Math. Prog. Comp. (2014) 6:33–54
DOI 10.1007/s12532-013-0060-9

FULL LENGTH PAPER

RENS

The optimal rounding

Timo Berthold

Received: 25 April 2012 / Accepted: 2 October 2013 / Published online: 1 November 2013
© Springer-Verlag Berlin Heidelberg and Mathematical Optimization Society 2013

Abstract This article introduces RENS, the *relaxation enforced neighborhood search*, a large neighborhood search algorithm for mixed integer nonlinear programs (MINLPs). It uses a sub-MINLP to explore the set of feasible roundings of an optimal solution $\bar{x}$ of a linear or nonlinear relaxation. The sub-MINLP is constructed by fixing integer variables x_j with $\bar{x}_j \in \mathbb{Z}$ and bounding the remaining integer variables to $x_j \in \{\lfloor \bar{x}_j \rfloor, \lceil \bar{x}_j \rceil\}$. We describe two different applications of RENS: as a standalone algorithm to compute an optimal rounding of the given starting solution and as a primal heuristic inside a complete MINLP solver. We use the former to compare different kinds of relaxations and the impact of cutting planes on the so-called *roundability* of the corresponding optimal solutions. We further utilize RENS to analyze the performance of three rounding heuristics implemented in the branch-cut-and-price framework SCIP. Finally, we study the impact of RENS when it is applied as a primal heuristic inside SCIP. All experiments were performed on three publicly available test sets of mixed integer linear programs (MIPs), mixed integer quadratically constrained programs (MIQCPs), and MINLPs, using solely software which is available in source code. It turns out that for these problem classes 60 to 70 % of the instances have *roundable* relaxation optima and that the success rate of RENS does not depend on the percentage of fractional variables. Last but not least, RENS applied as primal heuristic complements nicely with existing primal heuristics in SCIP.

Keywords Mixed integer programming · Mixed integer nonlinear programming · Primal heuristic · Large neighborhood search · Rounding

Electronic supplementary material The online version of this article (doi:10.1007/s12532-013-0060-9) contains supplementary material, which is available to authorized users.

T. Berthold (✉)
Department of Optimization, Zuse Institute Berlin,
Takustr. 7, 14195 Berlin, Germany
e-mail: berthold@zib.de

Mathematics Subject Classification (2000) 90C11 · 90C20 · 90C30 · 90C59

1 Introduction

Primal heuristics are algorithms that try to find feasible solutions of good quality for a given optimization problem within a reasonably short amount of time. There is typically no guarantee that they will find any solution, let alone an optimal one.

For mixed integer linear programs (MIPs) it is well known that general-purpose primal heuristics like the Feasibility Pump [3,34,35] are able to find high-quality solutions for a wide range of problems. Over time, primal heuristics have become a substantial ingredient of state-of-the-art MIP solvers [15,20]. Discovering good feasible solutions at an early stage of the MIP solving process has several advantages:

- The bounding step of the branch-and-bound [45] algorithm depends on the quality of the incumbent solution; a better primal bound leads to more nodes being pruned and hence to smaller search trees.
- The same holds for certain presolving and domain propagation strategies such as reduced cost fixing. Better solutions can lead to tighter domain reductions, in particular more variable fixings. This, as a consequence, might lead to better dual bounds and the generation of stronger cutting planes.
- In practice, it is often sufficient to compute a heuristic solution whose objective value is within a certain quality threshold. For hard MIPs that cannot be solved to optimality within a reasonable amount of time, it might still be possible to generate good primal solutions quickly.
- Improvement heuristics such as RINS [31] or Local Branching [33] need a feasible solution as starting point.

Similar statements hold for other classes of mathematical programs. Often, techniques such as reduced cost fixing or cutting planes are more heavily or even exclusively applied at the root node of a branch-and-bound search tree. Therefore, already knowing good solutions during root node processing is significantly more beneficial than finding them later during tree search.

The last fifteen years have seen several publications on general-purpose heuristics for MIPs, including [3,6,7,9,16,10,35–37,41,48,49,53,61]. For an overview, see [15, 38,39]. For mixed integer nonlinear programming, the last three years have shown a rising interest in the research community for general-purpose primal heuristics [11–13,23,21,28,30,46,52,51].

A *mixed integer nonlinear program (MINLP)* is an optimization problem of the form

$$\begin{aligned} \min \quad & d^T x \\ \text{s.t.} \quad & g_i(x) \leq 0 && \text{for all } i \in \mathcal{M} \\ & L_j \leq x_j \leq U_j && \text{for all } j \in \mathcal{N} \\ & x_j \in \mathbb{Z} && \text{for all } j \in \mathcal{I}, \end{aligned} \tag{1}$$

where $\mathcal{I} \subseteq \mathcal{N} := \{1, \dots, n\}$ is the index set of the integer variables, $d \in \mathbb{R}^n$, $g_i : \mathbb{R}^n \to \mathbb{R}$ for $i \in \mathcal{M} := \{1, \dots, m\}$, and $L \in (\mathbb{R} \cup \{-\infty\})^n$, $U \in (\mathbb{R} \cup \{+\infty\})^n$ are lower and upper bounds on the variables, respectively. Note that a nonlinear objective function can always be reformulated by introducing one additional variable and constraint, hence form (1) is general. We assume without loss of generality that $L_j \leq U_j$ for all $j \in \mathcal{N}$ and $L_j, U_j \in \mathbb{Z}$ for all $j \in \mathcal{I}$.

There are many subclasses of MINLP, the following four will be considered in this article:

- If all constraint functions g_i are quadratic, problem (1) is called a *mixed integer quadratically constrained program (MIQCP)*.
- If all constraint functions g_i are linear, problem (1) is called a *mixed integer program (MIP)*.
- If $\mathcal{I} = \emptyset$, problem (1) is called a *nonlinear program (NLP)*.
- If $\mathcal{I} = \emptyset$ and all g_i are linear, problem (1) is called a *linear program (LP)*.

With a slight abuse of notation, we will use the abbreviation LP for the term linear programming as well as for the term linear program. The same holds for MINLP, MIQCP, MIP, and NLP.

At the heart of many MIP improvement heuristics, such as Local Branching [33], RINS [31], and DINS [36], lies *large neighborhood search (LNS)*, the paradigm of solving a small sub-MIP which promises to contain good solutions. Recently, those LNS improvement heuristics have been extended to the more general case of MINLP [13,21,51]. In contrast, Undercover [11,12] is an LNS start heuristic for MINLP that does not have an equivalent in MIP.

In this paper, we introduce the *relaxation enforced neighborhood search* (RENS), a large neighborhood search algorithm for MINLP. It constructs a sub-MINLP of a given MINLP based on the solution of a relaxation. RENS is designed to compute the optimal—w.r.t. the original objective function—rounding of a relaxation solution.

Many LNS heuristics, diving and of course all rounding heuristics are based on the idea of fixing some of the variables that take an integral value in the relaxation solution. Therefore, the question of whether a given solution of a relaxation is *roundable*, i.e., whether all variables with a fractional solution value can be shifted to integral values without losing feasibility for the constraint functions, is particularly important for the likelihood of other primal heuristics to succeed.

We use RENS to analyze the roundability of instances from different classes of mathematical programs, demonstrate the computational impact of using different relaxations, and use these results to evaluate the performance of several rounding heuristics from the literature. Finally, we investigate the effectiveness of RENS applied as a start heuristic at the root node of a branch-and-cut solver. For these experiments, we use general, publicly available MIP, MIQCP and MINLP test sets obtained from the MIPLIB 3.0 [19], the MIPLIB 2003 [1], the MIPLIB 2010 [44], the MINLPLib [24] and the MIQCP test set compiled in [18].

