, with OCTANE and all five ray selection rules

activated, and with Octane called only at the root-node. For the second and third variant we choose to call Octane at every tenth depth of the Branch-And-Bound-tree.

The test run was made on the same set like before, namely all BPs from our two standard test sets. Tables B.9 and B.10 show the results of this test.

All four variants yielded very similar results. Regarding the easy instances, in most cases the running time of SCIP was roughly the same. However, the two variants calling Octane frequently tended to take slightly more time. There were only few instances for which the difference of the solving time was remarkable, namely `cap6000` and `irp`, for which the variants which call Octane at most once were superior. There were only two instances, namely `p0282` and `p0548`, for which calling Octane helped decreasing the number of solving nodes, but without an improvement in the solving time.

Regarding the hard instances, we can observe that the two variants with frequent calls of Octane yield a better primal bound for `markshare2`, although this solution was not found by Octane.

Summarizing, the integration of Octane has only a slight influence on the overall performance. The improvements made on few instances result in a higher running time for most instances. The average normal ray and the average weighted slack ray seem to complement each other in a good way. Furthermore, the usage of the difference ray causes no remarkable changes in the performance of Octane.

Chapter 4

Improvement Heuristics

Since MIP-solving is $\mathcal{NP}$-hard [54], finding any feasible solution of a MIP is in theory as hard as finding an optimal one. In practice, however, there are classes of MIP instances for which finding a feasible solution is much easier than finding the optimal one, e.g., set covering problems [18]. State-of-the-art MIP-solvers are able to produce feasible solutions for a lot of different MIP instances in early steps of the Branch-And-Bound-process (compare Section 3.1.2, [48] or [15]).

The goal of an improvement heuristic is to form a new feasible solution of better objective value out of one or more given feasible solutions, trying to use some provided information. One of the first improvement heuristics described in literature was the *k-OPT* heuristic for TSPs [43].

The *Local Search* [57, 31] approach generalizes the idea of k-OPT: one defines a neighborhood of some reference point (the feasible solution), determines a point of this neighborhood which is optimal for some function (e.g., the objective function of the MIP or a feasibility measure), which is then used as a new reference point in the next iteration step. Most improvement heuristics can be formulated as Local Search methods.

Classical Local Search uses relatively small neighborhoods which can be quickly explored and performs a couple of iteration steps, building up a network of visited points. Several classical Local Search methods have been developed for specific optimization problems (see [23, 51, 55] among others). There have also been some advances in integrating classical Local Search into MIP solving [57, 32].

Large Neighborhood Search or shortly *LNS* is a variant of Local Search. It incorporates the complexity of general MIPs by defining a relatively large neighborhood of some reference point (normally the incumbent solution) and performing just a single iteration step, where the neighborhood is completely or partially searched. A lot of the recent methods which integrate Local Search ideas into general MIP solving are based on the idea of LNS [21, 26, 52].

Obviously, finding a good definition of the neighborhood is the crucial part of LNS heuristics. The neighborhood should contain high quality solutions,

these solutions should be easy to find, and the neighborhood should be easy to process. Often, these three goals are conflicting in practice.

For example, neighborhoods for LNS can be defined by evaluating a function measuring the distance to the reference point in some metric as in Local Branching (see Section 4.1), by comparing the reference point with the optimum of the LP-relaxation as in RINS (see Section 4.2) or with other feasible solutions as in Crossover (see Section 4.3), or by just randomly fixing a certain number of integer variables in a given feasible solution as in the Mutation heuristic (see Section 4.4).

There are several methods of how to search such a neighborhood. One could just enumerate it, one could construct a special heuristic to search the neighborhood or, as it has an analog structure, process it in the same fashion as one processed the original problem. All the methods described in this section will be of the last type.

In our cases, the neighborhood can be expressed as a sub-MIP of the original MIP. The Local Branching sub-MIP is constructed by adding constraints to the original MIP. The sub-MIPs used in RINS, Crossover, and Mutation are generated by fixing an adequate number of variables of the original problem to the value they take in at least one feasible solution.

Shrinking the problem size by fixing variables often results in a much easier sub-MIP. One can easily observe that fixing a single variable of an BP halves the number of integer feasible solutions and a similar statement holds for MIPs with bounded variable domains. One can hope that this behavior is transmitted to the solving process of the sub-MIP. In addition, many MIPs regarded in practice have a strong combinatorial structure, whose difficulty decreases rapidly if some variables are fixed. For example, if one fixes one variable of a set partitioning constraint (see [18]) to 1, all other variables of this constraint can be fixed to 0.

The SCIP-implementations of all heuristics presented in this chapter have been done by the author of this thesis.

4.1 Local Branching

Local Branching was introduced by Fischetti and Lodi [25] as a new branching strategy for MIPs which can also be effectively used as a heuristic improving the incumbent solution. The same authors [26] suggested to use Local Branching as a method for driving nearly feasible solutions towards feasibility. This can also be used for the analysis of infeasible MIPs (see [26]).

The usage of Local Branching as an improvement heuristic relies on the observation that the neighborhood of a feasible MIP-solution in terms of the Manhattan distance of two points often contains further solutions of possibly better quality.

The Idea: Soft Fixing as Motivation for Local Branching

Diving heuristics (Section 3.1) usually perform *hard fixings* on variables until they attain a feasible solution or an infeasible subproblem. That means they eliminate integer variables by fixing them to some promising value and resolve some LP, in order to determine the next fixing candidates. This procedure is iterated, and thereby the subproblems get smaller and smaller.

Unfortunately, diving heuristics often end in an infeasible subproblem. This is due to the fact that on the one hand in the early steps of the diving process one cannot predict how the fixing decisions will affect the feasibility of the later subproblems and on other hand in the later steps it is hard to find out which former fixings caused the infeasibility.

The crucial idea is to perform a *soft fixing* for a given integer feasible point $\tilde{x}$ in order to get a linearized expression of its neighborhood which can be integrated into the original MIP description. Soft fixing demands that a certain number of variables, e.g., 90%, takes the same values as in the incumbent solution, but it does not concretely fix any of those variables. The following definition induces such a soft fixing scheme, as it describes the set of all integer feasible points which do not differ in more than k variables from a reference point x.

Definition 4.1 *Let $x \in \mathbb{R}^n$ with $x_j \in \mathbb{Z}$ for all $j \in I$, and let $k \in \mathbb{N}$. The set*

$$\{y \in P(A, b, l, u) \mid \sum_{j \in I} |x_j - y_j| \leq k,\ y_j \in \mathbb{Z} \text{ for all } j \in I\} \tag{4.1}$$

is called the k-neighborhood *of x.*

An example for a 2-neighborhood of a feasible solution of a MIP with general integer variables is given in Figure 4.1.

If there is an incumbent solution $\tilde{x}$ of the MIP, one wants to search its k-neighborhood, which hopefully contains further solutions of better quality. If one wants to do this using a MIP-solver, you need to find a linearized expression of (4.1) which you can integrate into the MIP. Local Branching in its variant for MBPs realizes this by adding a single linear constraint. If one wants to apply Local Branching on MIPs with general integer variables, you need to introduce some extra variables, see the definition of the MIP (4.5) at the end of this section.

Obviously, if regarding MBPs, the distance function Δ^B (compare Definition 3.2) from Section 3.1.2 is suited for the linearization of the k-neighborhood stated in Definition (4.1).

The *local branching cut*

$$\Delta^B(x, \tilde{x}) = \sum_{j \in B: \tilde{x}_j = 0} x_j \ + \sum_{j \in B: \tilde{x}_j = 1} (1 - x_j) \leq k \tag{4.2}$$

describes the k-neighborhood of the central point $\tilde{x}$ to improve. $k \in \mathbb{N}$ should be a given parameter here, which can be updated by Local Branching

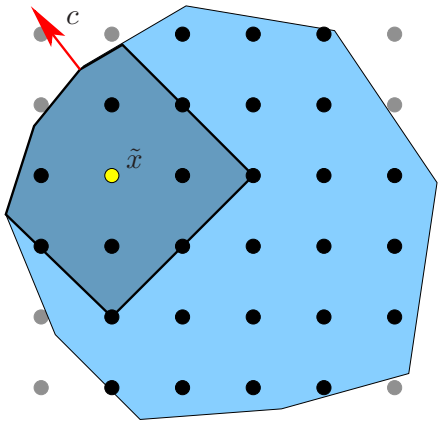


Figure 4.1. Idea of Local Branching: $P(A, b, l, u)$ intersected with the 2-neighborhood of $\tilde{x}$

in further calls. The original MBP supplemented by constraint (4.2) and an objective cutoff

$$c^T x \leq (1 - \epsilon) c^T \tilde{x} \text{ for some } \epsilon > 0 \tag{4.3}$$

tends to be a sub-MBP noticeably easier to solve than the original MBP and it is embedded in an area of good objective function values [26]. Algorithm 11 sketches Local Branching in its variant as an improvement heuristic for MBPs.

Input: $\tilde{x}$ feasible solution
Parameters: k neighborhood size
1 Create a sub-MBP by adding the two constraints (4.2) and (4.3) to the original sub-MBP ;
2 Solve the sub-MBP ;

Algorithm 11: Outline of Local Branching

Intermezzo: Local Branching as Branching Strategy

If one prefers to use Local Branching as a branching strategy rather than as a heuristic, the according branching rule would be

$$\Delta^B(x, \tilde{x}) \leq k \textbf{ or } \Delta^B(x, \tilde{x}) \geq k + 1. \tag{4.4}$$

Indeed, this divides the solution space into two disjoint parts. The left one should be relatively small for well chosen parameter k and can therefore be examined by a MIP-solver in short time. It is the hope that it contains further solutions of better quality than the current incumbent $\tilde{x}$. The other only needs to be explored, if no improving solution can be found. Note that this yields an exact solution strategy.

Implementation Details

If you want to use Local Branching as an improvement heuristic, the time it consumes will be a crucial factor for the evaluation of its performance. One main problem which arises is that you have no reliable control whether the sub-MBP is actually much easier to solve than the original MBP is. That is why one should introduce some limit on the solving process of the sub-MBP, e.g., a time limit or a node limit. Another problem could be that the sub-MBP is too restricted, which should mean that it can be processed relatively fast but does not improve the incumbent solution, or does not drive an infeasible solution more towards feasibility, respectively.

Both issues can be handled by varying the value of the parameter k, which is responsible for the size of the neighborhood of $\tilde{x}$. For example, one could reduce k by $\lfloor \frac{k}{2} \rfloor$ if the solving process of the sub-MBP was interrupted due to some limit without having found a new solution and enlarge k by $\lceil \frac{k}{2} \rceil$, if the sub-MBP was completely solved, but did not contain a new best reference solution. Local Branching used as a heuristic would then restart its solving routine or one would postpone this to a later point of time in the Branch-And-Bound process, if no new incumbent was found up to then. It is obvious that one should not apply both of these mechanisms consecutively at one reference point $\tilde{x}$, because this would lead into a cycling process. Furthermore, one should not apply one of these diversification schemes too often at one point $\tilde{x}$, in order to not waste too much time for exploring a part of the MBP that is highly related to some where the Local Branching process already failed.

Fischetti and Lodi [25] proposed another strategy to deal with the case that the Local Branching process terminated within the limits, but without having found any improved solution even after the size of the neighborhood was increased. They suggest to search for a new solution in the vicinity of $\tilde{x}$ which needs not to improve the objective value. The local branching cut is updated to $1 \leq \Delta^B(x, \tilde{x}) \leq k + 2\lceil \frac{k}{2} \rceil$ and the objective cutoff is left out. The resulting MBP is solved and the solving process is stopped as soon as a feasible solution, probably of worse quality than $\tilde{x}$, has been found. This new solution is then taken as the new reference solution $\tilde{x}$, hoping that it has a neighborhood which can be better explored than the last one.

The above considerations result in Algorithm 12. In contrast to the one published by Fischetti and Lodi [25] this one is made to be used as a part of a MIP-solver, strictly concentrating on the improvement of an incumbent solution. It starts only, if a feasible solution is at hand, it sets limits on the subproblem size, restarts the solving process from the same reference point at most once, and it desists from the use of diversification schemes which lead to solutions of inferior quality.

Input: $\tilde{x}$ feasible solution
Parameters: nl node limit for the sub-MBP
k_0 initial neighborhood size

if *first call of Local Branching* **or** $\tilde{x}$ *has changed since last call* **then**
  exec $\leftarrow$ **true** ;
  k $\leftarrow k_0$;
else if exec = **false then**
  stop!
Set node limit nl for MIP-solver;
Solve the sub-MBP, which is obtained from the original MBP plus the two constraints (4.2) and (4.3);
if *new solution found* **then**
  stop!
if *node limit* nl *reached* **then**
  if k $= k_0$ **then**
    Set k $\leftarrow k_0 - \lfloor \frac{k_0}{2} \rfloor$ for next call;
  else
    Set exec $\leftarrow$ **false**;
else
  if k $= k_0$ **then**
    Set k $\leftarrow k_0 + \lceil \frac{k_0}{2} \rceil$ for next call;
  else
    Set exec $\leftarrow$ **false**;

Algorithm 12: Local Branching

Generalization to MIPs

Lodi [44] described how to generalize Local Branching to a LNS scheme to MIPs with general integer variables. This requires the introduction of some artificial variables to model the non-linear function $\Delta^I(x, \tilde{x})$ (see (4.2) and Section 3.1.2). Note that the resulting MIP would no longer be a sub-MIP of the original one. The task of finding a good solution for the original MIP can then be expressed as the following MIP:

$$
\begin{aligned}
\min \quad & \sum_{j \in I: \tilde{x}_j = l_j} (x_j - l_j) \; + \sum_{j \in I: \tilde{x}_j = u_j} (u_j - x_j) \; + \sum_{j \in I: l_j < \tilde{x}_j < u_j} (x_j^+ + x_j^-) \\
\text{such that} \quad & Ax \;\leq\; b \\
& x_j \;= \tilde{x}_j + x_j^+ - x_j^- \quad \text{for all } j \in I : l_j < \tilde{x}_j < u_j \\
& x^+ \;\geq 0 \\
& x^- \;\geq 0 \\
& l \;\leq\; x \;\leq u \\
& x_j \in \mathbb{Z} \quad \text{for all } j \in I \\
& x_j^+, x_j^- \in \mathbb{Z} \quad \text{for all } j \in I : l_j < \tilde{x}_j < u_j
\end{aligned}
\tag{4.5}
$$

4.2 RINS

The relaxation induced neighborhood search, shortly Rins, was first described by Danna, Rothberg, and Le Pape [21]. It is based on the observation that often the incumbent solution $\tilde{x}$ of a MIP and the optimum of the LP-relaxation $\bar{x}$ have a couple of variables set to the same values. As mentioned in Section 3.1.2, a "good" solution of a MIP has to fulfill three conditions: it has to be integer feasible, LP-feasible, and it should have a small objective value.

The Idea

The Objective Feasibility Pump described in Section 3.1.2 worked with two points in each iteration, one fulfilling the first condition, the other one the second and brought in the third by a little trick (see Definition (3.3)).

Rins also requires two points, one fulfilling the first two conditions and one fulfilling the last two. The incumbent MIP solution $\tilde{x}$ is integer feasible and LP-feasible. As well, at some node of the Branch-And-Bound-tree, the optimal solution $\bar{x}$ of the current LP is clearly LP-feasible and of an optimal objective value for the current subproblem.

Note that Rins needs an incumbent solution. This is different from Rens (see Section 3.2.1), which only took the optimum of the LP-relaxation into account, or Local Branching, where Fischetti and Lodi [26] described a general approach that could also be used to drive integer feasible, but LP-infeasible solutions towards feasibility.

If the current subproblem was not pruned by bounding, $c^T\bar{x} < c^T\tilde{x}$ holds. This implies $\bar{x} \neq \tilde{x}$, i.e., there are some variables for which $\tilde{x}$ and $\bar{x}$ take different values. If there is a sufficient number of variables, however, for which the values coincide, it is worth paying special attention to them. It seems likely that the values identical in an optimal LP-solution $\bar{x}$ and a feasible solution $\tilde{x}$ form a partial solution of good objective value.

Definition 4.2 *Let $\bar{x} \in P(A, b, l, u)$ be an optimal solution of the LP-relaxation at the current Branch-And-Bound-node, and let $\tilde{x} \in P(A, b, l, u)$ be the incumbent solution. The set*

$$\{y \in P(A, b, l, u) \mid y_j = \tilde{x}_j \text{ for all } (j \in I \text{ with } \tilde{x}_j = \bar{x}_j) \text{ and } y_j \in \mathbb{Z} \text{ for all } j \in I\}$$

is called the relaxation induced neighborhood *of $\tilde{x}$.*

This is also illustrated in Figure 4.2.

Implementation Details

Minimizing $c^T x$ over this set means to solve a sub-MIP of the original one, which concentrates on the variables differing in the two reference solutions. As usual for improvement heuristics, we are just interested in solutions that are better than the current incumbent. Thus, we add an objective cutoff $c^T x \leq (1-\epsilon)\, c^T\tilde{x}$ to the sub-MIP with $\epsilon > 0$. The resulting problem can be stated as:

$$\begin{aligned} \min \quad & c^T x \\ \text{such that} \quad & Ax \leq b \\ & c^T x \leq (1-\epsilon) c^T\tilde{x} \text{ for some } \epsilon > 0 \\ & x_j = \tilde{x}_j \text{ for all } j \in I \text{ with } \tilde{x}_j = \bar{x}_j \\ & l \leq x \leq u \\ & x_j \in \mathbb{Z} \text{ for all } j \in I \end{aligned} \tag{4.6}$$

As solving MIPs can be time consuming, one wants to avoid solving sub-MIPs which are very similar. One possibility would be to call RINS only if a new reference solution $\tilde{x}$ was found in the MIP-solving process, as for example our implementation of Local Branching does. The reason was, that the Local Branching sub-MIP only depended on the incumbent solution $\tilde{x}$ and on some fixed parameters.

Since the current optimal LP-solution $\bar{x}$ potentially changes at every branching node, RINS yields much larger diversification than Local Branching. However, since their LPs are very similar, consecutive nodes of the Branch-And-Bound-tree tend to have very similar optimal solutions. Hence the relaxation induced neighborhoods, if achieved from the same integer point $\tilde{x}$, will be strongly related. But if one takes an optimal LP-solution from another region of the Branch-And-Bound tree, it is likely that the RINS sub-MIP is fairly different, even if the same integer point $\tilde{x}$ is used.

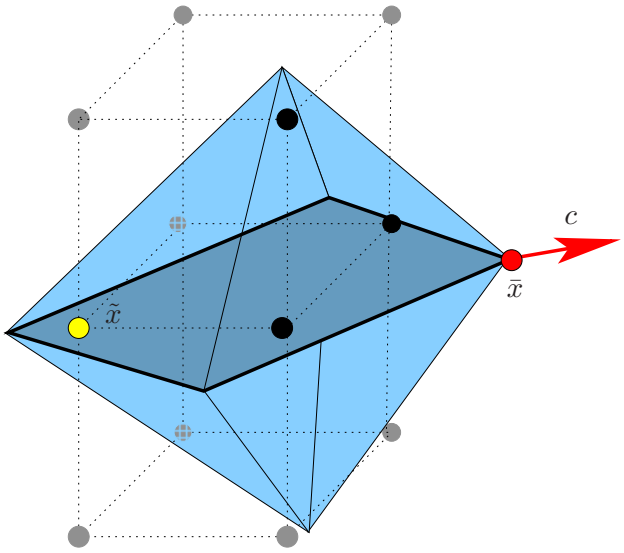


Figure 4.2. Fixing variables identical in $\bar{x}$ and $\tilde{x}$ implies a sub-MIP

One problem which is typical for procedures solving sub-MIPs also arises when analyzing RINS. Although the MIP (4.6) tends to be much smaller than the original one, it can still be difficult to solve. One way to control the difficulty of the sub-MIP is to call RINS only if a high enough percentage of variables can be fixed, i.e., if at least a certain number of variables takes the same value in the incumbent solution $\tilde{x}$ and the LP-solution $\bar{x}$. Moreover, one should introduce a node limit for the solving process of the sub-MIP. Putting all these ideas together, one obtains Algorithm 13.

Input: $\bar{x}$ optimal LP-solution
$\tilde{x}$ feasible solution
Parameters: nl node limit for the sub-MIP
mfr minimum ratio of integer variables to fix

1 Create sub-MIP (4.6) by fixing all variables x_j with $\tilde{x}_j = \bar{x}_j$ to $\tilde{x}_j$ and adding the cutoff constraint;
2 **if** *at least* mfr $\cdot |I|$ *integer variables were fixed* **then**
3 Set node limit nl for MIP-solver;
4 Solve MIP (4.6);
5 **else**
6 **stop!**

Algorithm 13: RINS

4.3 Crossover

If one already has a couple of solutions for a MIP at hand, these can be analyzed in order to construct further, better solutions. This section introduces an improvement heuristic, which relaxes variables with identical values in different feasible solutions.

RINS creates a sub-MIP by fixing variables with identical values in the incumbent solution and the optimum of the LP-relaxation of the current Branch-And-Bound-node. The idea of fixing variables which are identical in different, *partially* feasible solutions is a well-working improvement strategy and can easily be transformed into a heuristic operating with different feasible solutions.

The Idea

If one has more than one feasible solution at hand, comparing them and fixing variables with identical values will be a procedure, which promises to deliver another fruitful neighborhood.

Definition 4.3 *For $K = \{1, \ldots, \kappa\}$ let $\tilde{X} = \{\tilde{x}^k \mid k \in K\}$ be a set of $\kappa \in \mathbb{N}$ feasible solutions for a given MIP. The set*

$$\{y \in P(A, b, l, u) \mid y_j = \tilde{x}_j^1 \text{ for all } j \in I \text{ with } \tilde{x}_j^1 = \tilde{x}_j^k \text{ for all } k \in K \text{ and } y_j \in \mathbb{Z} \text{ for all } j \in I\}$$

is called the crossed neighborhood *of $\tilde{X}$.*

If you combine the Crossover heuristic with a mutation strategy (see Section 4.4), which diversifies single solutions, you get an evolutionary algorithm like the one Rothberg described in 2005 [52].

In terms of evolutionary algorithms you would call the elements of $\tilde{X}$ the *parents* of a solution possibly provided by Crossover, which itself would consequently be called the *offspring* of $\tilde{X}$.

Algorithm 14 and Figure 4.3 supply an impression of the fundamental proceeding of Crossover.

Input: S set of all MIP-solutions found so far
Parameters: κ number of solutions to involve
1 **if** $|S| \geq \kappa$ **then**
2 $\tilde{X} \leftarrow$ set of κ pairwise different elements of S;
3 Create a sub-MIP by fixing all integer variables x_j with $\tilde{x}_j^1 = \tilde{x}_j^k$ for all $k \in K$;
4 Solve the sub-MIP ;

Algorithm 14: Outline of Crossover

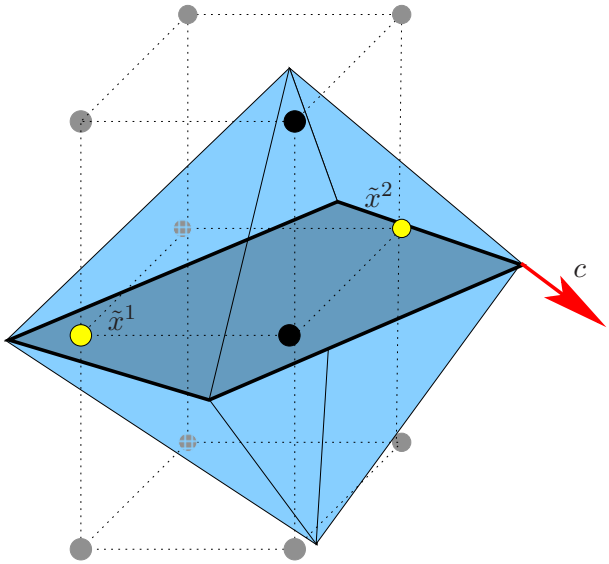


Figure 4.3. A sub-MIP implied by fixing one variable identical in two feasible solutions

Selection of Parent Solutions

The main issue on which the success or the failure of Crossover depends is the selection of the set $\tilde{X}$. One point of interest surely is the cardinality κ of $\tilde{X}$, the other one, how to choose its components $\tilde{x}^k$ from the pool of all feasible solutions found so far and how to make this depend on previous runs of Crossover.

