RENS– The Relaxation Enforced Neighborhood Search

Timo Berthold*

July 20, 2009

Abstract

In the recent years, a couple of quite successful large neighborhood search improvement heuristics for MIPs has been published. We present a new start heuristic called RENS for general MIPs working in the spirit of large neighborhood search. It constructs a sub-MIP which represents the space of all feasible roundings of some fractional point – normally the optimum of the LP-relaxation of the original MIP. Thereby, one is able to determine whether a point can be rounded to a feasible solution and which is the best possible rounding. This can be used for the analysis of MIP rounding heuristics. Furthermore, a slightly modified version of RENS proves to be a well-performing heuristic inside the branch-cut-and-price framework SCIP.

Keywords: mixed integer programming, primal heuristics, large neighborhood search

1 Introduction

Solving Mixed Integer Programs (MIPs) is one of the most important techniques to cope with issues that arise in various areas of Combinatorial Optimization and Operations Research. Although MIP-solving is an $\mathcal{NP}$-hard optimization problem, many practically relevant instances can be solved in reasonable time. A MIP is defined by a set of variables, a set of linear constraints, a set of linear constraints and a linear objective function which has to be optimized.

More formally stated:

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

$$\begin{array}{lll} \min & c^T x & \\ \textit{such that} & Ax \leq b & \\ & l \leq x \leq u & \\ & x_j \in \mathbb{Z} & \textit{for all } j \in I \end{array} \tag{1}$$

*Konrad-Zuse-Zentrum für Informationstechnik Berlin, berthold@zib.de. Supported by the DFG Research Center MATHEON *Mathematics for key technologies.*

is called a mixed integer program (MIP). *The optimization problem which arises if the integrality constraints are left out, is called the LP (linear program) relaxation of the MIP.*

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.

The standard approach to solve MIPs are LP-based branch-and-cut algorithms. These are implicit enumeration strategies, for which the solution space is recursively divided into smaller subproblems, building up a so-called branch-and-bound tree. At each node the LP-relaxation of the current subproblem is solved and usually a branching is performed by adding complementary bound changes on an integer variable with a fractional value in the current LP solution.

In state-of-the-art MIP-solvers like Cplex [25] or SCIP [3], primal heuristics play a major role in finding feasible solutions in early steps of the branch-and-bound process (often already at the root node). The knowledge of feasible solutions guides the remaining search process, thereby reduces the overall computational effort, and nevertheless proves the feasibility of the MIP model. Furthermore, a good heuristic solution may be sufficient for practical applications.

Various methods for heuristic MIP-solving have been presented in the literature, including Hillier [24], Balas and Martin [8], Saltzman and Hillier [31], Glover and Laguna [19, 20, 21], Glover et al. [28, 22], Balas et al. [7, 9], Løkketangen [27], Fischetti et al. [17, 16], Danna et al. [14] Bertacco et al. [10], Achterberg and Berthold [2], Eckstein and Nediak [15], Ghosh [18] and Berthold [12].

Broadly speaking, heuristic rounding procedures and Large Neighborhood Search (LNS) strategies are complementary with respect to the computational effort. The running time of rounding heuristics often is linear in the number of fractional variables, whereas LNS heuristics usually have to solve NP-hard subproblems.

In this paper, we will present a new LNS heuristic called Rens, which in contrast to the ones presented in the literature of the recent years does not need an incumbent MIP solution as a start point . On the one hand, it can be used as a tool to evaluate the performance of heuristic rounding methods, on the other hand it proves to be a reasonable start heuristic on its own.

The rest of this article is organized as follows: In the remainder of Section 1 we give a brief introduction into the idea of Large Neighborhood Search and review LNS strategies which have been recently proposed. In Section 2 we introduce our new heuristic and discuss implementational issues. Finally, Section 3 presents some computational results.

Large Neighborhood Search

The *Local Search* [21, 33] approach generalizes the idea of k-OPT [26]: one defines a neighborhood of some reference point, determines a point of this neighborhood which is optimal for some function (e.g., the objective function of the MIP or some 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. During the recent years, another approach got into the focus of attention.

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. Several LNS heuristics [17, 14, 30, 18] have been presented in the literature of the last five years. They all have in common that they use the incumbent solution as starting points and define the neighborhood to be a sub-MIP of the original MIP which is constructed by fixing variables and adding constraints.

The main difference is the way in which the sub-MIP is defined. Two of the proposed methods form a sub-MIP by fixing some of the integer variables, one adds further constraints, and one is a mixture of these approaches.