The remainder of the paper is organized as follows. Section 2 introduces the generic scheme of the RENS algorithm. In Sect. 3, we discuss the algorithmic design and describe implementation details, in particular for the application of RENS as a subsidiary method inside a complete solver. The setup for the computational experiments is presented in Sect. 4. Section 5 provides detailed computational results and Sect. 6 contains our conclusions.

2 A scheme for an LNS rounding heuristic

Given a mixed integer program, the paradigm of fixing a subset of the variables in order to obtain subproblems that are easier to solve has proven successful in many MIP improvement heuristics such as RINS [31], DINS [36], Mutation, and Crossover [15,53]. These strategies can be directly extended to MINLP, see [13].

For a given MINLP, the NLP which arises by omitting the integrality constraints ($x_j \in \mathbb{Z}$ for all $j \in \mathcal{I}$) is called the *NLP relaxation* of the MINLP. The LP relaxation of a MIP is defined analogously. For a point $\bar{x} \in [L, U]$ (i.e., $L_j \leq \bar{x}_j \leq U_j$ for all $j \in \mathcal{N}$) the set of all *fractional variables* is defined as $\mathcal{F} := \{j \in \mathcal{I} \mid \bar{x}_j \notin \mathbb{Z}\}$.

Before we formulate the RENS algorithm, let us formalize the notion of an (optimal) rounding:

Definition 1 (*rounding*) Let $\bar{x} \in [L, U]$. The set

$$\mathcal{R}(\bar{x}) := \{x \in \mathbb{R}^n \mid x_j \in \{\lfloor \bar{x}_j \rfloor, \lceil \bar{x}_j \rceil\} \text{ for all } j \in \mathcal{I}, L_j \leq x_j \leq U_j \text{ for all } j \in \mathcal{N}\}$$

is called the set of *roundings* of $\bar{x}$.

In general, $\mathcal{R}(\bar{x})$ is a mixed integer set, a disjoint union of $2^{|\mathcal{F}|}$ polyhedra. Note that in the special case of $\mathcal{I} = \mathcal{N}$, so-called pure integer problems, the set of roundings of $\bar{x}$ is a $2^{|\mathcal{F}|}$-elementary lattice, more precisely, the vertices of an $|\mathcal{F}|$-dimensional unit hypercube:

$$\mathcal{R}(\bar{x}) = \left\{ x \in \mathbb{Z}^n \mid x_{\mathcal{I}\setminus\mathcal{F}} = \bar{x}_{\mathcal{I}\setminus\mathcal{F}}, x_{\mathcal{F}} \in \bigtimes_{j \in \mathcal{F}} \{\lfloor \bar{x}_j \rfloor, \lceil \bar{x}_j \rceil\} \right\} \subseteq \bigtimes_{j \in \mathcal{I}} \{L_j, \ldots, U_j\}.$$

Here, $x_{\mathcal{F}}$ and $x_{\mathcal{I}\setminus\mathcal{F}}$ denote the projection of x to the space of fractional and integral variables, respectively.

Definition 2 (*optimal rounding*) Let $\bar{x} \in [L, U]$ and $\tilde{x} \in \mathcal{R}(\bar{x})$.

1. We call $\tilde{x}$ a *feasible rounding* of $\bar{x}$, if $g_i(\tilde{x}) \leq 0$ for all constraints $i \in \mathcal{M}$ of MINLP (1).
2. We call $\tilde{x}$ an *optimal rounding* of $\bar{x}$, if $\tilde{x} \in \operatorname{argmin}\{d^T x \mid x \in \mathcal{R}(x), g_i(x) \leq 0 \text{ for all } i \in \mathcal{M}\}$.
3. We call $\bar{x}$ *roundable* if it has a feasible rounding.

The idea of our newly proposed LNS algorithm is to define a sub-MINLP that optimizes over the set of roundings of a relaxation optimum $\bar{x}$. This is done by fixing all integer variables that take an integral value in $\bar{x}$. For the remaining integer variables, the bounds get tightened to the two nearest integral values, see Fig. 1. Note that in the case of a completely fractional relaxation solution to a problem where all integer variables are *binary*, the subproblem would be identical to the original. We will therefore use a threshold for the percentage of integral variables, see next section.

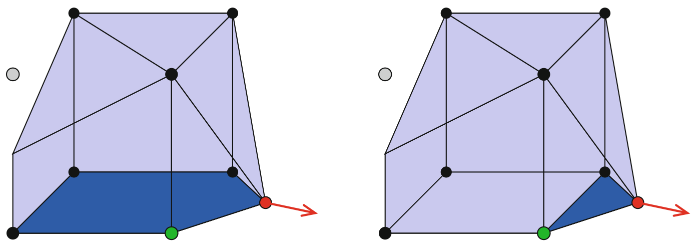

Fig. 1 RENS for MIP: original MIP (*light*), sub-MIP received by fixing (*dark*, *left*) and 0-1 sub-MIP by additional bound reduction (*dark*, *right*)

```
   Input: MINLP $P$ as in (1)
   Output: feasible solution $\tilde{x}$ for $P$ or $\emptyset$
1  begin
       /*compute optimal solution of the NLP relaxation of $P$                    */
2      $\bar{x} \leftarrow \text{argmin}\{d^T x \mid g_i(x) \le 0 \text{ for all } i \in \mathcal{M}, x \in [L, U]\}$;
3      forall $j \in \mathcal{I}$ do
4          if $\bar{x}_j \in \mathbb{Z}$ then
5              fix $x_j$: $L_j \leftarrow \bar{x}_j, U_j \leftarrow \bar{x}_j$;
6
7          else
8              change to binary bounds: $L_j \leftarrow \lfloor \bar{x}_j \rfloor$, $U_j \leftarrow \lceil \bar{x}_j \rceil$;
9
       /*solve the resulting sub-MINLP of $P$                                      */
10     $\tilde{x} \leftarrow \text{argmin}\{d^T x \mid g_i(x) \le 0 \text{ for all } i \in \mathcal{M}, x \in [L, U], x_j \in \mathbb{Z} \text{ for all } j \in \mathcal{I}\}$;
11     return $\tilde{x}$ ;
12 end
```

Fig. 2 Generic RENS algorithm

If the sub-MINLP is solved by using a linear outer approximation, tightening the variable bounds to the nearest integers often improves the dual bound, since reduced domains give rise to a tighter linear relaxation. Technically, all integer variables with tightened bounds can be easily transformed to binary variables, by substituting $x'_j = x_j - L_j$. Binary variables are preferable over general integers since many MIP-solving techniques such as probing [54], knapsack cover cuts [5,40,62], or the OCTANE heuristic [6] are only used for binary variables.

As the sub-MINLP is completely defined by the relaxation solution $\bar{x}$, we call the procedure *relaxation enforced neighborhood search*, or shortly RENS. Figure 2 shows the basic algorithm, which by construction has some important properties:

Proposition 1 *Let the starting point $\bar{x}$ be feasible for the NLP relaxation.*

1. *A point $\tilde{x}$ is a feasible solution of the sub-MINLP if and only if it is a feasible rounding of $\bar{x}$, in particular:*
2. *an optimal solution of the sub-MINLP is an optimal rounding of $\bar{x}$, and*
3. *if the sub-MINLP is infeasible, then no feasible rounding of $\bar{x}$ exists.*

Two major features distinguish RENS from other MIP and MINLP primal heuristics known from the literature. Firstly, the RENS algorithm does not require a known feasible solution, unlike other large neighborhood search heuristics that have been described for MIP, namely RINS [31], Local Branching [33], Crossover [15,53], DINS [36], or Proximity Search [32]. It is a start heuristic, not an improvement heuristic. The same holds for nonlinear variants of these heuristics [13,21,51].

Secondly, RENS solves a *single* sub-*MINLP*. In contrast, most primal heuristics for MINLP, in particular the various nonlinear feasibility pump versions [23,21,28,30], RECIPE [46] and Iterative Rounding [52], solve a *sequence* of auxiliary *MIPs*, often alternated with a sequence of NLPs, to produce a feasible start solution. The number of iterations is typically not fixed, but depends on the instance at hand.