Rothberg suggested to choose $\kappa = 2$ in the fashion of evolutionary algorithms, where it is usual to combine an offspring of exactly two parents. In general, every $\kappa > 1$ would be possible, but there is a tradeoff one has to keep in mind. The bigger κ, the greater is the chance that the fixings that Crossover produces are indispensable for feasible solutions and that Crossover does not coincidently fix variables to a misleading value. The smaller κ, the greater is the chance that a lot of variables get fixed and the resulting sub-MIP is easy to solve. Regrettably, both behaviors are desirable, but contrary.

The other point is how to select the set $\tilde{X}$ among all feasible solutions found so far. We could just take the best κ solutions concerning their objective value. Another method which was suggested by Rothberg [52] is to randomize the selection.

Generalizing his method for more than two reference solutions, it would work as follows: sort the solutions by nondecreasing objective value, then randomly choose a solution between the κ-th and the L-th position, where $L \geq \kappa$ is the size of the solution pool. Let the position of the chosen solution

be $t \geq \kappa$. Now iterate this process by randomly choosing a solution between the (κ-1)-th position and the t-th position. For $L > \kappa$, one obtains a randomized set $\tilde{X}$ of κ solutions. This randomization process tends to choose solutions with good objective value as in every iteration it takes only such solutions into account which are at least as good as the one chosen in the previous iteration.

Naturally, there is the possibility of combining these two selection methods. We suggest to call Crossover with the criterion to select the best κ solutions, if this set changes, hence every time a solution being among the κ best ones so far is found. In the meantime Crossover performs the selection in a randomized fashion.

As soon as we have a promising set of solutions $\tilde{X}$ with index set K which should be crossed, the remainder of the procedure is as follows. Crossover proceeds analogously to the RINS heuristic of Section 4.2, by solving a sub-MIP which consists of the original MIP extended by an objective cutoff and the constraint

$$x_j = \tilde{x}_j^1 \text{ for all } j \in I \text{ with } \tilde{x}_j^1 = \tilde{x}_j^k \text{ for all } k \in K. \tag{4.7}$$

Implementation Details

As Crossover is a time consuming heuristic like RINS and Local Branching, one wants to avoid solving the same sub-MIP more than once. This can happen when you call Crossover twice using the same solution set $\tilde{X}$.

Furthermore we want to prevent a behavior of memorizing special assignments for variables. For example, consider the following situation: Crossover is called with a solution set $\tilde{X}'$, which is the same like the set $\tilde{X}$ in a prior call with just one parent solution exchanged by the offspring solution Crossover found in this prior call.

Now every variable fixed in the prior call will be fixed again in the current call. Hence the current sub-MIP will itself be a sub-MIP of the old one. If the old one was solved to optimality, our new call will be redundant as it cannot deliver any better solution.

One method to avoid such redundant calls is to keep some kind of tabu list (see, e.g., [31]) containing all combinations of solutions which have been used in earlier calls of Crossover and all combinations of a solution found by Crossover with $\kappa - 1$ of its parents. Algorithm 15 describes Crossover with a combined solution selection strategy and a tabu list of likely fruitless solution sets.

4.4 Mutation

In the preceding section, we described an improvement strategy called Crossover, which resembled the combination step in Rothberg's evolutionary algorithm [52] in its idea, but differed in its details. The algorithm Rothberg

Input: S set of all MIP-solutions found so far
Parameters: nl node limit for the sub-MIP
mfr minimum ratio of integer variables to fix
κ number of solutions to involve

if *first call of Crossover in current MIP solving process* **then**
  Initialize tabu list $T \leftarrow \emptyset$;
if $|S| < \kappa$ **then**
  stop!
Sort S by nondescending objective value;
Initialize $\tilde{X} \leftarrow \emptyset$;
if $|S| = \kappa$ **or** *new solution among the first* κ *elements of* S **then**
  $\tilde{X} \leftarrow$ first κ elements of S;
else
  $k \leftarrow |S| + 1$;
  for $j \leftarrow 0$ **to** $\kappa - 1$ **do**
    $k \leftarrow$ randomized number in $\{\kappa - j, \ldots, k - 1\}$;
    $\tilde{X} \leftarrow \tilde{X} \cup \{\tilde{x}^k\}$, where $\tilde{x}^k$ is the k-th element of S;
  if $\tilde{X} \in T$ **then**
    goto step 10; `/* repeat at most 100 times, then stop */`
$T \leftarrow T \cup \{\tilde{X}\}$;
if *at least* mfr $\cdot |I|$ *of the integer variables are identical for all* $\tilde{x}^k \in \tilde{X}$ **then**
  Set node limit nl for MIP solver;
  Solve the sub-MIP which one gains by adding an objective cutoff and the constraints (4.7) to the original MIP ;
  if *new solution* $\tilde{x}$ *found by Crossover* **then**
    for $\tilde{x}^k \in \tilde{X}$ **do**
      $T \leftarrow T \cup \{\tilde{X} \backslash \{\tilde{x}^k\} \cup \{\tilde{x}\}\}$;

Algorithm 15: Crossover

published in 2005 consisted of some selection step, the combination step and another one, called the mutation step. This one will be presented in this section. As both steps could be performed independently, we will treat them as independent LNS heuristics, and call the mutation step from now on just Mutation.

If you want to use the evolutionary algorithm as a polishing method which should improve the $\tilde{x}$ solution after the run of the Branch-And-Bound-process (like Rothberg [52]), Mutation is necessary in order to get enough diversity into your solution pool. But if the Branch-And-Bound-framework itself keeps a pool of all (or at least a high number of) solutions it found, Mutation is not longer needed for this special issue, but nevertheless it could be a useful improvement heuristic.

The idea is quite simple and should thus be just sketched here: first of all, a solution $\tilde{x}$ to be mutated is determined. This could be the incumbent or any other feasible solution at hand. Secondly, a random set Y of say υ variables is chosen among all integer variables. Third, a sub-MIP is created which consists of the original MIP plus an objective cutoff and the fixing constraints $x_j = \tilde{x}_j$ for all $x_j \in Y$. Fourth, the MIP-solver tries to solve the sub-MIP within a certain time or node limit. If a feasible solution is found, it is also feasible for the original MIP and of better quality. This simple scheme is illustrated in Algorithm 16 once again.

The crucial parameter that balances a fast running time and enough freedom to achieve a feasible sub-MIP clearly is υ, the fraction of variables which should be fixed.

Input: $\tilde{x}$ feasible solution for the MIP
Parameters: υ ratio of variables to fix
nl node limit for the sub-MIP

1 $\tilde{x} \leftarrow$ incumbent solution of MIP ;
2 Create sub-MIP by fixing a random set of υ variables x_j to $x_j = \tilde{x}_j$ and adding an objective cutoff;
3 Set node limit nl for MIP-solver;
4 Solve sub-MIP ;

Algorithm 16: Mutation

4.5 Computational Results

We tested the four LNS improvement heuristics on our two standard test sets. Therefrom we made one SCIP-run without any LNS improvement heuristics and four runs, where one of the four was activated in each case. In order to keep this section clearly arranged, we will from now on call SCIP with all four LNS improvement heuristics deactivated SCIP-NO. SCIP with Local Branching activated and the other LNS improvement heuristics deactivated

Criterion	No LNS	Local Branching	Rins	Crossover	Mutation
Fewest Solving Nodes	5	5	10	**18**	6
Nodes Geom. Mean	3579	3581	3473	**3404**	3535
Heuristic Time Geom. Mean	–	3.1	2.1	**1.5**	1.6
Solving Time Geom. Mean	57.9	60.9	57.2	**56.0**	57.8
Smallest P Gap (hard test set)	3	5	7	6	**9**
Heuristic Time Geom. Mean (hard)	–	33.9	15.8	**4.6**	12.7

Table 4.1. Summarized performance of SCIP with different LNS heuristics

will be called SCIP-LB, SCIP with Rins activated SCIP-RI, SCIP with Crossover activated SCIP-CO and with Mutation SCIP-MU.

For the four versions where one improvement heuristic is activated, the particular heuristic was called at depth 10 of the Branch-And-Bound-tree and then at every 30th depth. The minimum improvement factor ϵ for the objective cutoff was always set to 0.01, and a node limit of 5000 nodes was set for solving the sub-MIP.

A node delay of 200 was set. This means that the heuristics will not be called until at least 200 nodes of the Branch-And-Bound-tree were processed since the last change of the incumbent. This avoids calling an expensive heuristic which solves a sub-MIP during a phase of the solving process where new incumbents are frequently produced by cheaper heuristics. One could also say that improvement heuristics "should wait for a good incumbent". This gave us the reason to remove all 40 instances from the test which can be solved to optimality by SCIP with default settings in less than 200 nodes.

We set the specific parameters of the heuristics as follows. For Local Branching, we chose $k = 18$, as recommended in the original description of Fischetti and Lodi [25]. For Rins, we decided to set $mfr = 0$, as this yielded the best results in our tests and no minimum fixing rate was given by Danna, Rothberg and Le Pape [21]. For Crossover we set $\kappa = 3$ and $mfr = \frac{2}{3}$. For Mutation we chose $\upsilon = 0.8$.

The results are shown in Tables B.11 and B.12 and summarized in Table 4.1.

Among the 57 easy instances which needed more than 200 solving nodes to get solved to optimality by SCIP with default settings, there are 32 instances for which the number of solving nodes differs among the 5 versions. Among these, SCIP-NO was a version with fewest solving nodes 5 times, SCIP-LB 5 times, SCIP-RI 10 times, SCIP-CO 18 times, SCIP-MU 6 times.

In the geometric mean taken over 57 instances, SCIP-NO needs 3579 solving nodes, SCIP-LB 3581, SCIP-RI 3473, SCIP-CO 3404, and SCIP-MU 3535. In the geometric mean, SCIP-NO needs 57.9 seconds to find an optimal solution and prove ist optimality, SCIP-LB needs 60.9 seconds, SCIP-RI 57.2 seconds, SCIP-CO 56.0 seconds, SCIP-MU 57.8. From these times, Local Branching took 3.1 seconds, Rins 2.1, Crossover 1.5 and Mutation 1.6 seconds running time in the geometric mean.

Note, that SCIP-CO was the only variant which could solve `neos3` to

optimality within the time limit of one hour.

Next, we examine the results for the hard instances. SCIP with default settings did not find a feasible solution within one hour for 6 of the hard instances, so none of the improvement heuristics was called for them. Furthermore, there are 3 instances, namely `noswot`, `opt1217`, and `t1717` for which all 5 variants finished their run with the same primal bound.

SCIP-NO found a solution smallest among the 5 variants 3 times, SCIP-LB 5 times, SCIP-RI 7 times, SCIP-CO 6 times, and SCIP-MU 9 times. For the primal-dual gap γ_{PD} similar results hold. In the geomtric mean, the heuristic running time was 33.9 seconds for Local Branching, 15.8 seconds for RINS, 4.6 seconds for Crossover and 12.7 seconds for Mutation.

Concluding, the use of a LNS heuristic, which constructs a sub-MIP by fixing variables is a promising approach: RINS, Crossover, Mutation all improve the performance of SCIP. The Local Branching approach of adding an extra constraint, but leaving all variables free, does not seem to work well. It slowed down the computation and increased the number of solving nodes.

Especially due to the performance on the easy test set we decided to integrate Crossover into SCIP as default LNS heuristic. It decreased the absolute running time by 9 %, the averaged running time by 3.3 %, the absolute number of solving nodes by 11 %, the averaged number of solving nodes by 4.9 %.

Regrettably, some further tests we did, showed that integrating more than one LNS heuristic into SCIP produced inferior results. Nevertheless, it would be desirable to combine the advantages of different LNS heuristics since there was a couple of instances for which RINS or Mutation yielded strictly better results than Crossover did. This could be an area of further research.

Chapter 5

Results

Throughout this thesis we saw that most of the presented heuristics perform well in finding or improving feasible solutions for the instances of our test sets. Only for a few of them we tested how they influence the overall performance of the Branch-And-Bound framework in which they are embedded. This will be done in the first section of this chapter. In the second section we will summarize the insights we have won throughout this thesis.

5.1 Impact of the SCIP Heuristics

According to the work of Bixby, Fenelon, Gu, Rothberg, and Wunderling [17], we want to investigate, how the performance of SCIP changes if switching off any one of the heuristics which are activated by default.

Bixby et al. chose a test set of instances which could be solved to optimality by Cplex 6.5 within a time limit of twelve hours. Similarly, we compare the performance on all instances of our test set which can be solved to optimality by SCIP 0.82b with default settings and a time limit of one hour. This is exactly our easy test set which can be seen in Table 2.1. Furthermore, we compare the performance on instances which cannot be solved to optimality within one hour. This is our hard test set, see Table 2.2.

We treated all heuristics equally without dividing them into classes, since those tests have been made in the preceding chapters.

Description of the Tests

First, we ran SCIP with default settings first, at next we made 15 runs with one heuristic switched off at a time, and last we tested SCIP without any heuristic at all.

Tables B.13, B.14, and B.15 show the results of these runs, Table 5.1 summarizes them. In Tables B.13, B.14 and B.15, the first column shows the names of the instances, further columns stand for the heuristics which were switched off.

Definition 5.1 *Let t be the running time* SCIP *needed to solve a specific instance with some settings and t_0 the running time* SCIP *needed to solve the same instance with default settings. We call $\frac{t+1}{t_0+1}$ the* relative running time *of the instance with respect to this settings. The* relative number of solving nodes *and the* relative primal-dual gap *are defined in a similar way.*

Table B.13 shows the relative running times, and Table B.14 shows the relative number of solving nodes. For the easy test set, the symbol " $\geq$" indicates, that the instance could not be solved to optimality if SCIP is called with these settings. Table B.15 shows the relative primal-dual achieved within the time limit using the specific settings. A bar "–" indicates that no solution could be found with these settings or the primal-dual gap with default settings is infinity. The symbol ∞ shows that a solution was found with these settings, but none using the default settings.

All values which indicate a difference by 20% or more from the default value have been marked. Values larger than or equal to 1.2 are written in a bold face, values smaller than or equal to 0.8 are written in italics.

In Table 5.1, all entries which indicate that disabling the according heuristic leads to an inferior performance are written in a green, bold face. All entries which indicate that disabling the according heuristic leads to a superior performance are written in red italics. In other words, a couple of green, bold entries in a column shows that using this heuristic seems reasonable, a bundle of red entries shows that the heuristic should not be used.

We used the following abbreviations for the heuristics: **Sim**ple**Rou**nding, **Round**ing, **Shift**ing, **Feas**ibility**P**ump, **Coef**ficient**Di**ving, **Ps**eudo**Co**st-**Di**ving, **Frac**tionality**Di**ving, **Vec**torlength**Di**ving, **O**bjective**Ps**eudocost-**Di**ving, **Roo**t**So**lution**Di**ving, **Line**search**Di**ving, **Gui**ded**Di**ving, **Octane**, **Cr**oss**Ov**er, **Rens**. **ALL** stands for all heuristics deactivated. **Diving** means that all diving heuristics were deactivated.

Rootsolution Diving and Objective Pseudocost Diving are two diving heuristics which were not presented in this thesis. They work similarly to the Feasibility Pump since they do not bound any variables during diving, but vary the objective function.

Rootsolution Diving determines the new objective based on the difference between the current LP solution and the optimum of the LP-relaxation, as it was done in Linesearch Diving when choosing variables for bounding. Objective Pseudocost Diving uses the pseudocosts of the variables. This resembles the strategy of Pseudocost Diving.

Computational Results for the Easy Instances

In the following, we will discuss the influence of deactivating a single heuristic.

SCIP with default settings was able to solve 97 instances to optimality within a time limit of one hour. Deactivating OCTANE or RENS did not change this number. For all other heuristics, there exists at least one instance which was no longer solved to optimality if this heuristic was deactivated.

Criterion	SimRou	Round	Shift	FeasP	CoefDi	PsCstDi	FracDi	VecDi	OPsDi
Instances Solved	**96**	**96**	**96**	**95**	**96**	**96**	**95**	**95**	**95**
Geom. Mean Time	**1.0436**	**1.0440**	*0.9735*	**1.1231**	*0.9945*	**1.0403**	**1.0639**	**1.0309**	*0.9983*
≥ 20% Variation Time	**3:0**	**3:0**	*2:3*	**21:5**	*5:8*	**7:5**	**14:2**	**7:3**	**7:4**
Geom. Mean Node	**1.0001**	**1.0470**	*0.9912*	**1.3035**	*0.9550*	*0.9839*	*0.9991*	*0.9696*	*0.9933*
≥ 20% Variation Nodes	**1:0**	**3:0**	*2:3*	**11:4**	8:8	**10:7**	**9:4**	*5:8*	**9:4**
Solutions Found	26	26	26	**25**	*27*	*27*	**25**	*29*	*27*
Geom. Mean P-D Gap	**1.0040**	**1.0034**	**1.0012**	*0.9112*	*0.9357*	*0.9025*	*0.9779*	*0.9930*	*0.9602*
≥ 20% Variation Gap	0:0	0:0	0:0	*0:5*	2:2	*3:6*	*1:4*	1:1	*4:5*

Name	RooSoDi	LineDi	GuiDi	Octane	CrOv	Rens	ALL	Diving
Instances Solved	**95**	**95**	**96**	97	**96**	97	**90**	–
Geom. Mean Time	**1.0214**	**1.0157**	**1.0114**	*0.9787*	**1.0229**	**1.0223**	**2.0183**	–
≥ 20% Variation Time	**7:0**	**7:3**	**5:2**	*1:4*	**6:2**	**10:2**	**86:2**	–
Geom. Mean Node	**1.0311**	**1.0041**	**1.0241**	**1.0037**	**1.0365**	**1.1453**	**1.9912**	–
≥ 20% Variation Nodes	**10:2**	**6:5**	**5:1**	**1:0**	**9:1**	**12:0**	**49:5**	–
Solutions Found	26	26	26	26	26	26	**17**	**23**
Geom. Mean P-D Gap	*0.9700*	*0.9497*	*0.9979*	**1.0042**	**1.0220**	**1.0004**	**1.4666**	**1.2558**
≥ 20% Variation Gap	*0:2*	*2:4*	*0:1*	0:0	**1:0**	2:2	**7:1**	**11:2**

Table 5.1. Summarized results: switch off one heuristic each time, switch off all heuristics, and switch off all divers for the hard test set

There are four heuristics, namely Shifting, Coefficient Diving, Objective Pseudocost Diving and Octane, for which the geometric mean of the relative running time will decrease if they are deactivated. If we consider the "extreme cases" where the solving time differs by at least 20% compared to the default, we will observe that switching off Shifting, Coefficient Diving, or Octane more often leads to a decrease of at least 20% than to an increase. This is not the case for Objective Pseudocost Diving, although the geometric mean of the relative running time decreases.

For six heuristics, switching them off yields a decrease of the geometric mean of the number of solving nodes. But only for two of them, namely Shifting and Vectorlength Diving, there are also more instances for which the number of solving nodes decreased by at least 20% than there are instances for which the number of solving nodes increased by at least 20%.

Altogether, deactivating Shifting and Coefficient Diving seems reasonable, since more criteria argue for disabling them than vice versa. Switching off Octane yields an improvement in the solving time and only causes slight changes in the number of solving nodes. Hence, we would recommend to deactivate these three heuristics in further versions of SCIP.

If Vectorlength Diving is switched off, this will cause an decrease of the relative running time, but an increase of the relative number of solving nodes. Similarly, switching off Objective Pseudocost Diving causes improvements in the geometric mean of solving time and solving nodes, but the "20% statistics" argue for not disabling it. If deactivating one of the two heuristics, there will be two instances which cannot be solved to optimality any longer. Therefore, we decided to let them activated in the default settings of the next SCIP version.

For Pseudocost Diving and Fractional Diving we have observed a slight improvement in the geometric mean of the number of solving nodes if switching them off. But the increase of the running time, the fact that not all easy instances can be solved to optimality and the "20% statistics" give reason to let these heuristics activated. Deactivating Simple Rounding, Rounding, the Feasibility Pump, Rootsolution Diving, Linesearch Diving, Guided Diving, Crossover, or RENS does not cause an improvement of any of the five criteria we have considered. For all of them we observe that the performance gets worse if disabling them. Hence, on average they are useful heuristics.

If we compare the results of SCIP with default settings to SCIP without any heuristic, we will see that the latter one is clearly inferior in all criteria we have considered. There are seven instances which could no longer be solved to optimality within one hour after we had deactivated all heuristics. The geometric mean of the running time doubled. In 86 out of 97 cases, SCIP, having all heuristics deactivated, was at least 20% slower than SCIP with default settings, only twice the opposite held. The geometric mean of the number of solving nodes doubled, too. In 49 cases the number increased by at least 20% when deactivating the heuristics, 5 times it decreased by at least 20%. These statistics indicate that the heuristics have a positive impact on the performance of SCIP.

Computational Results for the Hard Instances

Next, we will analyze the impact of the heuristics on those instances which could not be solved to optimality by SCIP with default settings and a time limit of one hour. We made the same tests like before: we compared SCIP with default settings to SCIP with one of the heuristics deactivated and to SCIP without any heuristics.

Table 5.1 shows that deactivating Simple Rounding, Rounding, Shifting, OCTANE, Crossover or RENS does not cause a change in the number of instances for which a feasible solution could be found, it stays 26 as well as for SCIP with default settings. Furthermore, the geometric mean of the relative primal-dual gap will increase if one switches off any of these six heuristics. Deactivating Crossover causes a change of 2.2%, whereas for the others the difference was only marginal, below 0.5%. Disabling Simple Rounding, Rounding, Crossover or RENS yielded an impairment for the easy instances, too. This substantiates the statement that they should be used in SCIP in order to achieve a good performance.

Since the geometric mean of the relative primal-dual gap only changes slightly if Shifting or OCTANE are deactivated, we still recommend to deactivate them.

The results of testing the diving heuristics on the hard instances was surprising. If we switch any of the divers off, the geometric mean of the relative primal-dual gap always gets smaller. The change is in a range from 0.2% to 10%. Additionally, for seven of nine diving heuristics, there

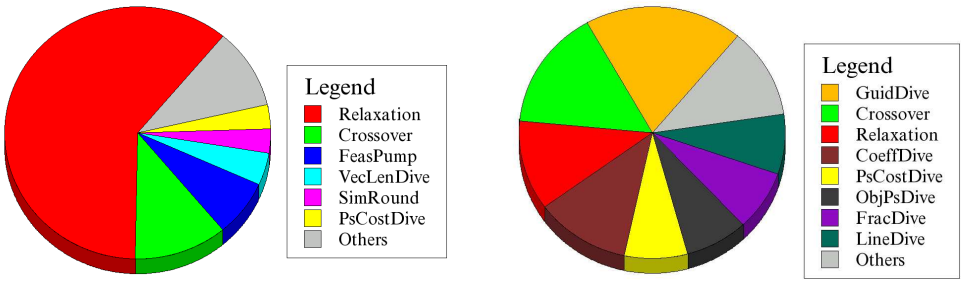


Figure 5.1. Distribution of heuristics that found the optimal or best solution, left: easy instances, right: hard instances

are more instances for which disabling them causes an increase of at least 20% of the relative primal-dual gap than there are instances for which this causes a decrease of at least 20%. Furthermore, if we switch off either Coefficient Diving, Pseudocost Diving or Objective Pseudocost Diving, we will find a feasible solution for 27 instances within the time limit. If we switch off Vectorlength Diving, we will find a feasible solution for 29 instances, compared to 26 if we use SCIP with default settings. If we switch off the Feasibility Pump or Fractional Diving, this number reduces to 25.