Rins (Relaxation Induced Neighborhood Search) which was described by Danna, Rothberg, and Le Pape [14] fixes variables which take identic variables in the incumbent and the optimum of the LP-relaxation in the current branch-and-bound node. Crossover which was independently developed by Rothberg [30] and Berthold [11] fixes variables which take identic values in a couple of feasible solutions. Local Branching which was introduced by Fischetti and Lodi [17] adds an additional distance constraint which guarantees that the solutions of the sub-MIP does not differ in more than say k variables from the reference point. DINS as suggested by Ghosh [18] is a mixture of all these strategies.

All of them are improvement heuristics, hence they rely on information of at least one feasible solution. We propose a LNS heuristic, which only uses the optimum of the LP-relaxation of a MIP and can therefore be applied as a start heuristic.

2 Rens

In this section, we will describe an LNS heuristic which investigates the set of all possible roundings of a (fractional) solution of the LP-relaxation.

In many practical applications, a couple of integer variables already takes an integral value in the optimum of the LP-relaxation. The idea is to fix these variables and perform a LNS on the remaining variables. For many MIP-solving techniques binary variables are preferred to general integers. A bound change automatically fixes a binary variable which is useful for branching, and specific algorithm like probing [32], knapsack cover cuts [6, 23, 34], or Octane [7] are only used for binary variables. Therefore, we do not only fix variables, but rebound all general integer variables with a fractional LP-value to the nearest integers. Summarized, the sub-MIP is created by changing the bounds of all integer variables x_j to $l_j = \lfloor \bar{x}_j \rfloor$ and $u_j = \lceil \bar{x}_j \rceil$, where $\bar{x}$ is the optimum of the LP-relaxation.

As the overall performance of the heuristic strongly depends on $\bar{x}$ we named it *relaxation enforced neighborhood search*, or shortly Rens.

Rens is of special interest for the analysis of rounding heuristics. If the sub-MIP created by Rens is proven to be infeasible, no 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.

Implementation Details

For the practical use as a start heuristic integrated into a branch-and-bound framework like SCIP, one should only call Rens, if the resulting sub-MIP seems to be substantially easier than the original one. This means that at least a specific ratio of all integer variables, say r_1, or a specific ratio of all variables including the continuous ones, say r_2, should be fixed.

The first criterion keeps control of the difficulty of the sub-MIP 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 10000 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-MIP would be nearly as expensive as solving the ones of the original MIP.

Another way to avoid spending too much time in solving sub-MIPs is to add some limit to the solving process of the sub-MIP. 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-MIP. The solving node limit l_1 is a hard limit on the maximum number of branch-and-bound nodes the MIP-solver should at most process. The stalling node limit l_2 indicates how many nodes the MIP-solver should at most process without an improvement in the incumbent solution of the sub-MIP.

The solving node limit keeps control of the overall running time of the heuristic. On the other hand one does not want to abort the sub-MIP solving too early if the objective value of the incumbent solution keeps increasing during the search process and hence use a stalling node limit. Therefore, we decided to use both node limitation strategies simultaneously with a relatively small stalling node limit l_2 and a large solving node limit l_1.

3 Computational Results

We integrated an implementation of Rens into the branch-and-cut framework SCIP [1, 4, 3]. All computations were made on a 2.20 GHz AMD Opteron with 1024 KB cache and 32 GB RAM.

3.1 Test Set and Settings

We use a wide test set which consists of very different classes of MIP instances. Altogether, there are 129 instances which were taken from:

- the Miplib 3.0 [13],
- the Miplib 2003 [5], and
- the MIP collection of Mittelmann [29].

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.

Our first test evaluates the roundability of LP optima of our test instances. We applied RENS on the optimum of the LP-relaxation after the processing of the root node, hence using MIP preprocessing and cutting planes.

We used a time limit of one hour for the overall computation; solving the root node of the original MIP and the RENS sub-MIP.

In order to evaluate the roundability and therefrom the potential power of RENS as a heuristic, we aimed to completely solve the RENS sub-MIP in all cases. Therefore, we set $l_1 = l_2 = +\infty$ and $r_1 = r_2 = 0$.

Next, we examined the integration of RENS into an LP-based branch-and-cut framework. Therefore, we performed a second test, where we set limits on the number of nodes $l_1 = 10000$ and $l_2 = 500$, $r_1 = 0.5$, and $r_2 = 0.25$. We applied SCIP without RENS, SCIP with RENS used in the root node only, and SCIP with RENS used at every tenth depth of the branch-and-bound tree.