3 Design and implementation details

In this section, we discuss the details of our RENS implementation. A particular focus is set on the application of RENS as a subsidiary method inside a complete branch-and-bound solver.

In principle, an arbitrary point may be used as starting point in line 2 of the RENS algorithm, see Fig. 2. Most complete solvers for MINLP are based on branch-and-bound and involve the solution of an NLP relaxation or a linear outer approximation. Their optima are natural choices as starting points. While the NLP optimum is supposed to be "closer" to the feasible region of the MINLP, the LP can usually be computed faster and often gives rise to smaller subproblems. More precisely, the NLP fulfills all nonlinear constraints $g_i(x) \leq 0$, whereas the LP, if solved with the simplex algorithm, tends to fulfill more integrality constraints, which reduces the computational complexity of the RENS subproblem. Thus, both relaxations have their pros and cons; which one proves better in practice will be investigated in our empirical studies, see Sect. 5.

When using a linear outer approximation (the LP relaxation in case of MIP), an important question is whether we should use the optimum of the initial LP relaxation or the LP solution after cutting planes have been applied. As before, cutting planes strengthen the formulation, but it is generally assumed that they tend to produce more fractional LP values. Which relaxation works best in practice shall be examined in the computational experiments in Sect. 5.

If RENS is used as a primal heuristic embedded in a complete solver, further modifications are necessary to obtain a good overall performance. When primal heuristics are considered as standalone solving procedures, e.g., the Feasibility Pump [3,9,23,28,30,34,35], the algorithmic design typically aims at finding feasible solutions for as many instances as possible, even if this takes substantial running time. However, if they are used as supplementary procedures inside a complete solver, the overall performance of the solver is the main objective. To this end, it is often worth sacrificing success on a small number of instances for a significant saving in average running time. The Stage 3 of the Feasibility Pump[1] is a typical example of a component that is crucial for its impressive success rate as a standalone algorithm, but it will

[1] Stage 3 of the Feasibility Pump solves (a reformulation of) the original MIP with a new objective function. It minimizes the distance to an infeasible point gained from the pumping algorithm; more precisely to the one which was closest to the polyhedron associated to the LP relaxation. For details, see [9].

not be applied when the Feasibility Pump is used inside a complete solver, see [15]. RENS principally is an expensive algorithm that solves an $\mathcal{NP}$-hard problem; therefore, the decision of when to call it should be made carefully to avoid slowing down the overall solving process. The remainder of this section describes some algorithmic enhancements, most of which are concerned with identifying which subproblems are the most promising for calling RENS and on which subproblems it should be skipped.

First, RENS should only be called if the resulting sub-MINLP seems to be substantially easier than the original one. This means that at least a specific ratio of all integer variables, say $r_1 \in (0, 1)$, or a specific ratio of all variables including the continuous ones, say $r_2 \in (0, 1)$, should be fixed. The first criterion limits the difficulty of the discrete part of the sub-MINLP itself, the second one limits the total size of the relaxations that will have to be solved. 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-MIP would be nearly as expensive as solving the ones of the original MIP. Since by propagation, fixing integer variables might also lead to fixed continuous variables, e.g., for variable bound constraints, we check the latter criterion only after presolving the subproblem.

Second, the sub-MINLP does not have to be solved to proven optimality. Therefore, we decided to use limits on the solving nodes and the so-called *stalling nodes* of the sub-MINLP. The absolute solving node limit l_1 is a hard limit on the maximum number of branch-and-bound nodes that should be processed. The stalling node limit l_2 indicates how many nodes should at most be processed without an improvement to the incumbent solution of the sub-MINLP.

Third, the partial solution of the sub-MINLP aims at finding a good primal solution quickly. Hence, algorithmic components that mainly improve the dual bound, such as cutting plane separation, and that are computationally very expensive, such as strong branching, can be disabled or reduced to a minimum. Further on this list are conflict analysis, pairwise presolving of constraints, probing and other LNS heuristics. As branching and node selection strategies we use inference branching and best estimate search, see, e.g. [2].

RENS could be either used as a pure start heuristic, calling it exclusively at the root node, or frequently throughout the branch-and-bound search to find rounded solutions of local LP optima. In particular when the integrality of the root LP relaxation falls below the minimum fixing ratio r_1, it seems reasonable to employ RENS at deeper levels of the tree where the number of fractional variables tends to be smaller. For the case of repeated calls of RENS, we implemented a few strategies to determine the points at which RENS should be called.

How often RENS should be called mainly depends on two factors: how expensive is it for a particular instance and how successful has it been in previous calls for that instance? The first can be estimated by the sum n_{RENS} of branch-and-bound nodes RENS used in previous calls in comparison to n_{all}, the number of branch-and-bound nodes already searched in the master problem. The second can be measured by the success rate $s = \frac{n_{\text{sols}}+1}{n_{\text{calls}}+1}$, where n_{calls} denotes the number of times RENS has been called and n_{sols} denotes the number of times it contributed an improving solution, respectively. In our implementation, we computed the stalling nodes limit as

$$l_2 = 0.3n_{\text{all}} \cdot s - n_{\text{RENS}} + 500 - 100n_{\text{calls}}.$$

The last term represents the setup costs for the subproblem which accrue even if subproblem solving terminates quickly. The offset of 500 nodes ensures that the limit is reasonable for the first few calls of RENS. We only start RENS if l_2 is sufficiently large. Thus, the stalling node limit considers different aspects of the previous behavior of the heuristic, starting with a budget of 500 nodes when called first at the root node. The individual parameter choices (0.3, 100, 500) are ad-hoc values and have not undergone extensive computational testing.

In an LP-based branch-and-bound search, consecutive nodes tend to have similar LP optima. This is due to the similarity of the solved problems as well as to the warm-starting technique of the simplex algorithm, which is typically used for this purpose. Since similar LP optima most likely lead to similar results for the quite expensive RENS heuristic, it should not be called in consecutive nodes, but the calls should rather be spread equally over the tree. Therefore, we use a calling frequency f: RENS only gets called at every f-th depth of the tree.

4 Experimental setup

This section proposes four computational experiments that analyze the potential of RENS to find optimal rounded solutions, evaluate the performing of rounding heuristics, compare RENS to other primal heuristics, and demonstrate the impact of RENS inside a full-scale branch-and-bound solver. We conduct these experiments on three different test sets of MIPs, MIQCPs, and MINLPs in order to analyze RENS on different classes of mathematical programs. All test sets are compiled from publicly available libraries.

Few existing softwares solve general nonconvex MINLPs to global optimality, including BARON [58], COUENNE [8], and LindoGlobal [47]. Others, such as BONMIN [22] and SBB [55], guarantee global optimality only for convex problems, but can be used as heuristic solvers for nonconvex problems. For a comprehensive survey of available MINLP solver software, see [25,29]. Recently, the solver SCIP [4,56] was extended to solve nonconvex MIQCPs [17] and MINLPs [59] to global optimality. SCIP is currently one of the fastest noncommercial solvers for MIP [44,50], MIQCP [50] and MINLP [59].

For all computational experiments, we used SCIP 2.1.1.1 compiled with SoPlex 1.6.0 [63,57] as LP solver, Ipopt 3.10 [60,42] as NLP solver, and CppAD 20110101 [27] as expression interpreter for evaluating general nonlinear constraints.The results were obtained on a cluster of 64bit Intel Xeon X5672 CPUs at 3.20 GHz with 12 MB cache and 48 GB main memory, running an openSuse 11.4 with a GCC 4.5.1 compiler. Hyperthreading and Turboboost were disabled. In all experiments, we ran only one job per node to avoid random noise in the measured running time that might be caused by cache-misses if multiple processes share common resources.