Altogether, deactivating either of the diving heuristics seems to improve the performance of SCIP on the hard instances. According to the results above, one could come to the conclusion that diving heuristics are not useful on difficult or large instances at all. The next test will demonstrate that this is not the case.

We applied SCIP with all diving heuristics deactivated to every instance of the hard test set. The last column of Table B.15 shows the relative primal-dual gaps we got and the last column of Table 5.1 summarizes them.

SCIP without any diving heuristic was inferior to SCIP with default settings. The number of instances for which a solution could be found was reduced by three. The geometric mean of the relative primal-dual gap increased by more than 25%. There were 11 instances for which we got an at least 20% increase of the primal-dual gap, but only two for which we got an at least 20% decrease. We conclude that the diving heuristics should not be completely deactivated, but their current settings, and the way they are combined in SCIP should be modified.

Further Results

We saw that SCIP without any heuristics is clearly inferior to SCIP with the heuristics as they are set in default. After we had disabled the heuristics, the geometric mean of the running time and the number of solving nodes, needed for the easy test set, doubled. There were 9 hard instances for which no solution could be found any longer and 7 easy instances for which no optimal solution could be found and proven.

When we analyzed our test data, we recognized that for the easy test set

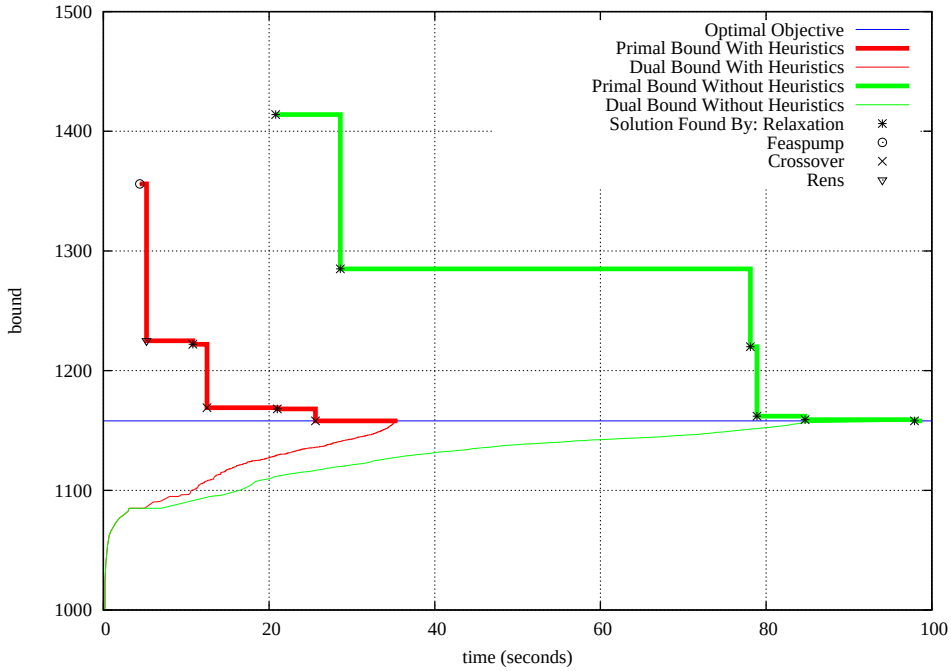


Figure 5.2. Development of the primal and dual bound, if SCIP processes instance `aflow30a`, with default settings (red) and without any heuristic (green)

in 59 out of 97 cases the optimal solution has been found by the relaxation of some Branch-And-Bound-node. Only in 38 cases a heuristic found the optimal solution. Crossover and the Feasibility Pump were the most successful heuristics. The former found 11 optimal solutions and the latter one found 7 optimal solutions.

Controversially, 23 of the 26 incumbent (after one hour) solutions of the hard instances have been found by a heuristic. The most successful was Guided Diving which found the best primal bound 5 times, Crossover succeeded 4 times, Coefficient Diving and the relaxation 3 times. Figure 5.1 demonstrates these distributions.

The tests of Chapter 3 showed us that in many cases heuristics already find feasible solutions at the root node. In Table B.13 we see that the running time reduced by at least 20% in nearly every case (86 out of 97) if we used heuristics, but in 60% of the cases the optimal solution was found by the relaxation. This shows us that the effect of heuristics is not mainly finding an optimal solution, but finding solutions in early steps of the solving process. This results in pruning suboptimal branches of the Branch-And-Bound-tree. Hence, SCIP processes the part of the Branch-And-Bound-tree containing optimal solutions earlier and therefore, it is able to terminate after a shorter period of time.

As an example, we discuss the solving process of the instance `aflow30a` which is part of the easy test set. If we solve this instance by using SCIP with default settings and SCIP with all heuristic deactivated, the differences

between the two solving processes are quite typical. Figure 5.2 visualizes the development of the primal and dual bound if we call SCIP for `aflow30a`. The red line indicates the development, if we use SCIP with default settings, while the green line shows the development, if we deactivate all primal heuristics. We recognize that heuristics – in this case the Feasibility Pump, Rens, and Crossover– find feasible solutions much earlier than the Branch-And-Bound-process on itself. With activated heuristics, solutions found by the relaxation occur in earlier steps, and the dual bound increases faster. Altogether, the solving process terminates within nearly a third of the time SCIP without heuristics needs.

5.2 Conclusions

We described five primal heuristics taken from the literature of the last ten years in this thesis: the Feasibility Pump, Octane, Local Branching, Rins, and Mutation. We presented special implementations of the general heuristic ideas of rounding and diving. Furthermore, we introduced two new heuristics, Rens and Crossover. Thereby, the last one was parallelly developed by Rothberg [52]. We presented an improvement of the Feasibility Pump, which we also published in [3] and we supplemented a new, well working ray selection rule to Octane.

We integrated all described heuristics into SCIP and showed that they have a big impact on the solving process. The default settings of SCIP will be modified according to the results of this thesis.

When we experimented with LNS heuristics and when we tested the diving heuristics on hard instances, we recognized that combining different heuristics of the same type can cause difficulties. This could be an area of further research.

Appendix A

Notation

A	a real-valued $(m \times n)$-matrix
b	a real-valued m-dimensional vector
c	the objective function vector of a MIP
l	the lower bound vector of a MIP
u	the upper bound vector of a MIP
N	the index set of all variables of a MIP
I	the index set of all integer variables of a MIP
B	the index set of all binary variables of a MIP
$P(A, b, l, u)$	the polyhedron defined by $Ax \leq b$, $l \leq x \leq u$
$\bar{x}$	an LP-feasible vector
$\bar{X}$	a set of LP-feasible vectors
$\tilde{x}$	an integer feasible vector
$\tilde{X}$	a set of integer feasible vectors
$x^\star$	an optimal or best known feasible solution
$\mathring{x}$	an optimal solution of the LP-relaxation
$f(x)$	the fractionality of x
$\gamma_P(x)$	gap of x to optimal or best known solution (in %)
$\gamma_{PD}(x)$	gap of x to dual bound (in %)

Appendix B

Tables

In this section, we present the results of our test runs in detail. The tables are all organized as follows.

Rows:

- The head row names the heuristics or settings which were tested, and the criteria which are shown in the columns.
- Every row lists the data of one specific instance from our test sets.
- The tail row shows the overall time, nodes, and/or heuristic time the solving took and the geometric mean taken over the number of instances in the test set.

Columns:

- "Name" shows you the name of the instance, this is always the first column.
- "Primal Bound" presents the objective value of the best solution, the heuristic found, or "–" if no feasible solution could be found.
- "P Gap" or "γ_P" gives the primal bound's primal gap in percent. A bar "–" means that no feasible solution could be found or the gap is infinity, "Large" means that the primal gap is above 10'000%.
- "Nodes" shows the number of Branch-And-Bound nodes SCIP processed until a problem was solved or the solving process was aborted due to some limit.
- "Time" presents the overall running time of SCIP in seconds until some time, memory or node limit is reached.
- "HeurTime" gives the running time the heuristic (as a SCIP plugin) took, "PresTime" gives you the time SCIP needs for presolving and solving the root LP.

A horizontal line between two rows separates the easy and the hard test set.

In Table B.2, Column "Time" shows the overall running time of the Feasibility Pump code. The geometric mean of the primal gap is taken over all instances for which the first five versions all found a solution. The version shown in the last column is excluded, since there were not enough instances for which it could find a feasible solution.

In Table B.3, Columns "FixInt" and "FixAll" show the percentages of integer and of all variables that could be fixed.

	Coefficient Diving			Fractional Diving			Vectorlength Diving		
Name	Primal Bound	P Gap	HeurTime	Primal Bound	P Gap	HeurTime	Primal Bound	P Gap	HeurTime
10teams	–	–	3.5	–	–	3.7	–	–	33.9
30:70:4_5:0_5:100	18	100.0	24.7	21	133.3	50.6	214	2277.8	38.9
30:70:4_5:0_95:98	13	8.3	32.8	18	50.0	57.7	155	1191.7	43.4
30:70:4_5:0_95:100	3	0.0	27.4	5	66.7	52.3	70	2233.3	39.2
acc-0	–	–	2.5	–	–	2.2	0	0.0	2.4
acc-1	–	–	3.5	–	–	3.2	0	0.0	8.2
acc-2	–	–	5.5	–	–	4.2	0	0.0	9.6
acc-3	–	–	18.8	–	–	10.2	–	–	38.1
acc-4	–	–	13.5	–	–	8.9	–	–	28.5
acc-5	–	–	6.2	–	–	4.5	–	–	16.2
acc-6	–	–	6.4	–	–	4.4	–	–	21.6
aflow30a	–	–	0.1	–	–	0.4	1267	9.4	1.6
air03	368644	8.4	0.6	343026	0.8	0.2	342998	0.8	3.5
air04	57971	3.3	10.4	–	–	9.5	–	–	93.8
air05	–	–	6.3	–	–	5.1	27276	3.4	9.0
bc1	–	–	2.3	3.4867846	4.4	2.0	3.4198842	2.4	3.7
bell3a	880414.281	0.2	0.1	880414.281	0.2	0.0	880414.281	0.2	0.0
bell5	–	–	0.0	–	–	0.0	9787786.46	9.2	0.1
bienst1	67.6666667	44.7	0.1	51.3333333	9.8	0.1	61.5	31.6	1.4
bienst2	74	35.5	0.1	73.3333333	34.3	0.2	61.6666667	12.9	2.1
blend2	–	–	0.0	–	–	0.0	235.593725	3000.3	1.4
cap6000	-2451215	0.0	0.6	-2450472	0.0	0.3	-2443599	0.3	1.8
dano3_3	576.804034	0.1	6.4	576.396385	0.0	7.4	–	–	45.9
dano3_4	578.708155	0.4	12.8	576.435225	0.0	11.3	–	–	44.1
dano3_5	579.838661	0.5	13.8	578.648269	0.3	17.8	–	–	48.6
disctom	–	–	14.1	–	–	14.4	-5000	0.0	72.2
dcmulti	–	–	0.1	–	–	0.1	–	–	0.0
dsbmip	-304.904815	0.1	0.0	-304.904815	0.1	0.0	-305.198175	0.0	0.1
egout	568.1007	0.0	0.0	579.718806	2.0	0.0	572.23465	0.7	0.0
eilD76	1536.78149	73.6	0.1	1467.63093	65.8	0.1	979.945566	10.7	2.9
enigma	–	–	0.0	–	–	0.0	–	–	0.2
fast0507	457	162.6	61.3	460	164.4	41.3	178	2.3	94.3
fiber	505640.95	24.6	0.2	448168.15	10.4	0.1	419895.5	3.4	1.1

continue next page

	Coefficient Diving			Fractional Diving			Vectorlength Diving		
Name	Primal Bound	P Gap	HeurTime	Primal Bound	P Gap	HeurTime	Primal Bound	P Gap	HeurTime
fixnet6	4240	6.5	0.1	5665.18198	42.2	0.1	3986	0.1	0.7
flugpl	–	–	0.0	–	–	0.0	–	–	0.2
gesa2-o	–	–	0.1	–	–	0.1	44681219.7	73.3	1.3
gesa2	25963975.9	0.7	0.1	25790902.5	0.0	0.1	27379074.5	6.2	1.5
gesa3	28528957.9	1.9	0.1	–	–	0.1	29316624.2	4.7	0.1
gesa3_o	28429843	1.6	0.1	27991042.6	0.0	0.1	47352002.7	69.2	0.7
gt2	30306	43.2	0.0	30306	43.2	0.0	49561	134.2	0.0
irp	14460.5068	18.9	0.3	12187.721	0.2	0.3	12295.6729	1.1	10.4
khb05250	106940226	0.0	0.0	111430400	4.2	0.0	106940226	0.0	0.2
l152lav	4957	5.0	0.2	–	–	0.9	4790	1.4	1.4
lseu	1748	56.1	0.0	–	–	0.0	1148	2.5	0.1
mas74	14372.8711	21.8	0.0	19189.9325	62.6	0.0	13755.8924	16.6	0.3
mas76	45879.978	14.7	0.0	42907.2181	7.3	0.0	40560.0518	1.4	0.1
mas284	97070.4637	6.2	0.0	97462.7524	6.6	0.1	94380.8188	3.3	0.5
misc03	–	–	0.0	–	–	0.0	4765	41.8	0.4
misc06	12857.3473	0.1	0.0	12869.2349	0.1	0.1	12881.2224	0.2	0.2
misc07	–	–	0.0	3020	7.5	0.0	4760	69.4	0.7
mitre	115155	0.0	1.0	115170	0.0	0.5	115185	0.0	0.1
mod008	418	36.2	0.1	456	48.5	0.1	307	0.0	0.1
mod011	-53664992.4	1.6	1.4	-53664992.4	1.6	1.4	–	–	9.0
modglob	21032193	1.4	0.0	21032640.2	1.4	0.0	20930259.7	0.9	1.3
mzzv11	–	–	17.6	–	–	18.1	–	–	15.5
mzzv42z	-20220	1.6	19.0	-20290	1.2	16.5	–	–	10.8
neos1	20	5.3	0.2	20	5.3	0.1	19	0.0	0.5
neos2	–	–	0.0	–	–	0.8	–	–	0.1
neos3	–	–	0.0	–	–	0.9	–	–	1.6
neos5	–	–	0.5	–	–	0.9	–	–	0.1
neos6	–	–	8.3	–	–	6.4	100	20.5	6.2
neos7	744934	3.2	0.1	–	–	0.2	721934	0.0	1.4
neos10	–	–	2.4	-403	64.5	0.9	-372	67.2	1.6
neos11	–	–	1.7	–	–	1.5	–	–	8.7
neos13	-28.0396395	70.6	4.9	-28.0396395	70.6	4.1	-63.6115258	33.4	32.5
neos21	11	57.1	0.1	–	–	0.8	11	57.1	3.0

continue next page

	Coefficient Diving			Fractional Diving			Vectorlength Diving		
Name	Primal Bound	P Gap	HeurTime	Primal Bound	P Gap	HeurTime	Primal Bound	P Gap	HeurTime
neos22	781155	0.2	1.0	781155	0.2	0.9	782905.625	0.4	0.3
neos632659	-94	0.0	0.0	-74	21.3	0.0	-94	0.0	0.2
nug08	–	–	9.9	–	–	9.1	232	8.4	11.0
nw04	90666	437.7	23.8	17094	1.4	8.7	23418	38.9	1.4
p0201	8405	10.4	0.0	–	–	0.0	7905	3.8	0.6
p0282	259890	0.6	0.0	260131	0.7	0.0	258945	0.2	0.1
p0548	–	–	0.0	8701	0.1	0.0	–	–	0.0
p2756	–	–	0.1	3135	0.4	0.5	3136	0.4	0.0
pk1	38	245.5	0.0	38	245.5	0.0	26	136.4	0.3
pp08a	7737.14286	5.3	0.0	10620.748	44.5	0.0	7550	2.7	0.8
pp08aCUTS	7550	2.7	0.1	9658.36859	31.4	0.0	7380	0.4	0.9
prod1	–	–	0.0	–	–	0.0	-48	14.3	0.0
qap10	–	–	88.2	–	–	88.2	–	–	136.1
qiu	805.158298	706.0	0.3	1117.04612	940.7	0.1	278.580725	309.7	1.0
qnet1	17104.4815	6.7	0.2	17716.0174	10.5	0.1	31820.57	98.5	0.6
qnet1_o	17412.5947	8.6	0.2	28665.8309	78.8	0.2	37674.1033	135.0	0.7
ran8x32	5838	11.3	0.2	6619.91379	26.2	0.1	5329	1.6	0.6
ran10x26	5076	18.9	0.3	5888.56508	37.9	0.2	4459	4.4	1.5
ran12x21	4012	9.5	0.2	5359.36364	46.3	0.1	3992	9.0	1.1
ran13x13	3640	11.9	0.1	4756	46.2	0.1	3572	9.8	1.0
rentacar	40802384.7	34.4	0.2	40802384.7	34.4	0.2	30356761	0.0	2.5
rgn	82.1999992	0.0	0.0	82.1999991	0.0	0.0	82.1999991	0.0	0.1
rout	–	–	0.1	–	–	0.0	1473.17	36.7	0.9
set1ch	54706	0.3	0.1	57447.6384	5.3	0.1	54628	0.2	0.3
seymour1	414.920399	1.0	2.6	430.456984	4.8	3.5	411.909941	0.3	12.9
stein27	18	0.0	0.0	18	0.0	0.0	22	22.2	0.1
stein45	32	6.7	0.0	32	6.7	0.0	36	20.0	0.1
swath1	–	–	1.5	–	–	0.6	379.071296	0.0	83.8
swath2	–	–	1.6	–	–	0.5	411.91766	6.9	88.8
vpm2	–	–	0.0	–	–	0.0	15	9.1	0.4
a1c1s1	18149.0069	56.5	1.8	–	–	1.7	–	–	15.2
aflow40b	–	–	0.3	–	–	1.9	1318	12.8	5.6
arki001	–	–	0.1	–	–	0.1	–	–	5.3

continue next page

	Coefficient Diving			Fractional Diving			Vectorlength Diving		
Name	Primal Bound	P Gap	HeurTime	Primal Bound	P Gap	HeurTime	Primal Bound	P Gap	HeurTime
atlanta-ip	–	–	88.5	–	–	43.0	–	–	41.3
binkar10_1	6981.14002	3.5	0.1	6889.07003	2.2	0.1	6835.61003	1.4	0.1
dano3mip	785.714286	12.6	30.2	–	–	40.3	743.464286	6.5	431.2
danoint	–	–	0.2	66.5	1.3	0.3	–	–	2.4
ds	1102.53266	289.0	82.2	725.010161	155.8	25.4	1437.2175	407.1	181.9
glass4	–	–	0.4	–	–	0.3	3.10002857e+09	158.3	0.8
liu	–	–	2.7	–	–	2.5	3132	167.2	7.6
markshare1	419	Large	0.0	243	Large	0.0	136	Large	0.0
markshare2	200	Large	0.0	496	Large	0.0	212	Large	0.0
mkc	–	–	2.2	-500.912	11.2	2.1	-533.432	5.4	2.6
mkc1	–	–	2.1	-597.997	1.5	1.9	-596.519	1.8	1.3
momentum1	–	–	21.3	–	–	9.0	–	–	97.6
momentum2	–	–	17.3	–	–	3.9	–	–	69.3
msc98-ip	–	–	44.2	–	–	16.0	–	–	38.4
neos616206	–	–	0.2	–	–	0.3	–	–	3.8
net12	–	–	10.7	–	–	10.4	–	–	34.9
noswot	-41	0.0	0.0	-40	2.4	0.0	-36	12.2	0.4
nsrand-ipx	64000	25.0	0.5	–	–	2.5	60480	18.1	16.2
opt1217	–	–	0.0	–	–	0.1	-16	0.0	0.1
protfold	–	–	15.3	–	–	2.0	–	–	13.9
rd-rplusc-21	–	–	5.1	–	–	4.3	–	–	0.0
roll3000	–	–	0.7	–	–	0.9	14945	15.6	6.3
seymour	442	4.5	7.5	465	9.9	4.2	431	1.9	27.3
sp97ar	750544279	12.9	3.3	–	–	14.4	723182495	8.8	38.2
swath	–	–	2.4	–	–	2.0	–	–	9.2
swath3	–	–	1.6	–	–	0.4	460.523571	15.8	64.5
t1717	–	–	29.4	–	–	28.6	–	–	81.6
timtab1	–	–	0.0	–	–	0.1	–	–	0.6
timtab2	–	–	0.0	–	–	0.0	–	–	1.1
tr12-30	132548	1.5	0.7	132057	1.1	0.6	159967.543	22.5	3.8
Total (129)			834.6			700.2			2370.7
Geom. Mean			2.3			2.1			3.9

Table B.1. Comparison of different diving heuristics applied to the optimum of the LP-relaxation