3.2 Results

Table 2 shows the results of applying RENS without regarding the number of fixed variables and node limits to the mentioned 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. 16 times the RENS sub-MIP 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.

This 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 could not be decided, 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.

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-MIP. Therefore, we suggest to set some limit on the number of total nodes and stalling nodes for the RENS sub-MIP, 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_1 and r_2 to a sufficiently large value.

Next, we examined the integration of RENS into SCIP. Therefore, we applied SCIP without RENS, SCIP with RENS used in the root node only, and SCIP with RENS used at every tenth depth of the branch-and-bound tree.

The test set is now separated into two groups:

- the *easy test set*, all 97 instances which could be solved to optimality by SCIP with default settings within a time limit of one hour, and

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
Best Primal Bound (hard)	3	**7**	4

Table 1. Summarized results of different integrations of RENS into SCIP, easy instances

- the *hard test set*, all 32 instances which could not be solved to optimality by SCIP with default settings within a time limit of one hour.

The results are shown in Tables 3 and 4.

To keep the following paragraphs short, we will 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. There are 10 instances for which the primal bound differs at the moment we reach the time limit. Among these, $\text{SCIP}_{\text{RRoot}}$ has found the best incumbent 7 times, $\text{SCIP}_{\text{RFreq10}}$ 4 times and SCIP_{NoR} 3 times.

The results of calling RENS at the root node with a stalling node limit of 500, an absolute node limit of 10000, $r_1 = 0.5$, and $r_2 = 0.25$ showed that 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 turns out to be a reasonable root node heuristic.

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 2. 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 3. 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 4. Integration of Rens into SCIP, hard instances

References

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

[2] T. Achterberg and T. Berthold. Improving the Feasibility Pump. *Discrete Optimization*, Special Issue 4(1):77–86, 2007.

[3] T. Achterberg, T. Berthold, M. Pfetsch, and K. Wolter. SCIP – Solving Constraint Integer Programs, documentation. `http://scip.zib.de`.

[4] T. Achterberg, T. Berthold, ThorstenKoch, and K. Wolter. Constraint integer programming: A new approach tointegrate CP and MIP. In L. Perron and M. A. Trick, editors, *Integration of AI and OR Techniques in ConstraintProgramming for Combinatorial Optimization Problems, 5th International Conference, CPAIOR 2008*, volume 5015 of *Lecture Notes in Computer Science*, pages 6–20. Springer, May 2008.

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

[6] E. Balas. Facets of the knapsack polytope. *Mathematical Programming*, 8:146–164, 1975.

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

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

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

[10] L. Bertacco, M. Fischetti, and A. Lodi. A feasibility pump heuristic for general mixed-integer problems. *Discrete Optimization*, Special Issue 4(1):77–86, 2007.

[11] T. Berthold. Primal Heuristics for Mixed Integer Programs. Master's thesis, Technische Universität Berlin, 2006.

[12] T. Berthold. Heuristics of the branch-cut-and-price-framework SCIP. In J. Kalcsics and S. Nickel, editors, *Operations Research Proceedings 2007*, pages 31–36. Springer-Verlag, 2008.

[13] 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, 1998.

[14] E. Danna, E. Rothberg, and C. L. Pape. Exploring relaxation induced neighborhoods to improve MIP solutions. *Mathematical Programming A*, 102(1):71–90, 2004.

[15] J. Eckstein and M. Nediak. Pivot, Cut, and Dive: a heuristic for 0-1 mixed integer programming. *Journal of Heuristics*, 13(5):471–503, 2007.

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

[17] M. Fischetti and A. Lodi. Local branching. *Mathematical Programming B*, 98(1-3):23–47, 2003.

[18] S. Ghosh. Dins, a mip improvement heuristic. In *IPCO*, pages 310–323, 2007.

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

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

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

[22] 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.

[23] P. L. Hammer, E. L. Johnson, and U. N. Peled. Facets of regular 0-1 polytopes. *Mathematical Programming*, 8:179–206, 1975.

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

[25] ILOG CPLEX 11.01. Reference Manual. `http://www.ilog.com/products/cplex`.

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

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

[28] A. Løkketangen and F. Glover. Solving zero/one mixed integer programming problems using tabu search. *European Journal of Operations Research*, 106:624–658, 1998.

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

[30] E. Rothberg. An Evolutionary Algorithm for Polishing Mixed Integer Programming Solutions. Technical report, ILOG Inc., 2005. to appear in INFORMS Journal on Computing.

[31] 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.

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

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

[34] L. A. Wolsey. Faces for a linear inequality in 0-1 variables. *Mathematical Programming*, 8:165–178, 1975.