Test sets. We used all instances from MIPLIB3.0 [19], MIPLIB2003 [1], and MIPLIB2010 [44] as MIP test set. We excluded instances `air03`, `ex9`, `gen`, `manna81`, `p0033`, `vpm1`, for which the optimum of the LP relaxation (after SCIP presolving) is already integral, instance `stp3d`, for which SoPlex cannot solve the

LP to optimality within the given time limit and instances sp97ar, mine-166-5, for which SoPlex 1.6.0 fails in computing an optimal LP solution. This leaves 159 instances. We will refer to this test set as MIPLIB.

For MIQCP, we used the test set described in [18] that is comprised of instances from several sources. We excluded instances ex1263, ex1265, sep1, uflquad-30-100, for which the LP optimum is already integral (but in none of the cases feasible for the quadratic constraints), instances nuclear14, isqp1, nuclearva, for which the LP relaxation is unbounded, instance 200bar, for which SoPlex produces an error, 108bar, isqp0, for which SCIP's separation loop has not terminated within the time limit, and those 18 instances that are linear after SCIP presolving, see [18]. This test set contains 70 instances.

We further tested RENS on general MINLPs from MINLPLib [24], excluding those that are MIQCPs, that are linear after SCIP presolving, or that contain expressions which cannot be handled by SCIP, e.g., sine and cosine. We also excluded 4stufen, csched1a, st_e35, st_e36, waters, for which the optimum of the LP relaxation is integral, and instances csched2, minlphix, uselinear, for which the LP relaxation is unbounded, leaving 105 instances. It remains to be said that this test set is not as balanced as the others, since there are many instances of similar type.[2]

Analyzing roundability and computing optimal roundings. In a first test, we employ RENS to analyze the roundability of an optimal relaxation solution. For this, we run RENS without any node limits or variable fixing thresholds on the test sets described above. A time limit of two hours, however, was set for solving the RENS subproblem.

We used the optimum of the LP relaxation as starting point for the MIP test. We compare the performance of RENS using the "original" LP optimum before the cutting plane separation loop versus the one after cuts. One question of interest here is how the integrality of the LP solution interacts with the feasibility of the sub-MIP. The desired situation is that the LP solution contains a lot of integral values, but still gives rise to a feasible RENS problem.

For the MIQCP and the MINLP test run, we further evaluate how different types of relaxations, the LP and the NLP relaxation, behave w.r.t. the roundability of their optima and the quality of the rounded solutions. The results shall give an insight into which solutions should be used as starting points for RENS and other primal heuristics. Here, the performance in terms of running time of the RENS heuristic has to be weighed up against the success rate and quality of solutions produced with different relaxations.

Evaluating the performance of rounding heuristics. In a second test, we use RENS for the analysis of several rounding heuristics. The results shall give an insight into how often these heuristics find a feasible rounding and how good the quality of this solution is w.r.t. the optimal rounding.

All considered rounding heuristics iteratively round all variables that take a fractional value in the optimum of the relaxation. One rounding is performed per iteration step, without resolving the relaxation.

[2] This holds, to a certain extent, for all general MINLP test sets that the author is aware of.

- *Simple Rounding* only performs roundings, that maintain feasibility;
- *ZI Round* conducts roundings, using row slacks to maintain primal feasibility;
- *Rounding* conducts roundings, that potentially violate some constraints and reduces existing violations by further roundings;

ZI Round and Rounding both are extensions of Simple Rounding. Both are more powerful, but also more expensive in terms of running time.

For more details on ZI Round, see [61], for details on the other rounding heuristics implemented in SCIP, see [15].

Note that these heuristics are quite defensive, in the sense that they often round opposite to the variable's objective function coefficient and sacrifice optimality for feasibility. Hence, we do not expect them to often detect the optimal rounding computed by RENS. The question is rather for how many of the roundable instances these heuristics find any feasible solution and only as a second point how big the gap to the optimal rounding is.

RENS *compared to other primal heuristics.* In a third test, we compare RENS to other primal heuristics embedded in SCIP and called at the root node. We measure RENS against the complete portfolio of root node heuristics and against the single best heuristic (as implemented in SCIP). In the case of MIP, this was the Feasibility Pump [34,3]; in the case of MIQCP and MINLP, this turned out to be Undercover [11,12].

SCIP applies eleven primal heuristics at the root node: three rounding heuristics (see previous experiment), three propagation heuristics, a trivial one, a feasibility pump, an improvement heuristic, a large neighborhood search, and a repair heuristic. The latter two only come into play for nonlinear problems. This experiment is done to check whether SCIP is competitive with heuristics that are more involved than the rounding heuristics from the previous experiment.

Impact of RENS *on the overall performance of* SCIP. In our final experiment, we evaluate the usefulness of RENS when applied as a primal heuristic inside a branch-and-bound solver. For comparison see the RINS algorithm [31], an improvement heuristic which is applied in CPLEX and GUROBI. The advantage of RENS in contrast to RINS is that it does not require a given primal solution and that it always fixes at least the same number of variables as RINS, if applied to the same relaxation solution. The advantage of RINS is that the RINS subproblem is guaranteed to contain at least one feasible solution, namely the given starting solution.

To assess RENS as a primal heuristic, we run SCIP with RENS applied exclusively as a root node heuristic and SCIP with RENS applied both at the root and throughout the search. For this experiment, we used a reduced version of RENS which requires a minimal percentage of variables to be fixed and which stops after a certain number of branch-and-bound nodes, see Sect. 3. For comparison, we ran SCIP with RENS deactivated.

The main criteria to analyze in this test are the impact of RENS on the quality of the primal bound early in the search and the impact of RENS on the overall performance. While we hope for improvements in the former, a major improvement in the latter is not to be expected. Different studies show that the impact of primal heuristics on time to optimality often is slim. Bixby et al. report a deterioration of only 9 % if deactivating all primal heuristics in CPLEX 6.5, Achterberg [2] presents a performance loss of 14 % when performing a similar experiment with SCIP 0.90i, in [15] differences of at most 5 % for deactivating single primal heuristics are given. Therefore, a good result for this experiment would be an improvement on the primal bound side, coming with no deterioration to the overall performance.

5 Computational results

As a first test, we ran RENS without node or variable fixing limits, to evaluate its potential to find optimal roundings of optimal LP and NLP solutions.

The results for MIP can be seen in Tables 4 and 5 in the online supplement of this paper, those for MIQCP in Tables 6 and 7, those for MINLP in Tables 8 and 9; aggregated results can be found in Table 1.

The correlation between the percentage of fixed variables and the success rate of RENS is depicted in Figs. 3, 4, 5 and 6. Each instance is represented by a cross, with the fixing rate being the x-coordinate, and 0 or 1 representing success or failure as y-coordinate. The dotted blue line shows a moving average taken over ten consecutive points and the dashed red line shows a moving average taken over 30 consecutive points. A thin gray line is placed at the average success rate taken over all instances of the corresponding test set.