Name	$\varphi = 0.0$		$\varphi = 0.8$		$\varphi = 0.9$		$\varphi = 0.95$		$\varphi = 0.99$		$\varphi = 0.999$	
	γ_P	Time	γ_P	Time	γ_P	Time	γ_P	Time	γ_P	Time	γ_P	Time
10teams	–	18	–	22	–	21	**5.8**	12	–	28	–	18
a1c1s1	108.0	7	76.0	3	78.6	5	77.3	3	**59.1**	3	–	3
aflow30a	185.9	0	219.3	0	306.7	0	256.0	0	52.2	0	**42.6**	0
aflow40b	327.7	1	106.7	2	246.9	1	230.5	2	47.4	2	**42.4**	4
air04	5.8	39	5.1	174	7.2	169	0.5	23	**0.1**	29	0.1	297
air05	19.5	7	2.4	4	2.1	2	8.4	37	1.5	1	**0.6**	6
arki001	–	16	–	15	–	14	–	13	–	21	–	13
atlanta-ip	–	75	–	104	–	112	–	119	–	273	–	608
cap6000	11.7	0	12.6	1	**0.4**	1	43.4	1	57.5	5	–	7
dano3mip	43.3	43	15.3	328	8.6	510	**7.9**	943	–	>3600	–	264
danoint	–	1	31.0	2	31.0	3	21.8	4	**15.0**	25	–	35
disctom	**0.0**	8	**0.0**	8	**0.0**	8	**0.0**	8	**0.0**	9	**0.0**	9
ds	–	3281	–	>3600	–	>3600	–	>3600	–	>3600	–	>3600
fast0507	5.7	23	1.7	31	1.7	41	1.1	28	1.1	43	**0.6**	252
fiber	2069.1	0	**274.4**	0	**274.4**	0	2476.9	0	2171.2	0	–	1
fixnet6	796.3	0	410.4	0	**89.8**	0	635.0	0	953.9	0	–	1
gesa2-o	135.6	1	89.0	0	40.4	0	104.1	0	**9.1**	1	12.2	1
gesa2	58.8	0	9.5	0	9.1	0	17.9	0	**0.1**	0	17.4	0
glass4	**280.2**	0	641.7	1	–	0	–	1	–	0	–	1
harp2	21.3	1	23.7	0	22.5	1	**21.0**	1	–	2	–	1
liu	450.3	0	616.6	2	616.6	4	**251.7**	2	–	9	–	9
manna81	1.9	2	**1.7**	2	1.7	2	1.9	2	1.9	4	–	19
markshare1	Large	0	Large	0	Large	0	**Large**	0	Large	0	–	0
markshare2	**Large**	0	Large	0	Large	0	Large	0	Large	1	–	0
mas74	**44.6**	0	84.8	0	91.4	0	74.4	1	–	0	–	1
mas76	9.4	0	11.6	0	24.8	1	**9.1**	0	–	0	–	0
misc07	66.2	1	53.7	1	39.5	0	**19.4**	0	39.1	0	–	0
mkc	75.3	1	53.5	0	53.5	0	53.5	0	**43.1**	3	–	6
mod011	100.0	0	16.8	0	16.8	1	18.0	1	**8.6**	4	–	2
modglob	69.5	0	4.2	0	2.8	0	3.2	0	**2.1**	0	–	1
momentum1	–	161	–	76	–	106	–	121	–	102	–	83
momentum2	–	178	–	165	–	206	–	148	–	212	–	174
momentum3	–	994	48.0	1440	**40.1**	1820	–	1841	–	3403	–	>3600
msc98-ip	32.9	53	32.9	57	32.9	65	29.3	55	**28.9**	72	–	190
mzzv11	–	153	–	173	–	164	11.9	65	**8.3**	84	–	146
mzzv42z	61.8	31	34.2	30	31.3	34	**19.7**	33	–	126	–	88
net12	**57.5**	12	**57.5**	18	**57.5**	22	**57.5**	22	**57.5**	27	–	46
noswot	68.3	0	31.7	0	**14.6**	0	–	0	34.1	0	–	1
nsrand-ipx	1023.1	1	78.4	1	107.5	1	**71.2**	1	420.0	2	144.4	7
nw04	13.4	4	**4.4**	5	13.3	9	**4.4**	8	17.5	41	–	77
opt1217	**0.0**	0	**0.0**	0	**0.0**	0	**0.0**	0	**0.0**	0	–	1
p2756	–	5	–	3	–	4	–	7	–	8	–	4
pk1	418.2	0	445.5	0	890.9	0	945.5	1	**336.4**	0	–	0
pp08a	83.0	0	**31.0**	1	41.1	0	96.1	0	66.7	0	–	0
pp08aCUTS	79.9	0	23.1	0	**13.7**	0	28.6	0	61.8	0	–	0
protfold	70.0	183	50.0	167	–	683	–	366	**30.0**	307	–	273
qiu	1146.5	1	950.3	1	224.3	0	82.5	0	**72.6**	1	–	2
rd-rplusc-21	–	151	–	175	–	182	–	93	–	174	–	152
roll3000	**45.8**	1	–	7	64.5	7	84.3	7	–	11	–	18
rout	53.8	1	**29.2**	1	52.3	0	52.5	0	52.8	0	–	2
set1ch	**53.4**	0	64.3	0	58.4	0	66.7	0	87.7	1	–	0
seymour	6.4	2	3.8	3	3.3	4	**1.4**	3	1.9	4	1.7	26
sp97ar	973.0	4	54.3	3	36.7	4	**25.1**	5	461.7	10	364.6	56
stp3d	–	>3600	–	>3600	–	>3600	–	>3600	–	>3600	–	>3600
swath	158.4	4	140.6	17	**13.0**	10	44.7	18	–	42	–	25
t1717	61.0	417	75.5	286	56.6	220	29.3	153	**7.1**	562	–	1317
timtab1	93.1	0	41.1	1	87.4	0	**33.6**	1	56.8	1	–	1
timtab2	–	3	–	3	63.7	3	**62.2**	1	–	7	–	2
tr12-30	109.5	0	53.9	0	**25.4**	0	48.2	0	43.4	1	–	0
vpm2	129.1	0	20.0	0	16.4	0	**12.7**	0	123.6	0	–	0
bell3a	Large	0	8832.0	0	5346.7	0	5436.1	0	8706.8	0	**1.0**	0

continue next page

Name	$\varphi = 0.0$		$\varphi = 0.8$		$\varphi = 0.9$		$\varphi = 0.95$		$\varphi = 0.99$		$\varphi = 0.999$	
	γ_P	Time	γ_P	Time	γ_P	Time	γ_P	Time	γ_P	Time	γ_P	Time
bell5	638.8	0	364.6	0	478.1	1	**322.8**	0	393.4	0	–	1
gesa3	28.3	0	23.0	0	65.2	0	**0.1**	1	0.1	0	1.3	0
gesa3_o	186.9	1	124.8	0	179.8	0	1.0	0	**0.2**	0	2.5	0
l152lav	4.5	0	10.7	1	3.5	1	1.2	0	**1.0**	0	–	2
stein45	13.3	0	13.3	0	36.7	0	20.0	0	6.7	1	**0.0**	1
ran8x32	101.2	0	15.5	0	**12.3**	0	12.5	0	46.9	0	–	0
ran10x26	83.1	0	**12.4**	0	12.7	0	13.0	0	161.6	0	–	1
ran12x21	269.1	0	**14.1**	0	18.4	0	67.9	0	85.5	0	–	0
ran13x13	136.7	0	25.8	0	22.0	0	**10.6**	0	130.8	0	–	0
binkar10_1	Large	0	**5.4**	0	**5.4**	0	**5.4**	0	**5.4**	0	13.0	1
eilD76	–	10	47.4	8	–	11	**45.4**	7	–	11	–	8
irp	2.9	2	17.7	3	3.1	2	4.6	4	0.4	9	**0.1**	3
mas284	14.2	0	**4.7**	0	5.8	0	10.3	0	–	0	–	0
prod1	44.6	0	35.7	0	**28.6**	1	30.4	0	39.3	0	–	0
bc1	495.6	2	63.3	2	93.1	2	**3.3**	3	212.0	5	91.8	32
bienst1	57.4	1	36.4	0	**16.9**	0	40.1	1	–	0	–	0
bienst2	99.3	0	48.4	0	40.1	1	**21.8**	0	–	1	–	0
dano3_3	6.4	38	**0.0**	53	**0.0**	62	**0.0**	104	**0.0**	337	–	205
dano3_4	13.1	39	**0.0**	79	**0.0**	95	**0.0**	157	0.0	395	–	197
dano3_5	17.5	39	0.2	117	**0.0**	154	**0.0**	275	0.1	752	–	193
mkc1	13.9	0	7.0	0	5.6	1	7.2	1	7.2	0	**1.4**	5
neos1	473.7	1	342.1	1	**268.4**	0	384.2	0	352.6	1	–	1
neos2	–	5	–	5	–	6	–	6	–	8	–	5
neos3	–	6	–	7	–	7	–	7	–	9	–	6
neos4	–	4	–	5	–	5	–	4	–	3	–	3
neos5	–	3	–	5	–	4	–	5	–	3	–	3
neos6	90.4	13	42.2	11	62.7	25	**13.3**	41	22.9	171	–	79
neos7	587.3	0	386.3	1	**10.3**	1	15.5	1	18.3	1	–	3
nug08	9.3	3	**0.0**	3	**0.0**	3	**0.0**	3	5.6	4	2.8	19
qap10	22.9	53	12.4	60	**0.0**	59	20.6	107	**0.0**	211	13.5	1385
seymour1	5.8	3	1.3	4	1.7	4	2.0	5	**0.3**	15	0.8	102
swath1	127.6	0	38.9	2	36.0	2	**30.4**	2	44.6	2	–	6
swath2	249.2	1	**17.0**	2	73.1	2	51.4	2	19.3	2	–	6
acc-0	**0.0**	0	**0.0**	0	**0.0**	0	**0.0**	0	**0.0**	1	**0.0**	0
acc-1	**0.0**	2	**0.0**	2	**0.0**	2	**0.0**	2	**0.0**	1	**0.0**	2
acc-2	**0.0**	5	**0.0**	4	**0.0**	4	**0.0**	4	**0.0**	4	**0.0**	4
acc-3	–	285	–	242	–	284	–	226	–	389	–	493
acc-4	–	254	–	293	–	291	–	293	–	479	–	613
acc-5	–	182	–	175	–	202	–	197	–	320	–	352
acc-6	–	198	–	216	**0.0**	96	–	211	–	346	–	392
ic97_potential	–	2	–	2	–	4	–	2	–	3	–	3
ic97_tension	–	1	**13.0**	1	–	2	–	1	–	2	–	1
icir97_tension	–	10	–	9	–	9	–	10	**3.4**	8	–	24
icir97_potential	–	6	–	6	–	7	–	6	–	8	–	11
nh97_potential	–	4	–	5	–	4	–	4	–	4	–	5
nh97_tension	–	7	–	7	–	8	–	6	–	7	–	10
B10-011000	544.6	3	544.6	2	509.8	1	**485.5**	2	524.5	3	–	5
B10-011001	423.4	3	**405.9**	2	424.6	2	467.5	2	470.8	2	–	4
B11-010000	560.8	9	559.1	9	**525.3**	8	560.0	7	532.3	10	–	20
B11-110001	**271.8**	12	278.9	16	297.4	28	274.7	18	316.4	22	–	38
B12-111111	–	65	–	49	–	62	–	78	–	85	–	66
C10-001000	1821.4	1	1829.8	1	1777.5	1	1645.8	2	**1302.7**	1	–	3
C10-100001	1096.1	3	994.5	5	663.1	5	**597.1**	4	–	7	–	8
C11-010100	450.4	4	**437.8**	5	450.4	6	450.4	5	450.4	12	–	18
C11-011100	539.6	3	549.1	2	547.1	3	**520.2**	3	535.2	4	–	6
C12-100000	1401.7	19	1387.6	21	1316.8	32	1337.6	33	**293.3**	23	–	50
C12-111100	215.0	14	202.8	6	209.1	2	189.3	20	**182.7**	12	–	10
neos10	101.3	514	**64.8**	11	100.2	9	101.6	336	100.7	354	–	153
neos16	–	1	–	1	–	2	–	2	–	2	–	2
neos20	–	3	–	3	–	3	–	2	–	2	–	2
G. Mean (75|121)	106.7	4.7	47.0	5.1	43.1	5.3	37.8	5.6	39.8	7.4	–	9.1

Table B.2. Comparison of different feasibility pump versions

Name	Primal Bound	P Gap	FixInt	FixAll	Nodes RENS	PreTime	HeurTime
10teams	–	–	91.312	91.312	3	5.7	9.5
30:70:4_5:0_5:100	25	177.8	82.086	82.086	506825	239.7	3361.7
30:70:4_5:0_95:98	155	1191.7	80.871	80.871	544291	189.2	3412.3
30:70:4_5:0_95:100	9	200.0	84.456	84.456	606852	227.3	3376.9
acc-0	–	–	79.383	79.383	1	10.7	2.2
acc-1	–	–	74.074	74.074	1	22.6	1.7
acc-2	–	–	68.889	68.889	1917	24.0	17.5
acc-3	–	–	59.427	59.427	2845	40.2	32.3
acc-4	–	–	65.350	65.350	197	39.9	9.4
acc-5	–	–	64.700	64.700	5	22.1	5.4
acc-6	–	–	59.921	59.921	23	18.8	6.5
aflow30a	1158	0.0	81.948	80.760	13639	4.0	12.7
air03	394874	16.1	99.662	99.662	15	35.7	0.3
air04	–	–	96.866	96.866	3	72.5	6.9
air05	–	–	96.323	96.323	5	48.8	4.9
bc1	3.4198842	2.4	95.635	48.104	125	21.9	2.9
bell3a	878430.316	0.0	98.214	57.273	2	0.0	0.1
bell5	–	–	76.923	43.617	0	0.0	0.0
bienst1	46.75	0.0	0.000	0.000	97684	1.1	258.1
bienst2	54.6	0.0	0.000	0.000	971372	2.5	3598.2
blend2	8.103921	6.6	96.761	96.552	2	0.1	0.0
cap6000	-2443599	0.3	99.966	99.966	1	1.3	0.1
dano3_3	576.344633	0.0	85.507	9.947	32	44.4	72.4
dano3_4	576.435225	0.0	80.435	12.651	88	57.6	153.3
dano3_5	576.924916	0.0	78.261	15.361	1115	63.3	561.3
disctom	–	–	99.259	99.259	0	47.2	2.0
dcmulti	188182	0.0	35.135	22.303	309	1.3	0.7
dsbmip	-305.198175	0.0	91.429	19.314	1	0.4	0.1
egout	572.23465	0.7	92.857	94.231	1	0.0	0.0
eilD76	979.945566	10.7	92.046	92.046	255	22.8	1.0
enigma	–	–	97.000	97.000	0	0.0	0.0
fast0507	177	1.7	99.562	99.562	4821	106.7	118.3
fiber	419895.5	3.4	96.777	96.777	107	0.4	0.1
fixnet6	3986	0.1	92.593	85.861	153	0.7	0.1
flugpl	–	–	22.222	13.333	0	0.0	0.0
gesa2-o	–	–	89.722	80.882	1395	4.5	1.0
gesa2	25782398.1	0.0	84.804	73.039	494	2.7	0.5
gesa3	27991430	0.0	89.062	76.773	71	2.6	0.1
gesa3_o	–	–	90.586	83.245	28	3.8	0.2
gt2	–	–	97.688	97.688	0	0.0	0.0
irp	12409.4125	2.1	99.850	99.850	1	10.1	0.4
khb05250	106940226	0.0	83.333	40.801	5	0.4	0.0
l152lav	–	–	97.788	97.788	3	1.8	1.2
lseu	1148	2.5	78.409	78.409	23	0.0	0.0
mas74	14372.8713	21.8	91.946	91.333	89	0.0	0.0
mas76	40560.0518	1.4	92.617	92.000	19	0.0	0.0
mas284	93597.2337	2.4	86.577	86.000	14244	0.1	2.9
misc03	–	–	86.957	86.957	1	0.3	0.0
misc06	12852.2468	0.0	94.643	41.649	19	0.2	0.1
misc07	–	–	98.276	98.276	0	0.2	0.0
mitre	115170	0.0	99.972	99.972	1	17.0	0.3
mod008	308	0.3	94.984	94.984	13	0.1	0.0
mod011	-54205576.1	0.6	57.292	21.969	2311	21.5	57.1
modglob	20930259.7	0.9	51.020	41.085	73591	0.3	39.5
mzzv11	–	–	88.569	88.060	0	341.9	0.3
mzzv42z	–	–	93.139	92.762	0	304.2	0.3
neos1	20	5.3	98.495	98.495	1	4.5	0.0
neos2	–	–	96.146	94.269	3	18.0	0.3
neos3	–	–	95.938	93.842	1	28.6	0.1
neos5	-4.86034408e+10	0.0	96.510	89.345	1	326.0	0.7
neos6	–	–	98.669	96.356	1549	19.2	3.4
neos7	721934	0.0	97.266	81.794	13	5.1	0.8
neos10	-372	67.2	99.748	99.748	1	312.9	0.1
neos11	–	–	71.429	61.770	325	8.9	3.5
neos13	-63.1134148	33.9	78.788	78.270	109547	139.8	3460.5

continue next page

Name	Primal Bound	P Gap	FixInt	FixAll	Nodes RENS	PreTime	HeurTime
neos21	7	0.0	74.291	74.291	5098	2.3	10.2
neos22	779715	0.0	85.683	31.192	10558	4.9	31.4
neos632659	-94	0.0	57.000	53.333	3569	0.2	0.8
noswot	–	–	90.526	89.167	39	0.0	0.2
nug08	–	–	61.458	61.458	5	75.6	8.2
nw04	22494	33.4	99.970	99.970	9	74.3	2.1
p0201	7905	3.8	64.103	64.103	33	0.2	0.5
p0282	258945	0.2	77.228	77.228	90	0.2	0.1
p0548	8763	0.8	99.127	99.127	1	0.3	0.0
p2756	3359	7.5	99.658	99.658	1	1.2	0.1
pk1	26	136.4	70.909	45.349	1140	0.0	0.2
pp08a	7480	1.8	59.375	33.750	2896	0.4	2.2
pp08aCUTS	7350	0.0	50.000	29.583	1388	0.5	1.3
prod1	–	–	73.826	73.092	49	0.3	0.5
qap10	–	–	69.494	69.494	3	262.3	55.3
qiu	-128.466917	3.3	25.000	25.000	806811	2.7	3599.2
qnet1	21159.9639	32.0	96.965	96.965	1	1.4	0.0
qnet1_o	19351.9	20.7	95.625	95.625	7	1.4	0.0
ran8x32	5329	1.6	81.250	82.422	1403	0.3	0.5
ran10x26	4347	1.8	74.615	75.769	292670	1.0	129.4
ran12x21	3754	2.5	73.809	74.603	162252	1.0	68.2
ran13x13	3364	3.4	70.414	71.302	335888	0.7	115.7
rentacar	30356761	0.0	75.000	7.756	11	1.8	2.2
rgn	82.1999991	0.0	78.000	44.571	363	0.0	0.1
rout	–	–	86.349	86.667	65	0.4	0.7
set1ch	54628	0.2	96.170	91.441	23	1.2	0.0
seymour1	410.963143	0.0	78.495	23.267	251965	14.9	3585.5
stein27	18	0.0	11.111	11.111	2302	0.0	1.9
stein45	30	0.0	17.778	17.778	12219	0.2	9.2
swath1	–	–	99.426	47.263	21	41.9	78.3
swath2	–	–	99.038	43.275	43	45.7	84.2
vpm2	13.75	0.0	56.627	50.276	6359	0.4	2.1
a1c1s1	–	–	19.271	8.188	207911	79.8	3521.4
aflow40b	1168	0.0	92.815	92.302	200655	32.8	197.1
arki001	–	–	86.542	67.188	89	2.9	1.8
atlanta-ip	–	–	91.201	88.210	0	265.1	0.4
binkar10_1	6746.76002	0.1	48.235	47.230	726439	0.9	833.6
dano3mip	743.133333	6.5	73.913	70.410	15191	168.8	3431.8
danoint	65.6666667	0.0	7.143	0.768	53320	3.5	175.2
ds	1045.71	268.9	99.211	99.211	224694	318.6	3282.5
glass4	2.90001895e+09	141.7	86.242	81.073	11939411	0.1	3021.3
liu	3132	167.2	53.634	50.520	1561530	10.3	3594.6
markshare1	136	Large	82.000	82.000	17	0.0	0.0
markshare2	212	Large	85.000	85.000	50	0.0	0.0
mkc	-541.112	4.0	97.133	97.114	72971	12.4	19.8
mkc1	-596.519	1.8	98.635	97.234	121	6.2	0.1
momentum1	–	–	81.203	72.316	323807	127.6	3472.9
momentum2	–	–	85.118	78.420	426888	128.7	3474.1
msc98-ip	–	–	93.313	91.532	0	373.9	0.3
neos616206	–	–	55.454	55.454	207953	2.0	119.2
net12	–	–	83.738	75.772	0	189.9	0.3
nsrand-ipx	57120	11.6	98.561	98.546	5304594	16.4	1381.6
opt1217	-16	0.0	97.230	97.101	19	0.2	0.0
protfold	–	–	74.060	74.060	757	24.6	4.2
rd-rplusc-21	–	–	78.934	64.330	17281	1073.0	30.8
roll3000	13974	8.1	85.501	72.077	11535	5.5	11.9
seymour	430	1.7	62.151	62.151	310809	30.6	3570.1
sp97ar	691318495	4.0	98.766	98.766	2395490	58.9	3555.1
swath	–	–	99.984	98.718	1	31.7	0.2
swath3	–	–	98.864	49.399	55	75.48	56.87
t1717	–	–	99.194	99.194	179	609.3	17.1
timtab1	902037	17.9	26.712	20.398	1621418	1.4	690.2
timtab2	–	–	13.386	9.971	16035	5.2	25.6
tr12-30	131616	0.8	65.625	45.134	2673240	23.6	3590.9

Table B.3. RENS applied to the optimum of the LP-relaxation

	Rens deactivated		Rens only at root node		Rens every 10th depth	
Name	Nodes	Time	Nodes	Time	Nodes	Time
10teams	324	30.8	324	31.0	324	31.2
30:70:4_5:0_5:100	167	349.5	167	364.7	167	366.1
30:70:4_5:0_95:98	125	298.9	125	312.1	125	313.8
30:70:4_5:0_95:100	109	315.4	109	323.5	109	324.7
acc-0	1	14.8	1	14.7	1	15.0
acc-1	1	29.3	1	29.3	1	29.5
acc-2	79	91.3	79	99.8	79	99.1
acc-3	75	169.5	75	186.1	75	186.6
acc-4	236	401.7	236	423.0	236	417.4
acc-5	5011	1449.4	5011	1434.5	5011	1435.1
acc-6	18	75.1	18	80.7	18	81.0
aflow30a	8309	47.8	6026	35.3	6026	35.3
air03	2	25.5	2	25.8	2	25.8
air04	153	157.8	153	158.0	153	159.0
air05	239	90.7	239	96.9	239	98.3
bc1	19008	843.1	16043	735.1	16043	741.2
bell3a	48995	44.4	48995	45.2	48995	45.1
bell5	1170	1.2	1170	1.2	1170	1.2
bienst1	9574	48.2	9574	48.0	10358	57.8
bienst2	101372	605.2	101372	598.5	91427	561.8
blend2	5761	9.8	879	3.4	879	3.4
cap6000	2937	37.5	2937	37.5	2937	37.8
dano3_3	19	191.1	19	190.7	19	191.2
dano3_4	41	247.9	41	248.2	41	248.8
dano3_5	186	513.7	186	511.8	186	513.7
disctom	1	66.7	1	66.5	1	66.7
dcmulti	168	4.8	168	4.8	168	4.9
dsbmip	1	0.8	1	0.8	1	0.8
egout	2	0.0	2	0.0	2	0.0
eilD76	4636	103.7	1359	107.7	1359	107.8
enigma	4455	1.5	4455	1.5	4455	1.6
fast0507	1488	2424.6	1488	2432.9	1488	2430.7
fiber	209	3.8	137	2.4	137	2.4
fixnet6	68	1.6	17	1.5	17	1.5
flugpl	474	0.3	474	0.4	474	0.4
gesa2-o	1604	12.0	1604	12.5	1604	12.6
gesa2	444	6.3	93	4.5	93	4.5
gesa3	928	10.6	28	3.7	28	3.8
gesa3_o	725	10.8	725	10.7	725	10.9
gt2	137	0.1	137	0.2	137	0.1
irp	544	94.7	312	59.1	312	59.2
khb05250	12	0.5	10	0.5	10	0.5
l152lav	63	5.6	63	7.1	63	7.1
lseu	415	0.4	302	0.8	302	0.8
mas74	4856815	1534.6	4856815	1527.1	4856815	1567.1
mas76	366438	178.1	347300	103.7	347300	105.1
mas284	17489	31.6	17379	31.6	17379	32.3
misc03	52	1.4	52	2.0	52	1.9
misc06	24	0.5	24	0.6	24	0.6
misc07	34963	47.0	34963	47.9	34963	47.0
mitre	27	86.7	27	86.8	27	86.8
mod008	212	0.8	212	0.9	212	0.8
mod011	2449	171.5	2449	172.6	2449	180.7
modglob	3125	5.5	3815	7.0	3815	7.0
mzzv11	2541	1138.7	2541	1159.6	2541	1143.0
mzzv42z	1452	677.3	1452	685.5	1452	677.8

continue next page

	RENS deactivated		RENS only at root node		RENS every 10th depth	
Name	Nodes	Time	Nodes	Time	Nodes	Time
neos1	1	6.1	1	6.0	1	6.0
neos2	51967	203.5	51967	203.4	51967	207.8
neos3	508140	2463.8	508140	2392.5	508140	2399.5
neos5	2	138.7	2	141.2	2	139.6
neos6	3738	396.3	3738	398.4	3738	399.8
neos7	55463	599.1	51407	557.0	51407	562.1
neos10	7	183.5	7	183.9	7	183.9
neos11	6796	788.0	6796	783.3	6796	799.9
neos13	11242	768.4	11242	798.4	11242	822.6
neos21	2223	41.3	2223	41.9	2223	42.3
neos22	35891	394.8	35891	419.3	35891	397.6
neos632659	45635	29.8	31971	20.8	31971	21.1
nug08	3	73.3	3	78.8	3	79.5
nw04	3	65.8	3	67.4	3	66.8
p0201	262	1.9	262	2.4	262	2.4
p0282	76	0.9	35	0.5	35	0.5
p0548	46	0.7	46	0.8	46	0.8
p2756	194	12.8	194	12.7	194	12.8
pk1	267839	116.2	293834	131.7	293834	142.3
pp08a	1817	4.1	1817	4.1	1874	3.8
pp08aCUTS	2408	6.3	2514	7.9	2514	8.0
prod1	59023	47.9	59023	48.9	59023	53.9
qap10	5	404.3	5	448.4	5	448.2
qiu	10584	184.7	10584	184.9	12232	216.3
qnet1	130	6.3	132	6.6	132	6.5
qnet1_o	275	6.6	134	6.1	134	6.0
ran8x32	15057	30.0	15693	28.6	15693	28.8
ran10x26	34306	75.0	29868	67.1	29868	67.3
ran12x21	122274	222.6	136566	242.1	136566	240.3
ran13x13	66728	83.2	67410	82.6	67410	81.5
rentacar	4	4.8	4	4.8	4	4.8
rgn	2341	1.0	2341	1.1	2341	1.1
rout	32398	68.1	32398	68.9	32398	69.1
set1ch	59	2.0	54	2.0	54	1.9
seymour1	4564	979.4	4564	988.1	4564	986.0
stein27	4173	4.0	4173	3.9	4173	3.7
stein45	53354	56.4	53354	55.7	54951	59.0
swath1	2435	87.5	2435	102.9	2435	172.9
swath2	12543	167.9	12543	181.8	12543	195.3
vpm2	10029	8.8	9890	9.6	9890	9.6
Total (96)	6929908	21447.7	6915451	21364.1	6909592	21537.8
Geom. Mean	704	36.7	614	35.9	615	36.3

Table B.4. Integration of RENS into SCIP, easy instances

	Rens deactivated		Rens only at root node		Rens every 10th depth	
Name	Primal Bound	Nodes	Primal Bound	Nodes	Primal Bound	Nodes
a1c1s1	12657.4229	24653	12657.4229	24391	12657.4229	20031
aflow40b	1232	264660	1235	229040	1250	217989
arki001	7584054.38	592707	7584054.38	591564	7584054.38	590693
atlanta-ip	1e+20	436	1e+20	436	1e+20	436
binkar10_1	6746.76002	433757	6746.76002	433757	6746.76002	433757
dano3mip	727.0625	1063	729.608696	877	729.608696	804
danoint	66.375	219986	66.375	220165	66.375	217115
ds	360.715161	617	360.715161	615	360.715161	615
glass4	1.7000151e+09	3726852	1.77663851e+09	3689102	1.70001355e+09	3771237
liu	2948	720469	2490	656324	2510	627469
markshare1	6	26340583	6	26340591	6	26340591
markshare2	15	18686795	15	18686795	15	18686795
mkc	-552.744	284256	-556.212	319354	-549.692	348444
mkc1	-606.717	610712	-606.767	614881	-606.767	610207
momentum1	160490.615	2205	160490.615	2205	160490.615	2197
momentum2	1e+20	1949	1e+20	1910	1e+20	1917
msc98-ip	1e+20	220	1e+20	220	1e+20	220
neos616206	937.6	815310	937.6	814177	937.6	802986
net12	296	1364	296	1360	296	1350
noswot	-41	9731424	-41	9760690	-41	9551961
nsrand-ipx	55680	144107	54880	153612	55520	119355
opt1217	-16.000007	2425474	-16.000007	2469635	-16.000007	2462413
protfold	1e+20	871	1e+20	864	1e+20	853
rd-rplusc-21	1e+20	37134	1e+20	36465	1e+20	36822
roll3000	12960	173560	12960	172091	12960	172575
seymour	426	5761	425	5550	426	5328
sp97ar	690159742	12132	674161010	11411	674161010	11515
swath	499.694345	189986	499.694345	189986	499.694345	189986
swath3	397.761344	227579	397.761344	227579	397.761344	227579
t1717	192060	447	192060	443	192060	443
timtab1	915760.996	3148630	915760.996	3110297	915760.996	3107288
timtab2	1661007	1858791	1661007	1855999	1661007	1857879
tr12-30	130723	387478	130701	405614	130701	405614

Table B.5. Integration of Rens into SCIP, hard instances

	Simple Rounding			Rounding			Shifting			RENS			Best Known Rounding	
Name	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap
10teams	–	–	0.0	–	–	0.0	–	–	0.0	–	–	11.0	–	–
30:70:4_5:0_5:100	–	–	0.0	2300	Large	0.3	2300	Large	0.3	214	2277.8	21.2	25	177.8
30:70:4_5:0_95:98	–	–	0.0	2574	Large	0.5	2574	Large	0.4	155	1191.7	22.1	155	1191.7
30:70:4_5:0_95:100	–	–	0.0	2048	Large	0.2	2048	Large	0.2	70	2233.3	20.5	9	200.0
acc-0	–	–	0.0	–	–	0.0	–	–	0.0	–	–	2.3	–	–
acc-1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	1.8	–	–
acc-2	–	–	0.0	–	–	0.0	–	–	0.0	–	–	10.5	–	–
acc-3	–	–	0.0	–	–	0.0	–	–	0.0	–	–	13.6	–	–
acc-4	–	–	0.0	–	–	0.0	–	–	0.0	–	–	9.7	–	–
acc-5	–	–	0.0	–	–	0.0	–	–	0.0	–	–	5.4	–	–
acc-6	–	–	0.0	–	–	0.0	–	–	0.0	–	–	6.7	–	–
aflow30a	–	–	0.0	–	–	0.0	–	–	0.0	1267	9.4	0.9	1158	0.0
air03	–	–	0.0	1184204	248.1	0.0	1184204	248.1	0.0	394874	16.1	0.3	394874	16.1
air04	–	–	0.0	–	–	0.1	–	–	0.2	–	–	7.0	–	–
air05	–	–	0.0	–	–	0.1	–	–	0.2	–	–	5.0	–	–
bc1	–	–	0.0	–	–	0.0	–	–	0.0	3.4198842	2.4	2.8	3.4198842	2.4
bell3a	880414.281	0.2	0.0	880414.281	0.2	0.0	880414.281	0.2	0.0	878430.316	0.0	0.1	878430.316	0.0
bell5	–	–	0.0	–	–	0.0	9787786.46	9.2	0.0	–	–	0.0	–	–
bienst1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	46.75	0.0
bienst2	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	54.6	0.0
blend2	–	–	0.0	–	–	0.0	–	–	0.0	8.103921	6.6	0.0	8.103921	6.6
cap6000	-2441736	0.4	0.0	-2441736	0.4	0.0	-2441736	0.4	0.0	-2443599	0.3	0.1	-2443599	0.3
dano3_3	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.3	576.344633	0.0
dano3_4	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.4	576.435225	0.0
dano3_5	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.4	576.924916	0.0
disctom	–	–	0.0	–	–	0.1	–	–	0.2	–	–	2.0	–	–
dcmulti	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	188182	0.0
dsbmip	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	-305.198175	0.0
egout	579.718806	2.0	0.0	579.718806	2.0	0.0	579.718806	2.0	0.0	572.23465	0.7	0.0	572.23465	0.7
eilD76	3630.85679	310.1	0.0	3630.85679	310.1	0.0	3630.85679	310.1	0.0	979.945566	10.7	1.0	979.945566	10.7
enigma	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	–	–
fast0507	508	192.0	0.0	508	192.0	0.1	508	192.0	0.1	179	2.9	20.6	177	1.7
fiber	–	–	0.0	–	–	0.0	1304481.19	221.4	0.0	419895.5	3.4	0.1	419895.5	3.4

continue next page

	Simple Rounding			Rounding			Shifting			Rens			Best Known Rounding	
Name	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap
fixnet6	8462.72852	112.5	0.0	8462.72852	112.5	0.0	8462.72852	112.5	0.0	3986	0.1	0.2	3986	0.1
flugpl	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	–	–
gesa2-o	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.7	–	–
gesa2	–	–	0.0	–	–	0.0	27379074.5	6.2	0.0	25782398.1	0.0	0.5	25782398.1	0.0
gesa3	–	–	0.0	–	–	0.0	39136522.8	39.8	0.0	27991430	0.0	0.1	27991430	0.0
gesa3_o	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.2	–	–
gt2	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	–	–
irp	18142.1226	49.2	0.0	18142.1226	49.2	0.0	18142.1226	49.2	0.1	12409.4125	2.1	0.6	12409.4125	2.1
khb05250	111896869	4.6	0.0	111896869	4.6	0.0	111896869	4.6	0.0	106940226	0.0	0.1	106940226	0.0
l152lav	–	–	0.0	–	–	0.0	–	–	0.0	–	–	1.2	–	–
lseu	–	–	0.0	–	–	0.0	3089	175.8	0.0	1148	2.5	0.0	1148	2.5
mas74	–	–	0.0	–	–	0.0	13755.8924	16.6	0.0	14372.8713	21.8	0.0	14372.8713	21.8
mas76	–	–	0.0	–	–	0.0	46793.6933	17.0	0.0	40560.0518	1.4	0.0	40560.0518	1.4
mas284	–	–	0.0	–	–	0.0	103611.404	13.4	0.0	95703.3781	4.7	0.3	93597.2337	2.4
misc03	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	–	–
misc06	12887.8134	0.3	0.0	12887.8134	0.3	0.0	12887.8134	0.3	0.0	12852.2468	0.0	0.1	12852.2468	0.0
misc07	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	–	–
mitre	–	–	0.0	115185	0.0	0.1	115185	0.0	0.2	115170	0.0	0.3	115170	0.0
mod008	1182	285.0	0.0	1182	285.0	0.0	1182	285.0	0.0	308	0.3	0.0	308	0.3
mod011	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.2	-54205576.1	0.6
modglob	21113576.7	1.8	0.0	21113576.7	1.8	0.0	21113576.7	1.8	0.0	20930259.7	0.9	0.6	20930259.7	0.9
mzzv11	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.3	–	–
mzzv42z	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.3	–	–
neos1	–	–	0.0	31	63.2	0.0	31	63.2	0.0	20	5.3	0.1	20	5.3
neos2	–	–	0.0	–	–	0.0	–	–	0.1	–	–	0.4	–	–
neos3	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.1	–	–
neos5	–	–	0.0	–	–	0.1	–	–	0.1	-4.86034408e+10	0.0	0.8	-4.86034408e+10	0.0
neos6	–	–	0.0	–	–	0.0	–	–	0.1	–	–	2.5	–	–
neos7	–	–	0.0	–	–	0.0	–	–	0.0	721934	0.0	0.8	721934	0.0
neos10	–	–	0.0	–	–	0.0	–	–	0.0	-372	67.2	0.1	-372	67.2
neos11	–	–	0.0	–	–	0.0	–	–	0.0	–	–	3.5	–	–
neos13	–	–	0.0	–	–	0.0	–	–	0.5	-62.3278436	34.7	34.4	-63.1134148	33.9
neos21	–	–	0.0	30	328.6	0.0	30	328.6	0.0	11	57.1	1.6	7	0.0

continue next page

	Simple Rounding			Rounding			Shifting			RENS			Best Known Rounding	
Name	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap
neos22	–	–	0.0	–	–	0.0	814466.191	4.5	0.0	782405.625	0.3	3.4	779715	0.0
neos632659	–	–	0.0	–	–	0.0	6	106.4	0.0	-94	0.0	0.2	-94	0.0
noswot	–	–	0.0	–	–	0.0	-35	14.6	0.0	–	–	0.2	–	–
nug08	–	–	0.0	–	–	0.0	–	–	0.0	–	–	8.4	–	–
nw04	–	–	0.0	23418	38.9	0.7	23418	38.9	2.5	22494	33.4	2.8	22494	33.4
p0201	–	–	0.0	–	–	0.0	–	–	0.0	7905	3.8	0.5	7905	3.8
p0282	–	–	0.0	327891	26.9	0.0	327891	26.9	0.0	258945	0.2	0.1	258945	0.2
p0548	–	–	0.0	–	–	0.0	–	–	0.0	8763	0.8	0.0	8763	0.8
p2756	–	–	0.0	–	–	0.0	3293	5.4	0.0	3359	7.5	0.1	3359	7.5
pk1	–	–	0.0	–	–	0.0	102	827.3	0.0	26	136.4	0.2	26	136.4
pp08a	12646.9248	72.1	0.0	12646.9248	72.1	0.0	12646.9248	72.1	0.0	7660	4.2	0.7	7480	1.8
pp08aCUTS	13379.1374	82.0	0.0	13379.1374	82.0	0.0	13379.1374	82.0	0.0	7350	0.0	1.2	7350	0.0
prod1	–	–	0.0	–	–	0.0	-48	14.3	0.0	–	–	0.5	–	–
qap10	–	–	0.0	–	–	0.3	–	–	0.4	–	–	56.6	–	–
qiu	1805.17714	1458.6	0.0	1805.17714	1458.6	0.0	1805.17714	1458.6	0.0	–	–	0.0	-128.466917	3.3
qnet1	–	–	0.0	–	–	0.0	45592.0759	184.4	0.0	21159.9639	32.0	0.0	21159.9639	32.0
qnet1_o	–	–	0.0	46163.758	188.0	0.0	46163.758	188.0	0.0	19351.9	20.7	0.1	19351.9	20.7
ran8x32	9139.68621	74.2	0.0	9139.68621	74.2	0.0	9139.68621	74.2	0.0	5329	1.6	0.3	5329	1.6
ran10x26	10100.4015	136.5	0.0	10100.4015	136.5	0.0	10100.4015	136.5	0.0	4459	4.4	1.2	4347	1.8
ran12x21	9034.10774	146.6	0.0	9034.10774	146.6	0.0	9034.10774	146.6	0.0	3782	3.2	1.1	3754	2.5
ran13x13	8045.50912	147.4	0.0	8045.50912	147.4	0.0	8045.50912	147.4	0.0	3636	11.8	0.6	3364	3.4
rentacar	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.1	30356761	0.0
rgn	–	–	0.0	–	–	0.0	–	–	0.0	82.1999991	0.0	0.1	82.1999991	0.0
rout	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.7	–	–
set1ch	57621.6608	5.7	0.0	57621.6608	5.7	0.0	57621.6608	5.7	0.0	54628	0.2	0.0	54628	0.2
seymour1	449.489972	9.4	0.0	449.489972	9.4	0.0	449.489972	9.4	0.0	–	–	0.1	410.963143	0.0
stein27	27	50.0	0.0	27	50.0	0.0	27	50.0	0.0	–	–	0.0	18	0.0
stein45	45	50.0	0.0	45	50.0	0.0	45	50.0	0.0	–	–	0.0	30	0.0
swath1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	79.4	–	–
swath2	–	–	0.0	–	–	0.0	–	–	0.0	–	–	84.9	–	–
vpm2	–	–	0.0	–	–	0.0	–	–	0.0	13.75	0.0	0.5	13.75	0.0
a1c1s1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	–	–
aflow40b	–	–	0.0	–	–	0.0	–	–	0.0	1318	12.8	1.4	1168	0.0

continue next page

	Simple Rounding			Rounding			Shifting			Rens			Best Known Rounding	
Name	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap
arki001	–	–	0.0	–	–	0.0	–	–	0.0	–	–	1.8	–	–
atlanta-ip	–	–	0.0	–	–	0.0	–	–	0.2	–	–	0.5	–	–
binkar10_1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	6746.76002	0.1
dano3mip	–	–	0.0	–	–	0.0	–	–	0.0	743.464286	6.5	276.1	743.133333	6.5
danoint	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	65.6666667	0.0
ds	5418.56016	1811.7	0.0	5418.56016	1811.7	0.4	5418.56016	1811.7	0.5	1437.2175	407.1	35.2	1045.71	268.9
glass4	–	–	0.0	–	–	0.0	–	–	0.0	4.3000342e+09	258.3	0.2	2.90001895e+09	141.7
liu	–	–	0.0	–	–	0.0	–	–	0.4	3132	167.2	5.0	3132	167.2
markshare1	854	Large	0.0	854	Large	0.0	854	Large	0.0	136	Large	0.0	136	Large
markshare2	1623	Large	0.0	1623	Large	0.0	1623	Large	0.0	212	Large	0.0	212	Large
mkc	–	–	0.0	–	–	0.0	-393.212	30.3	0.1	-533.432	5.4	0.6	-541.112	4.0
mkc1	–	–	0.0	–	–	0.0	–	–	0.3	-596.519	1.8	0.2	-596.519	1.8
momentum1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	17.0	–	–
momentum2	–	–	0.0	–	–	0.0	–	–	0.0	–	–	14.1	–	–
msc98-ip	–	–	0.0	–	–	0.0	–	–	0.1	–	–	0.4	–	–
neos616206	–	–	0.0	–	–	0.0	–	–	0.0	–	–	1.8	–	–
net12	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.3	–	–
nsrand-ipx	–	–	0.0	–	–	0.0	76800	50.0	0.3	61280	19.7	1.1	57120	11.6
opt1217	–	–	0.0	–	–	0.0	-14	12.5	0.0	-16	0.0	0.0	-16	0.0
protfold	–	–	0.0	–	–	0.0	–	–	0.0	–	–	3.8	–	–
rd-rplusc-21	–	–	0.0	–	–	0.0	–	–	0.0	–	–	4.3	–	–
roll3000	–	–	0.0	–	–	0.0	–	–	0.0	14945	15.6	1.6	13974	8.1
seymour	708	67.4	0.0	708	67.4	0.0	708	67.4	0.0	430	1.7	18.7	430	1.7
sp97ar	–	–	0.0	1.44316835e+09	117.2	0.1	1.44316835e+09	117.2	0.1	722387462	8.7	10.1	691318495	4.0
swath	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.2	–	–
swath3	–	–	0.0	–	–	0.0	–	–	0.0	–	–	57.4	–	–
t1717	–	–	0.0	–	–	0.1	–	–	0.2	–	–	18.0	–	–
timtab1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	902037	17.9
timtab2	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0	–	–
tr12-30	–	–	0.0	–	–	0.0	–	–	0.0	159967.543	22.5	2.9	131616	0.8
Total (129)			0.1			3.4			8.2			932.6		
Geom. Mean			1.0			1.0			1.0			2.0		

Table B.6. Comparison of different rounding heuristics applied to the optimum of the LP-relaxation

	Simple Rounding			Rounding			Shifting			RENS		
Name	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time
10teams	–	–	0.0	–	–	0.0	–	–	0.1	–	–	11.0
30:70:4_5:0_5:100	–	–	0.0	2196	Large	4.1	2196	Large	4.9	214	2277.8	21.2
30:70:4_5:0_95:98	–	–	0.0	2533	Large	4.5	2533	Large	4.5	155	1191.7	22.1
30:70:4_5:0_95:100	–	–	0.0	1996	Large	4.2	1996	Large	3.9	70	2233.3	20.5
acc-0	–	–	0.0	–	–	0.0	–	–	0.0	–	–	2.3
acc-1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	1.8
acc-2	–	–	0.0	–	–	0.0	–	–	0.0	–	–	10.5
acc-3	–	–	0.0	–	–	0.1	–	–	0.1	–	–	13.6
acc-4	–	–	0.0	–	–	0.0	–	–	0.1	–	–	9.7
acc-5	–	–	0.0	–	–	0.0	–	–	0.0	–	–	5.4
acc-6	–	–	0.0	–	–	0.0	–	–	0.0	–	–	6.7
aflow30a	–	–	0.0	–	–	0.0	–	–	0.1	1267	9.4	0.9
air03	–	–	0.0	1091684	220.9	0.2	1091684	220.9	0.2	394874	16.1	0.3
air04	–	–	0.0	–	–	0.9	–	–	1.3	–	–	7.0
air05	–	–	0.0	–	–	0.9	–	–	1.3	–	–	5.0
bc1	–	–	0.0	–	–	0.0	–	–	0.1	3.4198842	2.4	2.8
bell3a	880028.753	0.2	0.0	880028.753	0.2	0.0	880028.753	0.2	0.0	878430.316	0.0	0.1
bell5	–	–	0.0	–	–	0.0	9618234.44	7.3	0.0	–	–	0.0
bienst1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
bienst2	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
blend2	–	–	0.0	–	–	0.0	–	–	0.0	8.103921	6.6	0.0
cap6000	-2442801	0.3	0.0	-2442801	0.3	0.0	-2442801	0.3	0.9	-2443599	0.3	0.1
dano3_3	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.3
dano3_4	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.4
dano3_5	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.4
disctom	–	–	0.0	–	–	0.8	–	–	1.3	–	–	2.0
dcmulti	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
dsbmip	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
egout	579.718806	2.0	0.0	579.718806	2.0	0.0	579.718806	2.0	0.0	572.23465	0.7	0.0
eilD76	3630.85679	310.1	0.0	2987.4607	237.4	0.0	2987.4607	237.4	0.1	979.945566	10.7	1.0
enigma	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
fast0507	508	192.0	0.1	508	192.0	0.1	508	192.0	0.1	179	2.9	20.6
fiber	–	–	0.0	–	–	0.0	1237567.75	204.9	0.0	419895.5	3.4	0.1

continue next page

	Simple Rounding			Rounding			Shifting			Rens		
Name	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time
fixnet6	4536	13.9	0.0	4536	13.9	0.0	4536	13.9	0.0	3986	0.1	0.2
flugpl	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
gesa2-o	–	–	0.0	–	–	0.0	27997882.6	8.6	0.0	–	–	0.7
gesa2	–	–	0.0	–	–	0.0	26685746.9	3.5	0.0	25782398.1	0.0	0.5
gesa3	–	–	0.0	–	–	0.0	29346599.3	4.8	0.0	27991430	0.0	0.1
gesa3_o	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.2
gt2	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
irp	18142.1226	49.2	0.1	16555.0074	36.1	0.3	16555.0074	36.1	0.2	12409.4125	2.1	0.6
khb05250	111772902	4.5	0.0	111772902	4.5	0.0	111772902	4.5	0.0	106940226	0.0	0.1
l152lav	–	–	0.0	–	–	0.0	–	–	0.1	–	–	1.2
lseu	–	–	0.0	–	–	0.0	1405	25.4	0.0	1148	2.5	0.0
mas74	–	–	0.0	–	–	0.0	13755.8924	16.6	0.0	14372.8713	21.8	0.0
mas76	–	–	0.0	–	–	0.0	46793.6933	17.0	0.0	40560.0518	1.4	0.0
mas284	–	–	0.0	–	–	0.0	103611.404	13.4	0.0	95703.3781	4.7	0.3
misc03	–	–	0.0	–	–	0.0	–	–	0.1	–	–	0.0
misc06	12885.976	0.3	0.0	12885.976	0.3	0.0	12885.976	0.3	0.0	12852.2468	0.0	0.1
misc07	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
mitre	–	–	0.1	115185	0.0	0.1	115185	0.0	1.1	115170	0.0	0.3
mod008	510	66.1	0.0	432	40.7	0.0	432	40.7	0.0	308	0.3	0.0
mod011	–	–	0.0	–	–	0.0	–	–	0.4	–	–	0.2
modglob	20786787	0.2	0.0	20786787	0.2	0.0	20786787	0.2	0.0	20930259.7	0.9	0.6
mzzv11	–	–	0.0	–	–	0.1	–	–	0.5	–	–	0.3
mzzv42z	–	–	0.0	–	–	0.0	–	–	0.5	–	–	0.3
neos1	–	–	0.0	31	63.2	0.0	28	47.4	0.0	20	5.3	0.1
neos2	–	–	0.0	–	–	0.0	–	–	0.1	–	–	0.4
neos3	–	–	0.0	–	–	0.0	–	–	0.1	–	–	0.1
neos5	–	–	0.1	–	–	0.1	–	–	0.1	-4.86034408e+10	0.0	0.8
neos6	–	–	0.0	–	–	0.0	–	–	0.5	–	–	2.5
neos7	–	–	0.0	–	–	0.0	–	–	0.0	721934	0.0	0.8
neos10	–	–	0.0	50	104.4	0.1	50	104.4	0.5	-372	67.2	0.1
neos11	–	–	0.0	–	–	0.0	–	–	0.0	–	–	3.5
neos13	–	–	0.0	–	–	0.0	–	–	4.1	-62.3278436	34.7	34.4
neos21	–	–	0.0	30	328.6	0.0	30	328.6	0.0	11	57.1	1.6