Table 1 Computing optimal roundings (aggregated results)

	Integrality		Succ	Prim. gap		Comp. effort	
	>90 %	Avg (%)		Mean	Std dev	Nodes	Time (s)
MIP + cuts	55/159	71.7	95/159	8.22	20.34	814.4	22.6
MIP − cuts	62/159	73.6	80/159	10.88	21.19	719.9	21.7
MIQCP (LP)	9/70	59.9	49/70	11.61	13.14	627.7	30.9
MIQCP (NLP)	1/70	13.8	48/70	1.30	3.37	7,078.8	168.1
MINLP (LP)	6/105	63.5	65/105	13.80	17.73	11,175.6	83.0
MINLP (NLP)	1/105	15.0	73/105	4.60	14.99	93,908.0	262.7

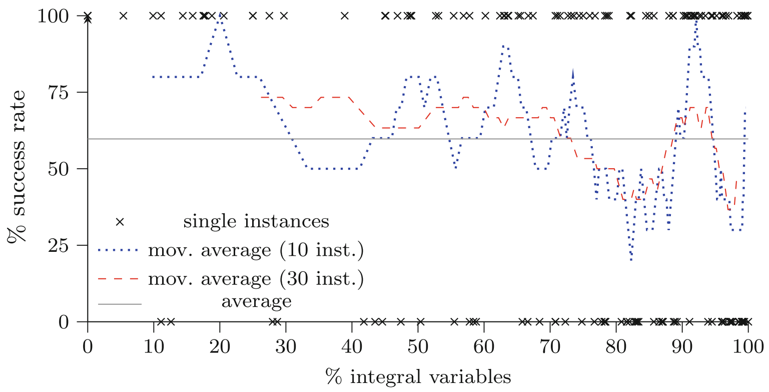

Fig. 3 Moving averages of success rate, MIPLIB instances, after cuts

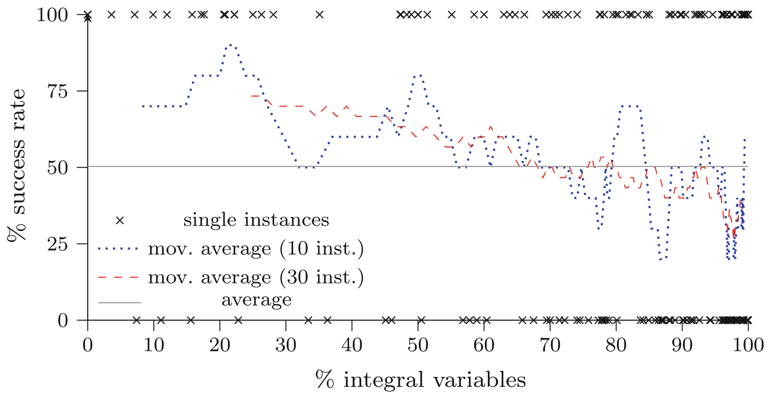

Fig. 4 Moving averages of success rate, MIPLIB instances, before cuts

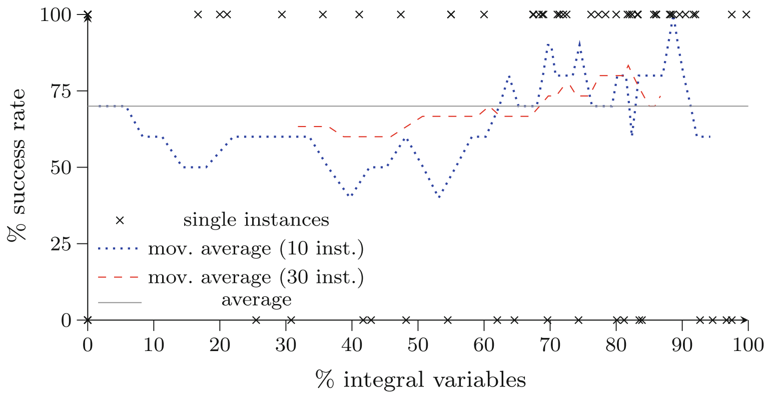

Fig. 5 Moving averages of success rate, MIQCP instances, LP sol., after cuts

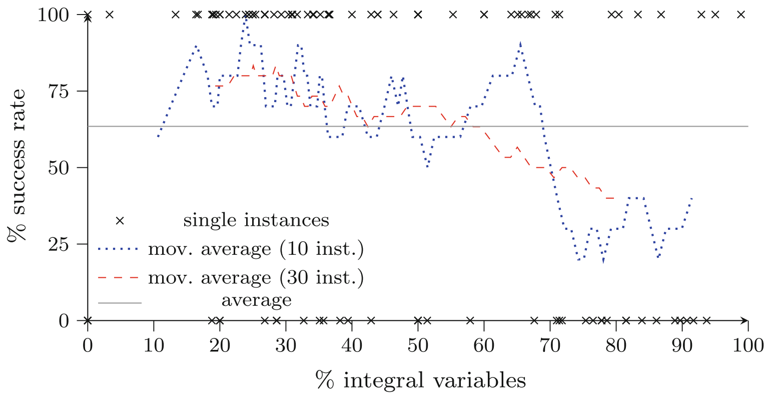

Fig. 6 Moving averages of success rate, MINLP instances, LP sol., after cuts

If we have to average running times or number of branch-and-bound nodes, we use a *shifted geometric mean*. The shifted geometric mean of values $t_1, \ldots, t_n$ with shift s is defined as $\sqrt[n]{\prod(t_i + s)} - s$. We use a shift of $s = 10$ for time and $s = 100$ for nodes in order to reduce the effect of very easy instances in the mean values. Further, using a *geometric* mean prevents hard instances at or close to the time limit from having a huge impact on the measures. Thus, the shifted geometric mean has the advantage that it reduces the influence of outliers in both directions.[3] In the given mean numbers, instances hitting the time limit are accounted for with the time limit and the number of processed nodes at termination.

In Table 1, Columns "$>$90 %" and "Avg (%)" show the number of instances for which more than 90 % of the integer variables were integral and the average percentage of integer variables taking integral values, respectively. Column "Succ" depicts the number of instances for which RENS found a feasible rounding. Columns "Mean" and "Std dev" show the arithmetic mean and the standard deviation of the primal gap (see, e.g., [14]), taken over those instances for which RENS found a feasible rounding. Note that these numbers have to be taken with a grain of salt, since even within one problem class they have been computed w.r.t. different instance sets for different settings. Columns "Nodes" and "Time (s)" give the shifted geometric means of the branch-and-bound nodes and running time required for solving the RENS subproblems.

Unless otherwise noted, the term variables will always refer to *integer* variables for the remainder of this section.

Computing optimal roundings: MIP. In Table 4 in the online supplement, we see that for roughly one third ($^{55}/_{159}$) of the instances, more than 90 % of the variables took an integral solution in the optimal LP solution. In contrast to that, there are only 22 instances for which the portion of integral solution values is less than 40 %. The average percentage of variables with integral LP solution value is 71.7 %. There are a few cases with many continuous variables for which fixing the majority of the integer variables did not result in a large ratio of all variables being fixed, see, e.g., `dsbmip` or `p5_34`. This is the reason that we will use two threshold values for later tests, see Sect. 3.

For 59.7 % ($^{95}/_{159}$) of the instances, RENS found a feasible rounding of the LP optimum. For 15 of these instances, the RENS subproblem hit the time limit, eleven of them are from MIPLIB 2010. For the remaining 80 instances, the solutions are optimal roundings of the given starting solutions. For 34 instances, the optimal rounding coincides with the global optimal solution.

We further observe that the success rate is only weakly correlated to the ratio of fixed variables. The success rate on the instances with more than 90 % fixed variables was nearly the same as on the whole test set, namely 58.2 %. This is an encouraging result for using RENS as a start heuristic inside a complete solver: very small subproblems contain feasible solutions.

The connection between the fixing rate and the success rate is also depicted in Fig. 3. We see that the success rate decreases slightly, at about 75 % fixed variables, but the difference between low and high fixing rates is not huge.

We further observe that proving the non-existence of a feasible rounding is relatively easy in most cases. For 59 out of 64 infeasible rounding subproblems, infeasibility could be proven in presolving or while root node processing of the subproblem. There

[3] For a detailed discussion of the shifted geometric mean, see Achterberg [2, Appendix A3].

is only one instance, `pigeon-10`, for which proving infeasibility takes more than 600 nodes. Considering the running time, infeasibility could be proven in less than a second in 56 of 64 cases, with only one instance, `app1-2`, taking more than 15 seconds. The instance `neos-1601936` is the only one for which feasibility could not be decided within the given time limit; hence, it is the only instance for which we could not decide whether the optimal LP solution is roundable or not.