continue next page

	Simple Rounding			Rounding			Shifting			RENS		
Name	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time
neos22	–	–	0.0	–	–	0.0	786525	0.9	0.0	782405.625	0.3	3.4
neos632659	–	–	0.0	–	–	0.0	6	106.4	0.0	-94	0.0	0.2
noswot	–	–	0.0	–	–	0.0	-39	4.9	0.0	–	–	0.2
nug08	–	–	0.0	–	–	0.2	–	–	0.4	–	–	8.4
nw04	–	–	0.0	22494	33.4	1.8	22494	33.4	9.6	22494	33.4	2.8
p0201	–	–	0.0	–	–	0.0	–	–	0.0	7905	3.8	0.5
p0282	–	–	0.0	307830	19.1	0.0	270471	4.7	0.0	258945	0.2	0.1
p0548	–	–	0.0	–	–	0.0	–	–	0.1	8763	0.8	0.0
p2756	–	–	0.0	–	–	0.0	3136	0.4	0.1	3359	7.5	0.1
pk1	–	–	0.0	–	–	0.0	102	827.3	0.0	26	136.4	0.2
pp08a	12083.8672	64.4	0.0	12083.8672	64.4	0.0	11445.4976	55.7	0.0	7660	4.2	0.7
pp08aCUTS	11481.4676	56.2	0.0	11481.4676	56.2	0.0	11481.4676	56.2	0.0	7350	0.0	1.2
prod1	–	–	0.0	–	–	0.0	-49	12.5	0.0	–	–	0.5
qap10	–	–	0.0	–	–	1.6	–	–	2.9	–	–	56.6
qiu	1805.17714	1458.6	0.0	1805.17714	1458.6	0.0	1805.17714	1458.6	0.0	–	–	0.0
qnet1	–	–	0.0	41530.6864	159.1	0.0	31236.0264	94.9	0.0	21159.9639	32.0	0.0
qnet1_o	28462.14	77.6	0.0	27183.378	69.6	0.0	25701.758	60.3	0.0	19351.9	20.7	0.1
ran8x32	5837	11.2	0.0	5837	11.2	0.0	5837	11.2	0.0	5329	1.6	0.3
ran10x26	4745	11.1	0.0	4745	11.1	0.0	4745	11.1	0.0	4459	4.4	1.2
ran12x21	4080	11.4	0.0	4080	11.4	0.0	4080	11.4	0.0	3782	3.2	1.1
ran13x13	3521	8.3	0.0	3521	8.3	0.0	3521	8.3	0.0	3636	11.8	0.6
rentacar	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.1
rgn	–	–	0.0	–	–	0.0	–	–	0.0	82.1999991	0.0	0.1
rout	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.7
set1ch	57621.6608	5.7	0.0	57621.6608	5.7	0.0	57621.6608	5.7	0.0	54628	0.2	0.0
seymour1	449.489972	9.4	0.0	449.489972	9.4	0.0	449.489972	9.4	0.0	–	–	0.1
stein27	27	50.0	0.0	21	16.7	0.0	21	16.7	0.0	–	–	0.0
stein45	45	50.0	0.0	40	33.3	0.0	40	33.3	0.0	–	–	0.0
swath1	–	–	0.0	–	–	0.0	–	–	0.1	–	–	79.4
swath2	–	–	0.0	–	–	0.0	–	–	0.1	–	–	84.9
vpm2	–	–	0.0	–	–	0.0	17.25	25.5	0.0	13.75	0.0	0.5
a1c1s1	–	–	0.0	–	–	0.1	20811.88	79.5	0.2	–	–	0.0
aflow40b	–	–	0.0	–	–	0.0	–	–	0.1	1318	12.8	1.4

continue next page

	Simple Rounding			Rounding			Shifting			Rens		
Name	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time	Primal Bound	P Gap	Time
arki001	–	–	0.0	–	–	0.0	–	–	0.0	–	–	1.8
atlanta-ip	–	–	0.0	–	–	0.0	–	–	1.3	–	–	0.5
binkar10_1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
dano3mip	–	–	0.0	–	–	0.0	–	–	0.1	743.464286	6.5	276.1
danoint	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
ds	5418.56016	1811.7	0.2	5281.30516	1763.3	1.5	5281.30516	1763.3	1.6	1437.2175	407.1	35.2
glass4	–	–	0.0	–	–	0.0	–	–	0.0	4.3000342e+09	258.3	0.2
liu	–	–	0.0	–	–	0.0	4130	252.4	0.1	3132	167.2	5.0
markshare1	854	Large	0.0	271	Large	0.0	271	Large	0.0	136	Large	0.0
markshare2	1623	Large	0.0	1211	Large	0.0	1211	Large	0.0	212	Large	0.0
mkc	–	–	0.0	–	–	0.1	-431.712	23.4	0.9	-533.432	5.4	0.6
mkc1	–	–	0.0	–	–	0.0	–	–	3.4	-596.519	1.8	0.2
momentum1	–	–	0.0	–	–	0.0	–	–	0.1	–	–	17.0
momentum2	–	–	0.0	–	–	0.0	–	–	0.2	–	–	14.1
msc98-ip	–	–	0.0	–	–	0.1	–	–	0.5	–	–	0.4
neos616206	–	–	0.0	–	–	0.0	–	–	0.0	–	–	1.8
net12	–	–	0.0	–	–	0.0	–	–	0.2	–	–	0.3
nsrand-ipx	–	–	0.0	–	–	0.1	71360	39.4	1.9	61280	19.7	1.1
opt1217	–	–	0.0	–	–	0.0	-14	12.5	0.0	-16	0.0	0.0
protfold	–	–	0.0	–	–	0.0	–	–	0.0	–	–	3.8
rd-rplusc-21	–	–	0.0	–	–	0.0	–	–	0.2	–	–	4.3
roll3000	–	–	0.0	–	–	0.0	–	–	0.1	14945	15.6	1.6
seymour	707	67.1	0.0	702	66.0	0.2	702	66.0	0.3	430	1.7	18.7
sp97ar	–	–	0.0	1.37580174e+09	107.0	0.3	1.37580174e+09	107.0	0.4	722387462	8.7	10.1
swath	–	–	0.0	–	–	0.0	–	–	0.3	–	–	0.2
swath3	–	–	0.0	–	–	0.0	–	–	0.2	–	–	57.4
t1717	–	–	0.0	–	–	0.6	–	–	1.4	–	–	18.0
timtab1	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
timtab2	–	–	0.0	–	–	0.0	–	–	0.1	–	–	0.0
tr12-30	–	–	0.0	–	–	0.0	163290.448	25.0	0.0	159967.543	22.5	2.9
Total (129)			0.6			23.6			54.5			932.6
Geom. Mean			1.0			1.0			1.1			2.0

Table B.7. Comparison of different rounding heuristics applied to every LP-feasible point of the LP-solving loop

	Objective Ray		Average Ray		Avg Weighted Slack Ray		Avg Normal Ray		SCIP root node heuristics	
Name	P Bound	P Gap%	P Bound	P Gap%	P Bound	P Gap%	P Bound	P Gap%	Primal Bound	P Gap%
air03	–	–	–	–	–	–	**997224**	193.2	727590	113.9
cap6000	**-2443599**	0.3	-2442801	0.3	-2442801	0.3	**-2443599**	0.3	-2442801	0.3
eilD76	**2836.25525**	220.3	–	–	–	–	**836.25525**	220.3	2987.4607	237.4
fast0507	554	218.4	570	227.6	–	–	**524**	201.1	178	2.3
fiber	**973221.9**	139.7	–	–	–	–	–	–	957344.94	135.8
irp	16146.0316	32.8	–	–	**13938.2651**	14.6	14677.6318	20.9	16266.3441	33.8
mod008	978	218.6	950	209.4	**308**	0.3	874	184.7	369	20.2
neos21	–	–	**31**	342.9	–	–	–	–	12	71.4
nw04	**22494**	33.4	–	–	–	–	**22494**	33.4	22494	33.4
p0282	396019	53.3	**259966**	0.6	**259966**	0.6	384266	48.7	307830	19.1
p2756	**3357**	7.5	–	–	–	–	3360	7.6	49141	1473.0
stein27	**25**	38.9	**25**	38.9	–	–	**25**	38.9	19	5.6
stein45	**43**	43.3	**43**	43.3	–	–	**43**	43.3	37	23.3
ds	5108.14516	1702.2	–	–	–	–	**4388.21516**	1448.2	5281.30516	1763.3
markshare1	381	Large	–	–	270	Large	**228**	Large	305	Large
markshare2	480	Large	–	–	**416**	Large	**416**	Large	452	Large
seymour	723	70.9	722	70.7	–	–	**693**	63.8	709	67.6
Solution found		**15**		8		6		**15**		
Best solution		7		4		4		**11**		

Table B.8. Comparison of different ray directions

	Octane deactivated		Octane without Diff Ray		Octane with Diff Ray		Octane only at root	
Name	Nodes	Time	Nodes	Time	Nodes	Time	Nodes	Time
10teams	324	31.1	324	31.5	324	31.2	324	31.7
30:70:4_5:0_5:100	192	505.3	192	519.6	192	521.7	192	519.9
30:70:4_5:0_95:98	125	293.6	125	301.9	125	300.3	125	298.8
30:70:4_5:0_95:100	109	305.9	109	313.6	109	312.4	109	312.6
acc-0	1	14.9	1	14.9	1	14.8	1	14.9
acc-1	1	29.2	1	29.6	1	29.7	1	29.5
acc-2	79	98.3	79	99.6	79	99.2	79	99.6
acc-3	75	185.3	75	186.1	75	186.6	75	184.8
acc-4	236	414.8	236	421.4	236	419.0	236	415.1
acc-5	8592	2138.4	8592	2113.8	8592	2125.6	8592	2122.9
acc-6	18	80.6	18	82.4	18	82.5	18	82.2
air03	2	22.9	2	26.0	2	25.1	2	25.8
air04	153	167.2	153	169.0	153	168.0	153	167.8
air05	239	98.0	239	98.8	239	98.9	239	98.9
cap6000	2585	24.4	2585	31.3	2585	33.2	2585	24.9
disctom	1	67.0	1	67.2	1	67.0	1	67.0
eilD76	1359	103.2	1359	105.7	1359	104.8	1359	104.2
enigma	1408	0.8	1408	0.8	1408	0.9	1408	0.8
fiber	42	5.1	42	5.1	42	5.1	42	5.2
irp	327	47.5	327	59.9	327	59.4	327	49.5
l152lav	63	7.2	63	7.3	63	7.4	63	7.3
lseu	302	0.3	302	0.3	302	0.3	302	0.3
misc03	52	1.9	52	2.0	52	2.0	52	1.9
misc07	34657	46.6	34657	47.5	34657	47.6	34657	46.4
mitre	5	70.1	5	70.4	5	70.1	5	70.3
mod008	228	0.5	228	0.9	228	1.0	228	0.6
neos1	1	5.9	1	6.0	1	6.0	1	5.9
neos21	2450	42.1	2450	47.0	2450	47.4	2450	42.5
nug08	3	79.7	3	80.0	3	80.3	3	80.3
nw04	7	161.0	7	165.2	7	167.0	7	166.7
p0201	217	2.2	217	2.3	217	2.3	217	2.2
p0282	71	0.7	61	0.9	61	0.8	71	0.8
p0548	63	0.9	46	0.9	46	0.9	63	0.9
p2756	185	5.5	185	7.8	185	8.0	185	6.6
qap10	5	445.7	5	454.6	5	448.3	5	445.3
stein27	4165	1.4	4165	3.9	4165	4.7	4165	1.4
stein45	52305	41.5	53165	58.5	53165	59.5	52305	40.8
Total (37)	110647	5546.9	111480	5633.3	111480	5638.8	110647	5576.5
Geom. Mean	103	28.4	102	30.6	102	30.8	103	28.8

Table B.9. Integration of Octane into SCIP (easy instances)

	Octane deactivated		Octane without Diff Ray		Octane with Diff Ray		Octane only at root	
Name	Primal Bound	Nodes	Primal Bound	Nodes	Primal Bound	Nodes	Primal Bound	Nodes
ds	305.105	688	305.105	657	305.105	655	305.105	672
fast0507	177	1022	177	1070	177	1079	177	1022
markshare1	6	27406458	6	27406449	6	27406448	6	27406456
markshare2	18	18232531	16	17651861	16	17651872	18	18232528
protfold	1e+20	713	1e+20	684	1e+20	722	1e+20	726
seymour	427	4942	427	4796	427	4825	427	4950
sp97ar	686076608	11416	686076608	10900	686076608	11177	686076608	11347
t1717	180160	299	180160	287	180160	301	180160	305
Total (8)		45658069		45076704		45077079		45658006
Geom. Mean		15967		15590		15861		15987

Table B.10. Integration of Octane into SCIP (hard instances)

	No LNS		Local Branching			Rins			Crossover			Mutation		
Name	Nodes	Time	Nodes	Time	HeurTime	Nodes	Time	HeurTime	Nodes	Time	HeurTime	Nodes	Time	HeurTime
10teams	324	32.3	324	31.6	0.1	324	31.5	0.0	324	31.6	0.0	324	32.0	0.0
acc-4	236	420.3	236	421.9	0.0	236	418.1	0.0	236	419.8	0.0	236	423.6	0.0
acc-5	5011	1475.3	5011	1470.5	0.0	5011	1457.9	0.0	5011	1469.6	0.0	5011	1461.6	0.0
aflow30a	9408	57.8	9408	64.8	7.0	7740	49.1	0.9	6026	35.5	0.4	8872	48.9	0.3
air05	239	98.6	239	98.7	0.0	239	98.6	0.0	239	98.2	0.0	239	98.5	0.0
bc1	17114	753.6	17114	819.1	64.0	17114	776.2	18.5	16043	733.6	29.8	16443	748.5	17.8
bell3a	48908	43.2	48908	43.4	0.0	48682	46.4	4.1	48995	45.4	2.5	48908	46.8	3.6
bell5	1170	1.1	1170	1.1	0.0	1170	1.1	0.0	1170	1.2	0.1	1170	1.2	0.1
bienst1	9574	48.0	9574	48.0	0.0	9638	55.8	6.9	9574	48.6	0.0	11797	61.4	4.8
bienst2	101293	607.7	101293	601.6	0.0	101293	622.5	18.6	101372	608.1	0.5	100584	618.0	13.7
blend2	1066	3.3	1066	4.5	0.9	1520	4.1	0.6	879	3.5	0.3	1066	3.4	0.0
cap6000	2937	36.9	2937	37.2	0.4	2937	37.0	0.2	2937	37.0	0.2	2937	37.4	0.4
eilD76	1359	108.2	1359	121.3	11.9	1359	109.0	0.2	1359	108.0	0.1	1359	108.3	0.1
enigma	4455	1.6	4455	1.5	0.0	4455	1.6	0.0	4455	1.5	0.0	4455	1.5	0.0
fast0507	1488	2451.7	1488	2479.4	29.6	1488	2469.8	21.2	1488	2465.3	2.5	1488	2466.8	3.2
flugpl	474	0.4	474	0.4	0.0	474	0.4	0.0	474	0.4	0.0	474	0.3	0.0
gesa2-o	1656	12.4	1656	12.4	0.1	1495	12.5	0.3	1604	12.4	0.2	1656	12.6	0.1
gesa3_o	725	10.7	725	10.7	0.1	725	10.6	0.1	725	10.7	0.1	725	10.8	0.1
irp	312	59.7	312	74.7	15.5	312	59.6	0.2	312	60.0	0.2	312	59.6	0.2
lseu	302	0.8	302	0.8	0.0	302	0.8	0.0	302	0.8	0.0	302	0.8	0.0
mas74	5206250	1660.9	5206250	1687.0	53.9	3612338	1178.3	30.3	4856815	1551.7	31.7	4842766	1575.0	77.3
mas76	618955	211.9	618955	228.6	13.4	326851	110.5	10.4	347300	104.7	0.3	460271	167.9	3.5
mas284	17379	27.9	17379	31.3	3.6	17379	29.4	1.4	17379	31.2	3.5	17379	28.5	0.7
misc07	34963	47.6	34963	50.5	3.2	34963	46.7	0.2	34963	46.8	0.2	34963	47.1	0.5
mod008	212	0.8	212	1.1	0.3	212	0.8	0.0	212	0.8	0.0	212	0.7	0.0
mod011	2449	164.7	2449	214.3	49.5	2260	168.8	19.6	2449	170.8	4.8	2449	168.1	1.0
modglob	4283	7.7	4283	9.6	1.8	4389	9.2	0.9	3815	6.8	0.5	4224	8.0	0.4
mzzv11	4261	1510.2	4338	1626.2	97.3	3369	1225.5	2.9	2541	1165.1	0.6	4261	1485.3	1.1
mzzv42z	1744	731.3	1744	813.5	71.9	1688	718.1	1.2	1452	678.1	0.5	1744	744.0	0.7
neos2	51967	199.5	51967	211.4	10.8	45449	186.4	5.0	51967	201.6	1.2	51967	205.4	4.5
neos3	784004	3600.0	783940	3600.0	159.5	828482	3600.0	22.6	508140	2391.8	8.1	768020	3600.0	24.6

continue next page

	No LNS		Local Branching			Rins			Crossover			Mutation		
Name	Nodes	Time	Nodes	Time	HeurTime	Nodes	Time	HeurTime	Nodes	Time	HeurTime	Nodes	Time	HeurTime
neos6	3738	395.9	3738	489.8	94.5	3738	398.6	3.8	3738	398.5	2.5	3738	396.6	1.3
neos7	51431	552.8	51431	554.2	3.5	51431	553.2	2.9	51407	549.0	7.2	51431	542.0	1.6
neos11	6790	793.2	6790	795.4	10.2	7000	796.4	4.0	6796	788.6	0.2	6790	801.3	0.3
neos13	19071	1198.3	19071	1258.3	53.1	13862	1016.8	59.3	11242	793.6	8.9	7770	635.5	8.9
neos21	2223	41.9	2223	45.4	3.7	2223	43.3	1.2	2223	41.9	0.1	2223	41.7	0.1
neos22	32200	289.5	32200	292.3	2.6	32876	344.9	12.2	35891	399.3	14.3	29486	288.3	4.0
neos632659	31971	21.2	31971	22.2	1.4	31971	21.3	0.5	31971	21.1	0.1	31971	21.5	0.6
p0201	262	2.4	262	2.4	0.0	262	2.4	0.0	262	2.4	0.0	262	2.4	0.0
pk1	293834	130.0	293834	136.2	7.1	356629	172.8	8.4	293834	130.9	1.8	293965	148.8	19.1
pp08a	1852	3.9	1852	6.3	2.4	1829	4.5	0.6	1817	4.1	0.2	1929	4.0	0.1
pp08aCUTS	2374	7.2	2374	9.4	2.2	1853	6.5	0.5	2400	7.7	0.4	2374	7.4	0.2
prod1	59396	46.9	61929	51.3	1.6	59736	49.3	0.8	59023	48.7	0.7	59396	47.7	0.6
qiu	10584	186.5	10584	194.9	8.7	10584	192.8	8.7	10584	186.4	0.1	10584	187.0	0.6
ran8x32	14780	26.6	14780	29.3	2.9	14983	25.3	0.5	15693	28.8	0.5	14780	27.0	0.2
ran10x26	36989	78.0	36989	85.8	8.2	30491	69.2	1.3	29868	67.0	0.9	42665	88.0	1.1
ran12x21	132269	233.1	132269	253.3	21.8	124737	231.1	3.7	136566	240.9	4.2	133245	235.0	2.0
ran13x13	72276	87.5	72276	96.4	9.9	100478	117.0	2.5	67410	82.8	2.2	76602	93.0	1.1
rgn	2341	1.1	2341	1.4	0.4	2349	1.1	0.1	2341	1.1	0.1	2341	1.1	0.0
rout	33069	74.4	33069	80.2	5.8	23572	56.3	0.7	32398	69.0	0.5	33069	75.4	0.5
seymour1	4837	1010.6	4837	1019.2	7.9	4837	1037.6	25.2	4564	994.4	23.0	4813	1012.4	13.7
stein27	4173	4.0	4173	3.8	0.0	4173	4.0	0.1	4173	3.9	0.0	4173	3.9	0.0
stein45	53354	58.8	53354	61.6	2.4	53354	62.8	3.3	53354	57.3	0.1	53354	58.6	0.5
swath1	2435	100.7	2435	108.6	8.2	2435	102.2	1.1	2435	101.7	0.6	2435	101.5	0.4
swath2	18520	226.6	18520	252.4	26.1	12993	190.6	5.6	12543	182.3	3.0	18520	228.6	2.7
swath3	131638	1004.2	131638	1099.6	99.6	97650	800.5	15.2	181669	1330.1	12.2	131848	1027.5	23.1
vpm2	10724	9.8	10724	10.9	1.1	10724	10.1	0.5	9890	9.6	0.3	10724	9.9	0.2
Total (57)	7967649	20970.9	7970195	21777.9	979.9	6140664	19846.8	329.1	7094650	19081.6	172.2	7427102	20357.8	241.8
Geom. Mean	3579	57.9	3581	60.9	3.1	3473	57.2	2.1	3404	56.0	1.5	3535	57.8	1.6

Table B.11. The overall performance of SCIP with different LNS heuristics, easy instances

	No LNS		Local Branching			Rins			Crossover			Mutation		
Name	Primal Bound	γ_{PD}	Primal Bound	γ_{PD}	HeurTime	Primal Bound	γ_{PD}	HeurTime	Primal Bound	γ_{PD}	HeurTime	Primal Bound	P-D Gap%	HeurTime
a1c1s1	12664.722	44.0	12664.722	44.1	57.8	12664.722	44.2	132.1	12643.7169	43.2	39.5	12382.4229	40.6	34.8
aflow40b	1299	13.7	1299	13.8	305.8	1213	5.9	15.7	1285	12.4	2.9	1299	13.7	11.9
arki001	7585236.61	0.1	7585236.61	0.1	10.3	7582538.72	0.0	54.7	7584890.09	0.1	81.3	7584383.5	0.1	606.9
atlanta-ip	–	–	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
binkar10_1	6747.31001	0.2	6747.31001	0.2	4.5	6746.76002	0.2	28.0	6743.24002	0.2	22.8	6746.76002	0.2	7.5
dano3mip	767.043478	33.0	767.043478	33.0	957.9	767.043478	33.0	285.2	767.043478	33.0	0.0	766.954545	32.9	5.3
danoint	66.5	4.5	65.6666667	2.9	76.6	66.375	4.2	99.8	66.5	4.5	0.3	66.25	4.1	4.6
ds	360.715161	514.3	411.835161	607.4	1709.4	360.715161	514.3	22.8	360.715161	514.3	1.6	360.715161	514.3	1.7
glass4	1.800015e+09	124.8	1.76668522e+09	120.7	166.6	1.80001365e+09	125.0	51.0	1.78573286e+09	123.2	83.5	1.80001452e+09	125.0	48.8
liu	2771.99994	395.0	2596	363.6	254.6	2449.99999	337.5	331.2	2771.99994	395.0	0.7	2132	280.7	81.2
markshare1	1	–	1	–	22.9	1	–	15.3	1	–	11.8	1	–	249.9
markshare2	16	–	16	–	19.5	12	–	30.7	16	–	6.7	16	–	191.7
mkc	-551.012	2.4	-548.844	2.8	845.7	-552.15	2.2	23.6	-551.412	2.4	8.1	-551.282	2.4	23.0
mkc1	-606.56	0.1	-606.66	0.1	287.4	-606.56	0.1	4.7	-606.717	0.1	9.3	-606.46	0.1	47.6
momentum1	–	–	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
momentum2	–	–	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
msc98-ip	–	–	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
neos616206	937.6	2.4	937.6	2.4	44.0	937.866667	1.8	25.7	937.6	2.4	3.3	937.6	2.4	11.5
net12	337	175.0	337	175.0	17.9	255	108.0	7.0	337	175.1	0.0	337	174.1	0.6
noswot	-41	4.7	-41	4.7	10.3	-41	4.7	28.9	-41	4.7	9.2	-41	4.7	77.5
nsrand-ipx	55520	9.5	55520	9.5	329.1	55360	9.2	14.8	54400	7.3	29.3	55840	10.2	36.8
opt1217	-16	16.8	-16	16.8	31.7	-16.000007	16.8	32.3	-16.000007	16.8	16.2	-16.000001	16.8	46.5
protfold	–	–	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
rd-rplusc-21	–	–	–	–	0.0	–	–	0.0	–	–	0.0	–	–	0.0
roll3000	12968	1.7	12968	1.7	78.2	13006	2.0	20.6	12933	1.5	10.8	12968	1.7	7.1
seymour	425	2.7	425	2.7	1.7	427	3.2	20.9	426	3.0	5.5	425	2.7	0.4
sp97ar	689918770	5.1	689918770	5.1	255.8	686756724	4.7	24.3	689918770	5.1	23.0	699799048	6.6	25.9
swath	519.023309	33.1	480.36465	23.0	327.5	502.07608	28.8	7.4	518.076418	32.7	5.7	519.023309	33.1	25.1
t1717	190337	40.2	190337	40.2	266.9	190337	40.2	8.2	190337	40.2	0.9	190337	40.2	1.3
timtab1	909726	36.5	909726	36.7	140.5	909726	36.6	39.0	909726	36.5	2.0	903265	36.6	645.9
timtab2	1606961	160.7	1606961	161.1	100.3	1606961	161.0	70.6	1606961	160.7	0.6	1563771	154.1	103.4
tr12-30	130658	0.1	130658	0.1	5.4	130658	0.1	27.2	130658	0.1	48.7	130643	0.1	22.7
Total (32)					6328.5			1421.8			423.7			2319.6
Geom. Mean					33.9			15.8			4.6			12.7

Table B.12. The overall performance of SCIP with different LNS heuristics, hard instances

Name	SimRou	Round	Shift	FeasP	CoefDi	PsCstDi	FracDi	VecDi	OPsDi	RooSoD	LineDi	GuiDi	Octane	CrOv	Rens	ALL
10teams	1.01	1.01	0.98	*0.58*	1.01	1.03	1.06	1.01	1.00	1.00	1.00	1.00	0.99	1.02	0.98	1.06
30:70:4_5:0_5:100	1.02	0.98	0.98	1.01	1.14	1.95	1.17	1.09	1.01	1.00	1.00	1.00	0.98	1.16	1.01	**1.62**
30:70:4_5:0_95:98	1.02	1.00	0.98	1.06	1.01	1.05	1.07	1.06	1.00	1.00	1.00	1.00	0.98	1.03	0.95	**1.72**
30:70:4_5:0_95:100	1.02	0.99	0.98	1.02	1.01	1.05	1.11	1.02	1.00	1.00	1.00	0.99	0.98	1.01	0.97	**1.78**
acc-0	1.03	1.02	0.97	1.94	1.02	1.06	1.10	1.02	1.00	1.00	1.01	1.00	0.99	1.01	1.00	**4.10**
acc-1	1.02	1.02	0.97	1.91	1.01	1.06	1.08	**1.24**	0.99	0.99	0.99	1.00	0.99	1.00	0.99	**4.01**
acc-2	1.01	1.01	0.97	1.18	1.00	1.04	1.08	1.13	0.99	0.99	0.99	0.99	0.98	1.00	0.90	**1.95**
acc-3	1.03	1.02	1.00	1.00	1.02	1.06	1.10	1.03	1.00	1.00	1.01	1.00	0.99	1.01	0.91	**1.95**
acc-4	1.04	1.01	0.98	1.08	1.02	1.06	1.13	1.08	1.00	1.00	0.99	1.00	0.99	1.00	0.96	**2.12**
acc-5	1.03	1.01	0.98	1.15	*0.68*	*0.67*	**1.22**	1.05	*0.66*	1.17	**1.45**	1.02	0.99	1.00	0.98	>**2.45**
acc-6	1.04	1.01	0.97	0.96	1.01	1.05	**1.25**	1.05	0.98	0.98	0.98	0.99	0.99	0.98	0.92	**1.99**
aflow30a	1.04	1.01	0.98	1.33	**1.27**	1.15	**1.23**	1.15	0.87	0.90	0.96	1.03	1.00	**1.61**	1.35	**2.74**
air03	1.03	1.01	0.98	1.05	1.02	1.05	1.12	1.05	1.01	1.02	1.01	1.00	0.90	1.01	1.00	**1.78**
air04	1.02	1.00	0.97	0.85	1.05	1.06	1.08	1.11	1.00	0.99	0.99	0.99	0.99	1.00	0.98	**1.61**
air05	1.06	1.01	0.98	1.15	1.03	1.06	1.11	1.13	1.01	1.00	1.01	1.01	1.00	1.00	0.94	**1.85**
bc1	1.04	1.03	0.98	1.11	0.99	1.12	1.11	1.05	1.01	1.00	1.01	1.03	1.02	1.03	1.15	**2.04**
bell3a	1.10	1.02	0.98	1.08	0.99	1.07	1.03	1.05	1.00	0.98	1.04	0.92	0.99	0.95	0.98	**1.23**
bell5	1.00	1.00	0.94	1.01	0.99	1.00	0.97	0.99	0.98	0.99	1.00	1.01	1.00	0.97	0.98	1.16
bienst1	1.02	1.01	0.99	1.08	0.96	1.17	1.67	0.96	0.89	0.98	1.00	0.97	0.99	0.99	1.00	**1.34**
bienst2	1.03	1.01	0.99	**1.20**	1.02	0.89	**1.20**	**1.47**	1.06	1.10	1.19	0.99	1.00	1.00	0.99	**1.72**
blend2	1.02	0.98	0.95	0.67	0.96	0.98	**1.20**	1.12	1.12	1.10	0.98	0.98	0.98	0.96	**2.43**	**3.53**
cap6000	1.06	1.12	0.98	1.04	0.94	1.02	1.13	0.93	0.94	1.01	1.01	0.99	*0.74*	1.00	1.01	>**94.81**
dano3_3	1.07	1.01	0.99	0.90	1.02	1.05	1.02	1.02	1.00	1.00	0.99	1.00	1.00	1.00	0.99	**1.90**
dano3_4	1.05	1.02	0.99	0.98	0.88	1.05	1.03	1.09	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.68**
dano3_5	1.03	1.01	0.98	0.90	1.00	0.98	0.99	0.96	0.86	0.94	0.99	0.99	1.00	1.00	0.99	**1.64**
disctom	1.02	1.01	0.99	>**52.68**	1.00	1.04	1.02	1.09	1.00	0.99	1.00	1.00	1.00	1.00	1.00	>**52.68**
dcmulti	1.01	1.03	0.99	1.09	1.02	1.05	1.01	1.02	1.01	1.00	1.01	1.01	1.00	1.00	1.00	**1.50**
dsbmip	0.99	1.01	0.98	1.73	0.99	1.02	0.99	0.99	0.99	0.99	0.99	0.99	1.00	1.00	0.98	**2.53**
egout	1.00	1.00	1.01	1.01	1.01	1.01	1.00	1.01	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00
eilD76	1.01	1.01	0.97	1.24	1.01	1.06	1.02	1.03	0.99	0.98	0.99	1.01	0.97	1.00	0.96	**1.53**
enigma	1.01	0.97	0.98	0.96	*0.61*	*0.70*	*0.73*	*0.73*	*0.65*	0.95	*0.80*	1.00	0.98	1.02	0.99	*0.71*
fast0507	>**1.46**	>**1.46**	>**1.46**	*0.73*	1.19	*0.65*	>**1.46**	>**1.46**	>**1.46**	>**1.47**	>**1.46**	*0.68*	0.94	0.99	0.99	>**1.46**
fiber	1.05	1.02	1.02	0.94	1.02	1.06	**1.23**	1.02	1.00	1.00	1.00	1.01	0.91	1.01	**1.42**	**1.98**

continue next page

Name	SimRou	Round	Shift	FeasP	CoefDi	PsCstDi	FracDi	VecDi	OPsDi	RooSoD	LineDi	GuiDi	Octane	CrOv	Rens	ALL
fixnet6	1.03	1.02	0.99	0.81	1.01	1.04	1.19	1.00	0.98	1.01	1.00	1.00	1.00	1.00	1.02	**1.28**
flugpl	1.01	1.01	0.99	*0.78*	1.01	1.01	1.07	1.01	1.00	0.99	1.00	1.00	1.01	1.01	0.99	0.82
gesa2-o	1.05	1.02	0.98	0.98	0.98	**1.21**	**1.36**	1.01	0.92	0.92	1.06	1.06	1.02	1.00	0.95	**1.84**
gesa2	1.06	1.04	0.92	0.88	1.02	1.05	1.05	1.02	0.99	0.99	1.00	1.00	1.02	1.00	**1.32**	**2.53**
gesa3	1.06	1.03	0.98	0.95	1.02	1.05	1.02	1.01	1.04	1.01	1.02	1.02	1.01	1.00	**2.46**	**4.97**
gesa3_o	1.09	1.01	0.98	1.05	1.02	1.04	0.99	1.06	0.98	0.96	1.05	1.03	1.00	1.00	1.00	**1.98**
gt2	1.03	1.01	1.01	1.01	1.13	0.97	1.01	1.01	1.01	1.01	1.00	1.00	1.00	1.02	1.00	**1.36**
irp	1.10	0.99	0.96	1.03	1.00	1.04	1.02	1.06	0.99	0.99	0.99	0.99	*0.76*	1.00	**1.56**	**1.50**
khb05250	**1.21**	1.03	1.01	1.06	1.02	1.03	1.01	1.01	1.02	1.01	1.01	1.01	1.02	1.01	1.04	**1.43**
l152lav	1.12	1.01	0.99	1.06	1.03	1.07	1.04	1.03	1.01	1.00	1.00	1.01	1.00	1.01	*0.80*	**1.28**
lseu	1.06	0.99	0.94	1.07	0.99	1.02	1.01	1.01	1.00	0.98	1.00	0.99	0.97	0.98	*0.79*	0.84
mas74	1.04	1.00	*0.72*	1.16	*0.76*	0.87	1.01	0.81	*0.62*	0.87	*0.63*	0.91	0.99	1.07	0.99	>**2.32**
mas76	1.07	0.97	1.11	**1.59**	0.96	1.02	1.05	0.96	0.95	1.00	0.93	0.96	0.99	**2.01**	**1.72**	**6.90**
mas284	1.06	1.00	0.93	**1.20**	1.12	1.08	1.07	0.99	0.91	0.97	1.08	0.98	1.02	0.90	1.02	**1.75**
misc03	1.03	1.01	1.00	1.05	1.01	1.04	1.02	1.01	1.00	1.00	1.00	1.00	0.98	1.00	*0.79*	0.89
misc06	1.02	1.01	0.99	1.05	0.96	0.97	0.99	1.05	0.98	0.97	0.99	0.99	0.99	1.03	0.96	**1.35**
misc07	1.08	1.01	0.99	**1.35**	1.06	1.05	1.03	1.10	1.03	1.10	1.08	1.04	1.00	1.02	1.01	**1.95**
mitre	1.06	1.02	*0.58*	1.11	1.02	1.06	**1.21**	1.10	1.01	1.00	1.00	1.01	1.00	0.99	1.01	**5.84**
mod008	1.07	1.07	1.03	1.06	1.00	1.03	1.14	1.17	1.01	1.01	1.00	1.00	0.83	0.99	1.00	**2.62**
mod011	1.07	1.02	0.99	1.11	0.96	0.98	1.05	1.05	0.82	0.82	0.99	1.04	1.01	0.96	1.01	**2.25**
modglob	1.05	1.02	0.98	0.95	0.95	1.09	0.98	0.92	0.83	0.84	0.98	0.99	1.01	1.11	0.83	**1.35**
mzzv11	1.05	0.99	0.97	1.04	**1.40**	1.13	**1.47**	1.16	1.08	1.36	1.02	1.04	0.99	**1.30**	1.00	>**3.09**
mzzv42z	1.05	1.02	0.99	1.09	1.02	*0.80*	1.02	0.91	1.04	1.34	0.87	1.05	1.01	1.08	1.01	**1.84**
neos1	1.02	1.03	0.98	1.07	1.01	1.06	1.04	1.02	1.00	1.01	1.01	1.01	1.02	1.00	1.01	**5.25**
neos2	1.04	1.02	0.98	1.06	0.92	*0.79*	1.07	*0.75*	1.02	0.86	*0.76*	1.06	1.02	0.99	1.01	**3.12**
neos3	1.04	1.01	0.99	>**1.51**	>**1.50**	>1.50	>**1.50**	>**1.50**	>**1.50**	>**1.50**	>**1.50**	>**1.50**	1.00	>**1.50**	1.00	>**1.50**
neos5	1.04	1.05	0.97	**1.24**	1.02	1.07	1.03	1.09	1.01	1.01	1.00	1.02	1.01	0.99	1.00	**2.40**
neos6	1.04	1.01	0.99	**1.73**	*0.48*	1.15	0.97	*0.43*	**1.29**	**1.90**	0.99	**1.68**	1.01	0.99	1.00	**5.54**
neos7	1.03	1.02	1.01	**1.22**	1.08	1.11	1.09	1.14	1.00	1.01	1.00	1.03	1.00	1.01	1.09	0.82
neos10	1.04	1.02	0.98	1.04	1.02	1.06	1.02	1.02	1.01	1.01	1.01	1.01	1.01	1.00	1.01	**2.29**
neos11	1.03	1.03	1.00	**1.59**	0.98	1.17	**1.27**	**1.48**	1.05	1.14	**1.20**	1.19	1.01	1.01	1.00	**3.16**
neos13	1.03	**3.63**	0.88	0.91	*0.79*	0.83	*0.65*	0.81	1.02	1.02	1.01	1.02	1.01	**1.51**	0.98	1.11
neos21	1.02	0.86	1.00	1.05	1.02	1.00	1.18	1.03	1.00	0.99	1.03	1.01	0.89	1.00	0.97	**1.61**

continue next page

Name	SimRou	Round	Shift	FeasP	CoefDi	PsCstDi	FracDi	VecDi	OPsDi	RooSoD	LineDi	GuiDi	Octane	CrOv	Rens	ALL
neos22	1.02	1.01	*0.58*	**1.22**	1.18	**1.48**	**1.29**	1.13	**1.25**	**1.21**	**1.30**	1.06	1.00	*0.73*	1.02	**2.95**
neos632659	1.03	0.99	0.96	**1.53**	**1.42**	**1.56**	1.18	**1.90**	**1.40**	**1.49**	**1.55**	0.85	1.00	1.01	**1.40**	**1.72**
nug08	1.04	1.05	1.00	1.06	1.03	1.09	1.03	1.03	1.04	1.04	1.03	1.04	1.04	1.00	0.96	**2.04**
nw04	1.03	1.00	0.99	1.09	1.01	1.05	1.01	1.01	1.00	1.00	1.00	1.00	0.86	1.00	0.99	**1.75**
p0201	1.04	1.02	0.98	1.07	1.06	1.09	1.01	1.03	0.99	0.99	0.99	1.00	0.98	1.00	0.85	**1.33**
p0282	1.00	1.03	1.03	1.01	1.02	1.02	1.01	1.02	0.99	0.99	1.01	1.01	1.00	1.01	**1.29**	**3.23**
p0548	1.02	1.01	0.99	1.02	1.03	1.02	1.01	0.99	1.00	0.99	1.00	0.98	1.02	0.98	1.01	**2.41**
p2756	1.03	1.25	0.98	*0.68*	1.00	1.04	1.01	1.01	1.00	1.00	1.00	1.01	0.87	1.01	1.01	**1.52**
pk1	1.06	1.00	**1.37**	1.08	*0.78*	1.18	1.13	*0.80*	**1.21**	0.93	0.98	*0.78*	1.00	0.99	0.85	**3.31**
pp08a	1.04	1.00	0.98	1.01	1.03	1.04	1.03	0.97	0.95	1.05	1.03	1.01	0.99	0.96	1.00	**1.38**
pp08aCUTS	1.04	1.04	1.00	0.81	0.89	0.93	1.04	0.98	*0.76*	1.03	1.02	1.03	1.04	0.94	0.83	**1.58**
prod1	1.04	1.00	1.00	**1.45**	1.01	1.07	1.00	1.03	1.00	1.02	0.99	0.96	1.00	0.96	0.98	**1.71**
qap10	1.05	1.03	1.00	0.98	1.02	1.07	1.07	1.09	1.02	1.02	1.01	1.03	1.01	1.02	0.92	**1.97**
qiu	1.02	1.01	1.00	1.15	0.92	0.99	1.00	1.00	1.03	0.98	1.00	0.97	0.99	1.00	0.99	**1.55**
qnet1	1.02	1.01	0.99	1.16	1.03	1.04	1.02	1.03	1.00	1.01	1.00	1.00	1.01	1.01	0.97	**1.80**
qnet1_o	1.05	1.06	0.96	1.14	1.03	1.05	1.02	1.03	1.00	1.00	1.00	1.00	1.00	1.00	1.08	**1.75**
ran8x32	1.05	0.99	0.99	**1.30**	*0.78*	*0.75*	0.86	1.05	0.98	0.89	0.86	0.92	1.00	0.93	1.02	**1.63**
ran10x26	**1.26**	1.01	0.99	**1.26**	0.97	**1.24**	1.06	**1.36**	1.15	1.13	0.99	**1.22**	1.01	1.16	1.12	**1.84**
ran12x21	1.08	1.02	0.99	0.97	1.03	0.89	0.99	1.03	0.92	0.92	0.94	0.91	1.00	0.97	0.93	**1.65**
ran13x13	0.96	1.00	0.99	1.10	**1.27**	1.08	0.86	0.83	0.96	0.85	**1.28**	**1.42**	1.00	1.06	1.00	**2.93**
rentacar	1.03	1.02	0.99	**1.84**	1.01	1.05	1.02	1.02	1.00	1.00	1.00	1.01	0.99	0.99	0.99	**3.61**
rgn	1.01	1.00	0.96	1.02	1.02	1.02	0.98	1.02	1.00	0.99	0.97	0.97	1.00	0.97	0.95	**1.32**
rout	1.04	1.01	0.99	0.98	1.01	1.02	0.99	1.02	1.06	1.02	0.97	1.03	1.00	1.08	0.99	**2.72**
set1ch	1.01	1.01	0.95	0.94	1.01	1.03	1.00	1.02	1.00	1.02	1.01	1.01	1.01	1.00	1.02	**1.48**
seymour1	1.02	1.00	0.94	1.03	0.91	0.96	0.96	0.95	0.89	0.93	0.97	0.96	0.99	1.02	1.00	**1.86**
stein27	0.94	0.89	0.93	0.96	1.00	0.94	0.96	1.13	0.90	0.90	0.91	0.89	*0.51*	1.01	0.89	*0.72*
stein45	1.01	0.91	0.94	1.03	1.05	1.01	0.95	1.09	0.91	0.95	0.97	0.95	*0.75*	1.03	0.94	**1.44**
swath1	1.02	1.02	0.99	1.05	1.05	1.07	1.12	1.03	1.00	1.01	1.01	1.02	1.01	0.99	0.86	**1.64**
swath2	1.03	1.02	0.92	1.03	0.99	1.06	0.93	0.95	1.01	1.10	1.09	1.04	1.01	**1.24**	0.93	**1.87**
swath3	1.05	1.01	1.00	1.01	*0.65*	**1.34**	1.17	0.92	**1.61**	1.19	**1.26**	**1.29**	**1.35**	*0.76*	**1.30**	1.10
vpm2	1.04	1.02	0.88	1.08	1.06	1.07	1.00	0.89	1.05	1.07	1.03	1.03	1.01	1.02	0.93	**1.86**
Geom. Mean	1.0436	1.0440	0.9735	1.1231	0.9945	1.0403	1.0639	1.0309	0.9983	1.0214	1.0157	1.0114	0.9787	1.0229	1.0223	2.0183

Table B.13. Factor by which disabling a certain heuristic influences the computation time