The results for using the LP optimum before cutting plane separation are shown in Table 5 in the online supplement. Even more instances, 62 compared to 55, have an integral LP solution for more than 90 % of the variables. However, there is also one more (24 vs. 23) instance, for which the portion of integral solution values is less than 40 %. Contrary to what one might expect, the average percentage of variables with integral LP value is hardly affected by cutting plane separation: it is 73.6 % before separation and 71.7 % after.

The number of instances for which RENS found a solution, however, goes down: 80 instead of 95, which is only half of the test set. This is particularly due to those instances with many variables that take an integral value. Consequently, the success rate of RENS drops with an increase in the ratio of fixed variables. When RENS is called before cutting planes are added, fewer of the optimal roundings are optimal solutions to the original problem: 20 compared to 34, when called after cuts.

We conclude that, although the fractionality is about the same, LP solutions before cuts are less likely to be roundable and the rounded solutions are often of inferior quality. In other words: before cutting planes, integral solution values are more likely to be misleading (in the sense that they cannot be extended to a good feasible solution). This is an important result for the design of primal heuristics in general and confirms the observation that primal heuristics work better after cutting plane separation, see, e.g., [35].

Computing optimal roundings: MIQCP. For MIQCP, we tested RENS with LP solutions and with NLP solutions as starting points, see Tables 6 and 7, respectively, in the online supplement. We also experimented with the LP solution before cuts; the results were much worse and are therefore not shown.

The ratio of integral LP values is smaller compared to the MIP problems: there are only 9 out of 70 instances for which more than 90 % of the variables were integral, but there are 10 instances for which all variables were fractional. Note that this does not necessarily mean that the RENS sub-MIQCP is identical to the original MIQCP, cf. the presence of general integer variables. In this case, the RENS subproblem corresponds to the original problem intersected with the integral lattice-free hypercube around the starting solution. On average, 59.9 % of the variables took an integral value. The success rate of RENS is even better than for MIPs: In 49 out of 70 instances (70 %), RENS found a feasible rounding. Note that this is not due to the 10 instances for which all variables were fractional: three of them also fail. Moreover, the success rate appears not to depend on the percentage of fixed variables, see Fig. 5.

Deciding feasibility, however, seems to be more difficult. Out of ten instances hitting the time limit, there were eight for which RENS did not find a feasible rounding. For 13 instances, infeasibility of the rounding problem was proven, mostly in presolving or within a few branch-and-bound nodes. Nine times, the optimal rounding was identical to the optimal solution of the MIQCP.

The next observation we made is that the NLP solution tends to be much less integral than the LP solution, on average only 13.8 % of the variables take an integral value, see Table 7 in the online supplement and Fig. 5. This is due to the fact that in our experiments the LP solution was computed with the simplex algorithm which tends to leave variables at their bounds, whereas the NLP solution was computed with an interior point algorithm that tends to choose values from the interior of the variables' domains.

Surprisingly, this did not enhance the roundability. For 48 instances, RENS found a feasible rounding of the NLP optimum, compared to 49 for the LP. Worth mentioning, this was nearly the same set of instances, and there were 46 on which both versions found a solution. The solution quality, however, was typically better when using an NLP solution: 27 times the NLP solution yielded a better rounding, only once the LP was superior. 26 times, the optimal rounding was even an optimal solution of the original MIQCP.

The higher fractionality of the NLP relaxation is expressed in a much larger search space. In shifted geometric mean, RENS processed 628 search nodes if starting from an LP solution, 7,078 if starting from an NLP solution. The geometric mean of the running time is roughly 5.5 times larger: 30.9 vs. 168.1 s.

We conclude that the same observation holds as in the MIP case: small subproblems generate high-quality feasible solutions. Although the solution quality is improved by using an NLP relaxation, the computational overhead and the success rate are not encouraging to make this a standard setting if using RENS inside a complete solver.

Computing optimal roundings: MINLP. For MINLP, we again compared two versions of RENS: one using the LP solution and one using the NLP solution as starting point, see Tables 8 and 9, respectively, in the online supplement. For the same reason as in the MIQCP case we omitted the results for the LP solution before cuts.

The integrality of the LP solutions is comparable to the MIQCP case. On average, 63.5 % of the variables take an integral value; there are 6 out of 105 instances for which more than 90 % of the variables are integral, and only four instances for which all variables are fractional. For this test set, we see a clearer connection between the ratio of fractional variables and the success rate of RENS. The more variables are integral, the lower the chance for RENS to succeed, see Fig. 6.

For seven instances, the RENS subproblem hit the time limit of two hours, always without having found a feasible solution. Overall, 65 out of 105 (62 %) of the LP solutions proved to be roundable, which is similar to the MIP results. In all cases, RENS found the optimal rounding. Generally, RENS needs much more nodes to solve the rounding problem as compared to the other tests.

Using an NLP instead of an LP relaxation increases the success rate: 73 times, RENS finds a feasible rounding. As for MIQCPs, the quality is typically better (37 vs. 2 times), which comes with a much lower integrality of 15 % on average, 68 instances having all variables fractional, and a huge increase in running time: a factor of more than three in shifted geometric mean.

Computing optimal roundings: summary. Interestingly, the fractionality and roundability of LP solutions is very similar for MIPs, MIQCPs and MINLPs: on average, only 30–40 % of the variables are fractional and for 60–70 % of the instances RENS found a feasible rounding. We further observed that most often the RENS subproblem could be solved to proven optimality and that the success rate of RENS is only weakly correlated to the fractionality. These three insights are very encouraging for applying RENS as a start heuristic inside a complete solver, see below. A summary of the results on computing optimal roundings can be found in Table 1.

We further performed a McNemar test to analyze the statistical significance of the results. As null hypothesis we assume that the LP and the NLP solution (or the LP before and after cuts) are equally likely to yield a feasible rounding. For the MIP test set, the null hypothesis gets rejected with a p-value of 0.0011 and for MINLP with 0.0114. For MIQCP, the p-value is 0.6547. This means that for MIP the LP solution after cuts is more likely to be roundable with very high probability, for MINLP the NLP solution is more promising with high probability, for MIQCP there is no statistically significant difference.

We conclude that the solutions found by RENS are usually better when it is applied after cutting plane separation and that using an NLP instead of an LP relaxation gives more chances to find a feasible rounding for MINLPs, although this comes at a large computational cost. For applications where CPU time is critical, the NLP relaxation does not give a good trade-off between solution quality and running time.

Analyzing rounding heuristics. Our next experiment compares RENS applied to the LP solution after cuts with the three pure rounding heuristics that are implemented in SCIP. The results for the MIPLIB instances are given in Table 10 in the online supplement.

As implied by definition, the solutions found by RENS (if the subproblem has been solved to optimality) are always better or equal to the solutions produced by any rounding heuristic. As expected, the solution quality of Rounding and ZI Round is always better or equal to Simple Rounding, and ZI Round often is superior to Rounding. Since Simple Rounding, Rounding, and ZI Round all endeavor to feasibility and neglect optimality, it is not too surprising that there are only three instances, for which Simple Rounding and Rounding find an optimal rounding; four in the case of ZI Round.

A comparison of the number of solutions, however, shows that there is a big discrepancy between the number of instances which have a roundable LP optimum (95) and the number of instances for which these heuristics succeed (37 for ZI Round, 36 for Rounding, and 27 for Simple Rounding). Of course, this has to be seen under the fact that these heuristics are much faster than RENS. The maximum running time was attained by Rounding on instance `opm2-z7-s2`; it was only 0.09 s.

For the MIQCP and MINLP test sets, the situation was even more extreme. The rounding heuristics were unable to produce a feasible solution for any of the instances—even though the previous experiments proved that 60–70 % of the LP solutions are roundable. This is most likely due to the special design of these heuristics—they solely work on the LP relaxation—and demonstrates the need for rounding heuristics that take the special requirements of nonlinear constraints into consideration.