Name	SimRou	Round	Shift	FeasP	CoefDi	PsCstDi	FracDi	VecDi	OPsDi	RooSoD	LineDi	GuiDi	Octane	CrOv	Rens	ALL
10teams	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
30:70:4_5:0_5:100	1.00	1.00	1.00	1.00	**1.24**	**1.93**	**1.27**	1.15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.18
30:70:4_5:0_95:98	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
30:70:4_5:0_95:100	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
acc-0	1.00	1.00	1.00	**55.50**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**55.50**
acc-1	1.00	1.00	1.00	**47.50**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**47.50**
acc-2	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
acc-3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
acc-4	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
acc-5	1.00	1.00	1.00	1.15	*0.71*	*0.76*	1.18	1.18	*0.80*	**1.37**	**1.83**	1.00	1.00	1.00	1.00	>**1.24**
acc-6	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
aflow30a	1.00	1.00	1.00	1.19	**1.21**	1.13	0.91	1.11	0.83	0.82	0.93	1.05	1.00	**1.56**	**1.38**	**1.91**
air03	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
air04	1.00	1.00	1.00	1.00	0.92	1.21	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.10
air05	1.00	1.00	1.00	**1.48**	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.62**
bc1	1.00	1.00	1.00	1.01	0.98	1.05	1.07	0.99	0.99	0.99	0.99	1.02	1.00	1.07	**1.18**	1.17
bell3a	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.02
bell5	1.00	1.00	1.00	1.01	1.00	1.00	1.00	1.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.25**
bienst1	1.00	1.00	1.00	1.03	0.94	1.15	**1.61**	0.92	0.90	0.98	1.01	0.96	1.00	1.00	1.00	0.87
bienst2	1.00	1.00	1.00	1.06	0.96	0.88	1.10	**1.34**	1.06	1.09	1.15	0.97	1.00	1.00	1.00	1.03
blend2	1.00	1.00	1.00	1.00	**1.21**	**1.21**	**1.21**	1.19	**1.47**	**1.46**	0.98	0.99	1.00	**1.21**	**6.55**	**13.01**
cap6000	1.00	1.10	1.00	1.00	0.89	0.97	0.95	0.89	0.90	1.01	0.98	0.96	1.00	1.00	1.00	>**79.16**
dano3_3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.40**
dano3_4	1.00	1.00	1.00	1.00	*0.43*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.95
dano3_5	1.00	1.00	1.00	0.97	1.00	1.01	1.01	1.01	1.01	1.01	1.01	1.01	1.00	1.00	1.00	**1.25**
disctom	1.00	1.00	1.00	>**36639**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	>**18048**
dcmulti	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.01
dsbmip	1.00	1.00	1.00	**18.00**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**18.00**
egout	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
eilD76	1.00	1.00	1.00	1.00	0.94	0.93	0.93	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**3.41**	**1.99**
enigma	1.00	1.00	1.00	1.00	*0.09*	*0.23*	*0.29*	*0.32*	*0.19*	0.87	*0.54*	1.00	1.00	1.00	1.00	*0.24*
fast0507	>0.86	>0.88	>0.90	1.07	1.06	>0.91	>0.86	>0.79	>0.92	>0.87	>0.84	0.88	1.00	1.00	1.00	>**1.25**
fiber	1.00	1.00	1.00	1.00	0.99	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.52**	**1.59**

continue next page

Name	SimRou	Round	Shift	FeasP	CoefDi	PsCstDi	FracDi	VecDi	OPsDi	RooSoD	LineDi	GuiDi	Octane	CrOv	Rens	ALL
fixnet6	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**3.83**	**4.06**
flugpl	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.93	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.93
gesa2-o	1.00	1.00	1.00	1.00	0.93	**1.29**	1.11	0.95	*0.77*	*0.77*	1.03	1.02	1.00	1.03	1.00	**1.08**
gesa2	1.00	1.00	**1.56**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**4.73**	**10.16**
gesa3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**32.03**	**33.66**
gesa3_o	1.00	1.00	1.00	*0.78*	0.91	0.89	0.93	0.98	0.89	0.90	0.97	0.96	1.00	1.00	1.00	1.04
gt2	1.00	1.00	1.03	1.00	**2.25**	*0.44*	1.00	1.00	**1.25**	**1.25**	1.00	1.00	1.00	1.00	1.00	**4.05**
irp	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.05	0.97	0.97	1.00	1.00	1.00	1.00	**1.74**	**1.61**
khb05250	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.18	**2.91**
l152lav	1.00	1.06	1.00	*0.73*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.86
lseu	1.00	1.00	1.00	1.00	1.00	0.99	0.99	1.00	0.99	0.99	1.00	1.02	1.00	1.00	**1.37**	**1.52**
mas74	1.00	1.00	*0.75*	1.07	*0.75*	*0.80*	0.97	*0.76*	*0.64*	0.90	*0.67*	0.92	1.00	1.07	1.00	>0.91
mas76	1.00	1.00	1.03	**1.23**	0.93	0.96	1.00	0.93	0.94	1.02	0.94	0.97	1.00	**1.78**	1.06	**3.08**
mas284	1.00	1.00	1.01	1.08	1.06	1.10	1.08	1.00	0.96	1.02	1.04	1.06	1.00	1.00	1.01	**1.46**
misc03	1.00	1.00	1.00	0.94	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.91
misc06	1.00	1.00	1.00	1.00	0.96	0.96	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**3.28**
misc07	1.00	1.00	1.00	1.00	1.01	0.98	0.99	1.07	1.02	1.12	1.07	1.02	1.00	1.00	1.00	1.10
mitre	1.00	1.00	*0.50*	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	*0.79*
mod008	1.08	1.19	1.08	1.08	0.97	0.97	0.97	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**10.95**
mod011	1.00	1.00	1.00	1.18	0.89	0.94	1.11	0.94	0.96	0.96	0.94	1.01	1.00	1.00	1.00	**1.63**
modglob	1.04	1.00	1.00	1.00	0.92	1.00	0.94	0.85	0.83	*0.75*	0.98	0.98	1.00	1.12	0.82	1.19
mzzv11	1.00	1.00	1.00	1.00	**1.73**	**1.76**	**2.88**	**1.74**	**1.54**	**2.16**	**1.40**	**1.68**	1.00	**1.68**	1.00	>**3.14**
mzzv42z	1.00	1.00	1.00	1.00	**1.20**	0.82	0.82	0.87	1.14	**1.65**	*0.79*	1.12	1.00	1.20	1.00	**1.91**
neos1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**12.50**
neos2	1.00	1.00	1.00	0.97	*0.80*	*0.61*	0.88	0.63	1.00	0.81	*0.64*	1.08	1.00	1.00	1.00	**2.06**
neos3	1.00	1.00	1.00	>**1.43**	>**1.50**	>**1.51**	>**1.59**	>**1.43**	>**1.63**	>**1.49**	>**1.69**	>**1.58**	1.00	>**1.54**	1.00	>*0.61*
neos5	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
neos6	1.00	1.00	1.00	**1.66**	*0.32*	0.84	1.12	0.23	**1.22**	**2.73**	0.94	**2.48**	1.00	1.00	1.00	**6.00**
neos7	1.00	1.00	1.00	1.09	1.08	1.06	1.05	1.06	1.04	1.06	1.05	1.03	1.00	1.00	1.08	*0.52*
neos10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
neos11	1.00	1.00	1.00	**1.61**	0.98	**1.26**	**1.37**	**1.57**	1.09	**1.29**	1.18	**1.41**	1.00	1.00	1.00	**2.19**
neos13	1.00	**3.96**	1.00	*0.67*	*0.68*	*0.68*	*0.43*	*0.66*	1.00	1.00	1.00	1.00	1.00	**1.70**	1.00	*0.64*
neos21	1.00	0.86	1.00	1.00	1.06	0.93	1.20	0.95	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.87

continue next page

Name	SimRou	Round	Shift	FeasP	CoefDi	PsCstDi	FracDi	VecDi	OPsDi	RooSoD	LineDi	GuiDi	Octane	CrOv	Rens	ALL
neos22	1.00	1.00	*0.61*	1.17	1.13	**1.41**	**1.26**	1.11	**1.33**	**1.29**	**1.32**	1.08	1.00	0.90	1.00	**2.23**
neos632659	1.00	1.00	1.01	1.12	1.43	**1.58**	**1.23**	**1.91**	**1.49**	**1.59**	**1.59**	*0.80*	1.00	1.00	**1.43**	0.93
nug08	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
nw04	1.00	1.00	1.00	**2.50**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**2.50**
p0201	1.00	1.13	1.00	1.01	1.07	1.07	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.87
p0282	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**2.14**	**31.78**
p0548	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**1.36**	1.00	1.00	**4.79**
p2756	1.00	**2.00**	0.95	0.95	0.91	0.91	0.91	0.95	0.95	1.00	1.00	1.00	1.00	1.00	1.00	**3.90**
pk1	1.00	1.00	**1.29**	1.00	0.86	1.12	1.08	0.86	1.14	0.94	0.99	0.87	1.00	1.00	0.91	**2.00**
pp08a	1.00	1.00	1.00	1.00	1.00	1.02	1.00	0.99	1.01	1.05	1.06	0.98	1.00	1.02	1.00	1.18
pp08aCUTS	0.98	1.05	1.00	0.82	0.87	0.88	1.03	0.97	*0.66*	1.06	1.01	1.03	1.05	0.99	1.00	**1.94**
prod1	1.00	1.00	1.07	1.07	0.99	1.01	0.99	1.00	1.03	1.02	1.02	1.00	1.00	1.01	1.00	1.04
qap10	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
qiu	1.00	1.00	1.00	1.08	0.97	1.00	1.01	0.99	1.02	1.00	1.00	0.99	1.00	1.00	1.00	1.07
qnet1	1.00	1.12	1.00	1.00	1.01	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.98	**1.32**
qnet1_o	1.00	**1.31**	1.11	1.00	1.04	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**2.04**	**1.73**
ran8x32	1.00	1.00	1.00	0.99	*0.64*	*0.60*	*0.78*	0.95	1.01	0.85	*0.76*	0.90	1.00	0.94	0.96	1.07
ran10x26	**1.21**	1.00	1.00	1.07	0.96	**1.23**	1.04	1.14	**1.21**	1.12	1.00	**1.20**	1.00	**1.24**	1.15	**1.31**
ran12x21	0.99	1.00	1.00	0.89	0.99	0.84	0.85	0.94	0.96	0.89	0.91	0.88	1.00	0.97	0.90	1.04
ran13x13	0.93	1.00	1.00	1.04	**1.26**	1.00	0.87	0.84	1.00	0.84	**1.33**	**1.55**	1.00	1.07	0.99	**2.07**
rentacar	1.00	1.00	1.00	**7.80**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	**6.20**
rgn	1.00	1.00	1.00	0.84	0.99	0.99	1.00	1.02	0.99	0.99	0.99	1.00	1.00	1.00	1.00	1.01
rout	1.00	1.00	1.00	*0.78*	0.87	0.91	0.82	0.87	1.02	0.95	0.87	0.93	1.00	1.02	1.00	**1.80**
set1ch	1.00	1.00	1.09	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.09	1.09
seymour1	0.93	0.93	0.93	0.93	0.95	0.95	0.96	0.95	0.95	0.98	0.99	1.01	1.00	1.06	1.00	**1.54**
stein27	1.00	1.01	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.01
stein45	1.00	0.97	1.00	1.00	1.00	0.99	0.94	1.00	0.95	0.96	1.02	0.97	1.00	1.00	1.00	1.17
swath1	1.00	1.00	1.00	0.90	1.12	1.01	1.00	1.02	1.02	1.02	1.00	0.98	1.00	1.00	1.00	**1.20**
swath2	1.00	1.00	0.91	0.99	0.93	0.98	*0.74*	0.88	1.03	0.91	1.11	1.05	1.00	1.48	1.00	**1.30**
swath3	1.00	1.00	1.00	1.05	*0.59*	1.04	0.88	*0.66*	**1.25**	1.00	0.92	0.97	1.00	0.72	1.00	0.90
vpm2	1.00	1.00	0.86	1.03	1.00	1.05	0.99	*0.79*	1.10	1.11	1.05	1.07	1.00	1.08	1.01	**1.33**
Geom. Mean	1.0001	1.0470	0.9912	1.3035	0.9550	0.9839	0.9991	0.9696	0.9933	1.0311	1.0041	1.0241	1.0037	1.0365	1.1453	1.9912

Table B.14. Factor by which disabling a certain heuristic influences the solving nodes

Name	SimRou	Round	Shift	FeasP	CoefDi	PsCstDi	FracDi	VecDi	OPsDi	RooSoD	LineDi	GuiDi	Octane	CrOv	Rens	ALL	Diving
a1c1s1	1.00	1.00	1.00	1.00	0.90	1.02	1.04	1.07	1.28	0.95	0.97	1.00	1.00	1.02	1.00	–	2.71
aflow40b	1.00	1.00	1.00	0.78	0.38	0.78	1.23	0.57	0.52	0.97	0.77	0.87	1.00	1.09	1.32	1.88	1.18
arki001	1.00	1.00	1.00	0.96	1.00	0.98	0.98	1.05	1.00	1.00	0.99	1.01	1.00	1.00	1.00	1.01	1.01
atlanta-ip	–	–	–	–	–	–	–	∞	∞	–	–	–	–	–	–	–	–
binkar10_1	1.00	1.00	1.00	1.06	1.00	1.04	1.07	0.99	1.05	1.00	0.97	0.99	1.00	1.06	1.00	1.15	1.46
dano3mip	1.00	1.00	1.00	1.00	1.03	0.94	0.97	0.88	0.96	0.97	1.00	1.00	1.00	1.00	0.93	–	1.73
danoint	1.00	1.00	1.00	0.68	1.16	0.71	1.07	1.09	0.74	1.00	0.96	1.04	1.00	1.00	1.00	0.72	0.89
ds	1.00	1.00	1.00	1.00	0.87	0.88	0.97	0.82	0.70	1.03	1.09	0.90	1.00	1.00	1.00	–	2.42
glass4	1.00	1.00	1.01	0.77	0.89	0.88	0.91	0.91	1.01	0.91	0.98	0.89	1.00	1.01	1.01	1.32	1.40
liu	1.00	1.00	1.00	0.57	0.85	0.85	0.77	1.40	0.85	0.78	0.63	1.16	1.00	1.00	0.73	–	1.43
markshare1	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
markshare2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
mkc	1.00	1.00	0.89	0.78	0.90	0.75	0.78	1.05	0.92	0.85	0.80	0.93	1.00	1.02	0.73	1.60	1.22
mkc1	1.00	1.00	1.00	1.00	0.99	0.97	1.02	1.05	0.97	0.94	1.02	1.01	1.00	1.02	1.00	1.02	1.04
momentum1	–	–	–	–	–	–	–	∞	–	–	–	–	–	–	–	–	–
momentum2	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
msc98-ip	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–	–
neos616206	1.05	1.06	1.03	0.86	0.87	0.29	1.05	0.90	1.07	0.90	0.73	0.98	1.08	0.99	1.05	1.46	0.69
net12	1.02	1.02	1.00	–	0.48	0.79	0.77	0.82	0.72	0.44	0.89	0.64	1.01	1.00	1.02	–	–
noswot	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	0.18
nsrand-ipx	1.00	1.00	1.08	1.00	1.38	1.38	1.08	1.15	1.39	1.04	1.04	1.08	1.00	1.27	1.23	5.80	1.72
opt1217	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	1.00	1.00	1.00	0.99	1.00	1.00	1.00	–	0.98
protfold	–	–	–	–	–	–	–	∞	–	–	–	–	–	–	–	–	–
rd-rplusc-21	–	–	–	–	∞	∞	–	–	–	–	–	–	–	–	–	–	–
roll3000	1.01	1.00	1.00	1.00	1.07	0.86	1.07	1.11	1.06	0.98	1.04	0.97	1.01	1.09	1.18	4.32	2.96
seymour	1.01	1.00	1.00	0.95	1.00	1.00	1.01	1.01	0.94	0.93	0.99	1.00	1.00	0.94	0.94	–	1.63
sp97ar	1.00	1.00	1.00	0.86	0.96	1.08	1.04	1.16	1.08	1.02	0.89	1.17	1.00	1.00	1.06	1.74	2.06
swath	1.00	1.00	1.00	1.00	0.85	0.80	0.74	0.85	0.76	0.86	0.99	0.88	1.00	1.01	1.00	1.02	0.98
t1717	1.00	1.00	1.00	1.00	1.21	1.23	–	0.82	1.64	1.07	6.20	1.03	1.00	1.00	1.00	–	–
timtab1	1.01	1.00	1.00	0.92	0.81	1.23	0.91	0.93	1.27	1.12	0.88	1.19	1.01	1.00	1.00	1.20	1.14
timtab2	1.01	1.00	1.00	1.12	1.10	1.16	0.94	1.15	1.06	1.16	1.33	1.01	1.00	1.00	1.00	–	–
tr12-30	1.00	1.00	1.02	0.96	1.04	0.96	0.99	1.00	0.95	1.00	1.04	0.99	1.00	1.00	0.99	1.19	1.18
Geom. Mean	1.0040	1.0034	1.0012	0.9112	0.9357	0.9025	0.9779	0.9930	0.9602	0.9700	0.9497	0.9979	1.0042	1.0220	1.0004	1.4666	1.2558

Table B.15. Factor by which disabling a certain heuristic influences the primal-dual gap γ_{PD}

List of Algorithms

List of Figures

List of Tables

Bibliography

[1] T. Achterberg. SCIP - a framework to integrate constraint and mixed integer programming. Technical Report 04-19, Zuse Institute Berlin, 2004. `http://www.zib.de/Publications/abstracts/ZR-04-19/`.

[2] T. Achterberg. *Constraint Integer Programming*. PhD thesis, Technische Universität Berlin, 2007. To appear.

[3] T. Achterberg and T. Berthold. Improving the Feasibility Pump. Technical Report 05-42, ZIB, 2005. To appear in Operations Research.

[4] T. Achterberg, T. Berthold, T. Koch, A. Martin, and K. Wolter. SCIP (Solving Constraint Integer Programs), documentation. `http://scip.zib.de/doc`.

[5] T. Achterberg, T. Koch, and A. Martin. Branching rules revisited. *Operations Research Letters*, 33:42–54, 2005.

[6] T. Achterberg, T. Koch, and A. Martin. MIPLIB 2003. *Operations Research Letters*, 34(4):1–12, 2006. `http://miplib.zib.de`.

[7] K. R. Apt. *Principles of Constraint Programming*. Cambridge University Press, 2003.

[8] T. Bäck. *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK, 1996.

[9] E. Balas. Intersection cuts – a new type of cutting planes for integer programming. *Operations Research*, 19:19–39, 1971.

[10] E. Balas, S. Ceria, M. Dawande, F. Margot, and G. Pataki. OCTANE: A New Heuristic for Pure 0-1 Programs. *Operations Research*, 49, 2001.

[11] E. Balas and C. H. Martin. Pivot-and-Complement: A Heuristic for 0-1 Programming. *Management Science*, 26(1):86–96, 1980.

[12] E. Balas and C. H. Martin. Pivot-and-Shift: A Heuristic for Mixed Integer Programming. GSIA, Carnegie Mellon University, August 1986.

[13] E. Balas, S. Schmieta, and C. Wallace. Pivot and shift - a mixed integer programming heuristic. *Discrete Optimization*, 1(1):3–12, June 2004.

[14] L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. Technical Report OR-05-5, University of Padova, Italy, May 2005.

[15] R. E. Bixby. MIP Heuristics. ILOG Powerpoint presentation, 2005. `http://co-at-work.zib.de/download/CD/Talks/COatW-2005-10-08-Bixby-MIP3.pdf`.

[16] R. E. Bixby, S. Ceria, C. M. McZeal, and M. W. Savelsbergh. An updated mixed integer programming library: MIPLIB 3.0. *Optima*, (58):12–15, June 1998.

[17] R. E. Bixby, M. Fenelon, Z. Gu, E. Rothberg, and R. Wunderling. MIP: Theory and practice – closing the gap. In M. Powell and S. Scholtes, editors, *Systems Modelling and Optimization: Methods, Theory, and Applications*, pages 19–49. Kluwer Academic Publisher, 2000.

[18] R. Borndörfer. *Aspects of Set Packing, Partitioning, and Covering.* Shaker Verlag, Aachen, 1998. Ph.D. thesis, Technische Universität Berlin.

[19] N. Christofides. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. GSIA report 388, Carnegie-Mellon University, 1976.

[20] V. Chvátal. *Linear programming.* Freeman, 1983.

[21] E. Danna, E. Rothberg, and C. L. Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming, Ser. A*, 2004.

[22] Dash Optimization. XPRESS-MP. `http://www.dashoptimization.com/home/products/products_optimizer.html`.

[23] A. Eisenblätter, H.-F. Geerdes, T. Koch, A. Martin, and R. Wessäly. UMTS radio network evaluation and optimization beyond snapshots. *Math. Meth. Oper. Res.*, 63(1), 2006.

[24] M. Fischetti, F. Glover, and A. Lodi. The feasibility pump. *Mathematical Programming*, 104(1):91–104, 2005.

[25] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming 98*, 2003.

[26] M. Fischetti and A. Lodi. Repairing MIP infeasibility through Local Branching. Technical report, University of Padova, September 2005.

[27] J. Forrest and R. Lougee-Heimer. COIN branch and cut, user guide, 2005. `http://www.coin-or.org/Cbc`.

[28] F. Glover. Tabu Search: A Tutorial. *Interfaces*, 20(4):74–94, 1990.

[29] F. Glover and M. Laguna. General Purpose Heuristics for Integer Programming - Part I. *Journal of Heuristics 3*, 1997.

[30] F. Glover and M. Laguna. General Purpose Heuristics for Integer Programming - Part II. *Journal of Heuristics 3*, 1997.

[31] F. Glover and M. Laguna. *Tabu Search*. Kluwer Academic Publisher, Boston, Dordrecht, London, 1997.

[32] F. Glover, A. Løkketangen, and D. L. Woodruff. Scatter Search to Generate Diverse MIP Solutions. *OR Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research*, 2000.

[33] GNU. the GNU linear programming kit. `http://www.gnu.org/software/glpk`.

[34] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of the American Society*, 64:275–278, 1958.

[35] M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, second corrected edition, 1993.

[36] M. Grötschel and M. W. Padberg. Ulysses 2000: In search of optimal solutions to hard combinatorial problems. SC 93-34, Konrad-Zuse-Zentrum für Informationstechnik Berlin, Nov. 1993.

[37] G. Gutin and A. Punnen. *Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, 2002.

[38] S. Heipcke. *Applications of Optimization with Xpress-MP*. Dash Optimization, Blisworth, U.K., 2002.

[39] F. S. Hillier. Efficient heuristic procedures for integer linear programming with an interior. *Operations Research*, 17:600–637, 1969.

[40] T. Ibaraki, T. Ohashi, and H. Mine. A heuristic algorithm for mixed-integer programming problems. *Mathematical Programming Study*, 2:115–136, 1974.

[41] ILOG CPLEX 10.01. Reference Manual, 2006. `http://www.ilog.com/products/cplex`.

[42] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by Simulated Annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[43] S. Lin and B. W. Kernighan. An effective heuristic algorithm for the travelling-salesman problem. *Operations Research*, 21:498–516, 1973.

[44] A. Lodi. Local branching: a tutorial. presented at MIC2003, Kyoto, 2003. `http://www.or.deis.unibo.it/research_pages/ORinstances/mic2003-lb.pdf`.

[45] A. Løkketangen. Heuristics for 0-1 mixed integer programming. *Handbook of Applied Optimization*, 2002.

[46] A. Martin. Integer programs with block structure. Habilitations-Schrift, Technische Universität Berlin, 1998. `http://www.zib.de/Publications/abstracts/SC-99-03/`.

[47] M. Mitchell. *An Introduction to Genetic Algorithms*. The MIT Press, Cambridge, USA, 1999.

[48] H. Mittelmann. Decision tree for optimization software: Benchmarks for optimization software, 2003. `http://plato.asu.edu/bench.html`.

[49] G. L. Nemhauser and M. W. Savelsbergh. MINTO 3.1, 2004. `http://coral.ie.lehigh.edu/~minto`.

[50] T. K. Ralphs. SYMPHONY version 5.0 user's manual, 2004. `http://branchandcut.org/SYMPHONY/man`.

[51] Y. Rochat and E. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1:147–167, 1995.

[52] E. Rothberg. An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions. Technical report, ILOG Inc., 2005.

[53] R. M. Saltzman and F. S. Hillier. A heuristic ceiling point algorithm for general integer linear programming. *Management Science*, 38(2):263–283, February 1992.

[54] A. Schrijver. *Combinatorial optimization : polyhedra and efficiency*. Springer, 2003.

[55] E. Tsang and C. Voudouris. Guided Local Search and its Application to the Travelling Salesman Problem. *European Journal of Operational Research*, 113(2):469–499, 1999.

[56] P. J. M. van Laarhoven and E. H. L. Aarts, editors. *Simulated annealing: theory and applications*. Kluwer Academic Publishers, Norwell, MA, USA, 1987.

[57] J. P. Walser. *Integer Optimization by Local Search*, volume 1637 of *Lecture Notes in Computer Science*. Springer, Berlin et al., 1999.

[58] K. Wolter. Implementation of Cutting Plane Separators for Mixed Integer Programs. Master's thesis, Technische Universität Berlin, 2006. To appear.

Die selbständige und eigenhändige Anfertigung dieser Arbeit versichere ich an Eides statt.

Berlin, den 11.09.2006