RENS *compared to other primal heuristics.* This experiment compares RENS to other primal heuristics embedded in SCIP and called at the root node. For each of the three test sets, we evaluated three different settings of SCIP: one for which all default root node heuristics except RENS are employed, one for which only RENS is called, and one for which only the Feasibility Pump (for MIP) or only Undercover (for MIQCP and MINLP) is used.

Based on the results from our first experiment, considering the running times and the node numbers at which the RENS subproblems find their optimal solutions, we decided to use 50 % as a threshold value for r_1, the minimal fixing rate for integer variables, in this run. The minimal fixing rate for all variables r_2 was set to 25 %. We used an absolute node limit l_1 of 5,000 and computed the stalling node limit l_2 as given in Sect. 3. Because of the long running times, we refrained from using an NLP relaxation, although it might produce better solutions. We always used the LP solution after cutting planes as a starting solution. The results are given in Tables 11, 12 and 13 in the online supplement.

We observe that for all three test sets, RENS alone is inferior to the portfolio of root node heuristics, but superior to the single best heuristic in terms of problem instances for which a solution could be found. For the MIP instances, SCIP's root node heuristics found feasible solutions for 106 instances, RENS (with the described settings) for 56, the Feasibility Pump for 51. For MIQCP, the portfolio succeeded 56 times, RENS 33 times, Undercover 29 times. For MINLP the result was 28 for all, 12 for RENS, 6 for Undercover. Note that on this test set, as is typical for nonconvex MINLPs, finding a feasible solution is generally harder than for MIPs. Other solvers perform comparably: we additionally performed this root node test with the default settings and a time limit of two hours for COUENNE 0.4 [8] and BARON 11.1 [58]. They found feasible solutions for 35 and 40 instances, respectively.

There were two MIP instances, two MIQCP instances, and three MINLP instances, for which RENS found a feasible solution but the other SCIP root node heuristics did not. For a further 40, 17, and 5 instances, respectively, the solution found by RENS was better than the best solution produced by the other heuristics. We conclude that RENS is a valuable extension of SCIP's primal heuristic portfolio. Further, in terms of the number of solutions it produced when run as the only heuristic, it is comparable to other state-of-the-art primal heuristics, such as the Feasibility Pump (in an embedded version, compare Sect. 3) or Undercover.

Impact of RENS *on the overall performance of* SCIP. Finally, we evaluate whether a reduced version of the full RENS algorithm is suited to serve as a primal heuristic applied inside a complete solver. We use the same threshold settings as in the previous experiment. For this experiment, interactions of different primal heuristics among each other and with other solver components come into play. SCIP applies eleven primal heuristics at the root node. Of course, a primal heuristic called prior to RENS might already have found a solution which is better than the optimal rounding, or in an extreme case, the solution process might already terminate before RENS is called. Further, any solution found before RENS is called might change the solution path. It might trigger variable fixings by dual reductions, which lead to a different LP and hence to a different initial situation for RENS.

Table 2 RENS as primal heuristic inside SCIP (aggregated results), numbers of instances for which RENS was called and succeeded at least once

	At root		In tree	
	Called	Found	Called	Found
MIP (of 160)	124	63	154	87
MIQCP (of 70)	45	31	60	42
MINLP (of 105)	45	9	99	39

Table 3 RENS as primal heuristic inside SCIP (aggregated results), computational effort

	Arithmetic		Geometric		Shifted geom	
	Nodes	Time(s)	Nodes	Time(s)	Nodes	Time(s)
MIP No RENS	1,446,078	2,461.4	7,155	220.3	11,248	377.2
MIP Root RENS	1,442,400	2,427.0	5,870	209.6	10,390	366.3
MIP Tree RENS	1,443,404	2,414.3	5,810	209.4	10,346	365.8
MIQCP No RENS	659,740	2,872.3	3,823	84.5	6,457	229.9
MIQCP Root RENS	677,123	2,927.0	3,742	86.4	6,361	232.0
MIQCP Tree RENS	664,117	2,888.6	3,561	86.2	6,193	229.9
MINLP No RENS	2,338,903	3,274.5	45,334	288.0	58,758	466.5
MINLP Root RENS	2,324,208	3,274.7	44,723	291.4	58,406	467.1
MINLP Tree RENS	1,925,902	3,168.7	38,568	267.9	51,066	431.3

The results are shown in Tables 14, 15 and 16 in the online supplement. We compare SCIP without the RENS heuristic (No RENS) against SCIP with RENS applied at most once at the root node (Root RENS) and SCIP with RENS applied at every tenth depth of the branch-and-bound tree (Tree RENS).

A summary of the results is given in Tables 2 and 3. The Columns "called" and "found" in Table 2 show for how many instances RENS was called and found a feasible solution, respectively. Table 3 depicts the arithmetic means, the geometric means, and the shifted geometric means of the number of branch-and-bound nodes and the running time for each combination of the three different settings and the three test sets.

First, let us consider the results for MIP. Due to the a-priori limits, RENS was called at the root node for only 124 out of the 160 instances. Out of these, RENS found a feasible solution in 63 cases, which corresponds to a success rate of 50 %, compared to 59 % without any limits, see above. In 61 cases, this solution was the best solution found at the root node. Considering that there are ten other primal heuristics applied at the root node, this appears to be a very strong result. When RENS was additionally used during search, it was called on 154 instances, finding feasible solutions for 87 of them.

As is typical for primal heuristics, the impact on the overall performance is not huge. Nevertheless, we see that both versions, calling RENS only at the root and all over the tree, give slight decreases in the arithmetic and geometric means of the node numbers and the running time. Both versions were about 3 % faster and took 8 % less nodes in shifted geometric mean. For the time-outed instances, Root RENS and Tree RENS provided a better primal bound than No RENS eight and nine times, respectively, whereas both were inferior in two cases.

For the MIQCP test set, RENS was called at the root for 45 out of 70 instances, finding a feasible solution in 31 cases. This was always the best solution SCIP found at the root node. The overall performance was about the same: the running time stayed constant for Tree RENS and was increased by less than one percent for Root RENS, whereas the number of branch-and-bound nodes was reduced by 7 and 2 %, respectively. When RENS is called during search tree processing, there are four instances with a better primal bound at timeout, once it was worse. For calling RENS exclusively at the root, this ratio was 2:0. Also, there is one instance, namely `nuclear14a`, for which only Tree RENS provided a feasible solution.

For MINLP, the lower success rate for the root LPs with large ratios of integral variables is confirmed by this experiment. For 45 out of 105 instances, RENS was called, but in only nine cases it could improve the incumbent solution. Interestingly, the version that calls RENS during the tree performs really well. There were 42 instances, for which RENS could improve the incumbent at least once during search, `ghg_3veh` being the front-runner with 27 improving solutions in 44 calls of RENS.

The overall performance reflects that situation. The Root RENS setting shows the same behavior as No RENS, the running time is nearly equal on average and in geometric mean, the number of branch-and-bound nodes goes down by one percent, there are hardly any instances for which we see any change in performance. For Tree RENS, however, the geometric mean of the running time and the number of branch-and-bound nodes goes down by 8 and 13 %, respectively. One might argue that this is mainly because of `enpro48pb` and `fo8_ar4_1` which show a dramatic improvement in performance. But even if we excluded these two instances (and for fairness reasons also `enpro48` and `enpro56pb`, two outliers in the opposite direction), the mean performance gain is 3 % for running time and 8 % for number of branch-and-bound nodes.

We further performed a variant of the Wilcoxon signed rank test to analyze the statistical significance of the results, using the STATS package of the SCIPY project [43]. We ranked the results by the running time factors per instance and calculated one rank sum from the improving instances and one from those which showed a degradation. Instances that showed no or hardly any performance difference (<1 s or <1 %) were excluded. As null hypothesis, we assume that a version of SCIP using RENS at the root or throughout the tree does perform equally w.r.t. running time as SCIP without RENS. For the MIP test set, the null hypothesis gets rejected with a p-value of 0.0236 (for Root RENS) and 0.0178 (for Tree RENS) which is below the standard threshold of 0.05 used as level of significance. Not surprisingly, the results for MIQCP indicate that there are no performance differences for this test set: the p-values are 0.6465 and 0.8753 for Root RENS and for Tree RENS, respectively. For MINLP, p-values of 0.3980 and 0.2862 are achieved. Although failing to reject the null hypothesis when a standard threshold is applied, at least the latter could be taken as an indicator that it is more likely that the results are not simply acquired by chance.

Altogether, these experiments show that RENS, in particular for MIP and MIQCP, helps to improve the primal bound at the root node, and hence the initial gap before the branch-and-bound search starts. Applying RENS exclusively at the root node had a neutral to slightly positive effect on the overall performance, while giving a user the advantage of finding good solutions early. Applying RENS throughout the search was at least as good for all three test sets and showed a nice improvement in the case of MINLP—which was partly due to two outliers. Consequently, RENS is used in the default settings of SCIP. Furthermore, versions of RENS have been recently integrated into BONMIN [22] and CBC [26].

6 Conclusion

We introduced RENS, a large neighborhood search algorithm that, given a MIP or an MINLP, solves a subproblem whose solution space is the set of feasible roundings of a relaxation solution. We showed that most MIP, MIQCP, and MINLP instances have roundable LP and NLP optima and in most cases, the optimal roundings can be computed efficiently. Surprisingly, the roundability seems not to be related to the fractionality of the starting solution. Knowing the optimal roundings provides us with a benchmark for rounding heuristics; we discovered that the rounding heuristics implemented in SCIP often fail in finding a feasible solution, even though the provided starting point is roundable. They rarely find the optimal rounding.

We further investigated the impact of a reduced version of RENS if applied as a primal heuristic inside a complete solver. RENS directly helps to improve the primal bound known at the root node. The impact on the overall performance is minor but measurable, which is typical for primal heuristics.

RENS is part of the SCIP standard distribution and employed by default. The implementation presented in this article can be accessed in source code at [56].

Acknowledgments Many thanks go to Ambros M. Gleixner and Daniel E. Steffy for their thorough proofreading and to two anonymous reviewers for their helpful comments. This research has been supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.

References

1. Achterberg, T., Koch, T., Martin, A.: MIPLIB 2003. Oper. Res. Lett. 34(4), 1–12 (2006). http://miplib.zib.de/miplib2003/
2. Achterberg, T.: Constraint Integer Programming. PhD thesis, Technische Universität Berlin, 2007
3. Achterberg, T., Berthold, T.: Improving the feasibility pump. Discret Optim. **4**(1), 77–86 (2007). (Special Issue)
4. Achterberg, T.: SCIP: solving constraint integer programs. Math. Program. Comput. **1**(1), 1–41 (2009)
5. Balas, E.: Facets of the knapsack polytope. Math. Program. **8**, 146–164 (1975)
6. Balas, E., Ceria, S., Dawande, M., Margot, F., Pataki, G.: Octane: a new heuristic for pure 0–1 programs. Operat. Res. **49**, 207–225 (2001)
7. Balas, E., Schmieta, S., Wallace, C.: Pivot and shift—a mixed integer programming heuristic. Discret. Optim. **1**(1), 3–12 (2004)
8. Belotti, P., Lee, J., Liberti, L., Margot, F., Wächter, A.: Branching and bounds tightening techniques for non-convex MINLP. Optim. Methods Softw. **24**, 597–634 (2009)
9. Bertacco, L., Fischetti, M., Lodi, A.: A feasibility pump heuristic for general mixed-integer problems. Discret. Optim. **4**(1), 77–86 (2007). (Special Issue)
10. Berthold, T., Feydy, T., Stuckey, P.J.: Rapid learning for binary programs. In: Lodi, A., Milano, M., Toth, P. (eds.) Proceedings of CPAIOR 2010, vol 6140 of LNCS, pp. 51–55. Springer, Berlin (2010)
11. Berthold, T., Gleixner, A.M.: Undercover—a primal heuristic for MINLP based on sub-MIPs generated by set covering. In: Bonami, P., Liberti, L., Miller, A.J., Sartenaer, A. (eds.) Proceedings of the EWMINLP, pp. 103–112 (2010)

12. Berthold, T., Gleixner, A.M.: Undercover—a primal MINLP heuristic exploring a largest sub-MIP. Math. Program. A (2013). doi:10.1007/s10107-013-0635-2
13. Berthold, T., Heinz, S., Pfetsch, M.E., Vigerske, S.: Large neighborhood search beyond MIP. In: Gaspero, L.D., Schaerf, A., Stützle, T. (eds.) Proceedings of the 9th Metaheuristics International Conference (MIC 2011), pp. 51–60 (2011)
14. Berthold, T.: Measuring the impact of primal heuristics. Oper. Res. Lett. **41**(6), 611–614 (2013)
15. Berthold, T.: Primal heuristics for mixed integer programs. Technische Universität Berlin, Diploma thesis (2006)
16. Berthold, T.: Heuristics of the branch-cut-and-price-framework SCIP. In: Kalcsics, J., Nickel, S. (eds.) Operations Research Proceedings 2007, pp. 31–36. Springer-Verlag, Berlin (2008)
17. Berthold, T., Heinz, S., Vigerske, S.: Extending a CIP framework to solve MIQCPs. In: Lee, J., Leyffer, S. (eds.) Mixed Integer Nonlinear Programming. The IMA Volumes in Mathematics and its Applications, pp. 427–444. Springer, Berlin (2011)
18. Berthold, T., Gleixner, A.M., Heinz, S., Vigerske, S.: Analyzing the computational impact of MIQCP solver components. Numer. Algebra, Control Optim. **2**(4), 739–748 (2012)
19. Bixby, R.E., Ceria, S., McZeal, C.M., Savelsbergh, M.W.: An updated mixed integer programming library: MIPLIB 3.0. Optima **58**, 12–15 (1998)
20. Bixby, R.E., Fenelon, M., Gu, Z., Rothberg, E., Wunderling, R.: MIP: theory and practice—closing the gap. In: Powell, M., Scholtes, S. (eds.) Systems Modelling and Optimization: Methods, Theory, and Applications, pp. 19–49. Kluwer Academic Publisher, Dordrecht (2000)
21. Bonami, P., Gonçalves, J.: Heuristics for convex mixed integer nonlinear programs. Comput. Optim. Appl. **51**(2), 729–747 (2012)
22. Bonami, P., Biegler, L., Conn, A., Cornuéjols, G., Grossmann, I., Laird, C., Lee, J., Lodi, A., Margot, F., Sawaya, N., Wächter, A.: An algorithmic framework for convex mixed integer nonlinear programs. Disc. Opt. **5**, 186–204 (2008)
23. Bonami, P., Cornuéjols, G., Lodi, A., Margot, F.: A feasibility pump for mixed integer nonlinear programs. Math. Program. **119**(2), 331–352 (2009)
24. Bussieck, M., Drud, A., Meeraus, A.: MINLPLib—a collection of test models for mixed-integer nonlinear programming. INFORMS J. Comput. **15**(1), 114–119 (2003)
25. Bussieck, M.R., Vigerske, S.: MINLP solver software. In: Cochran, J.J., Cox, L.A., Keskinocak, P., Kharoufeh, J.P., Smith, J.C. (eds.) Wiley Encyclopedia of Operations Research and Management Science. Wiley (2010)
26. CBC user guide - COIN-OR. http://www.coin-or.org/Cbc
27. CppAD. A Package for Differentiation of C++ Algorithms. http://www.coin-or.org/CppAD/
28. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: Experiments with a feasibility pump approach for nonconvex MINLPs. In: Festa, P. (ed.) Experimental Algorithms. Lecture Notes in Computer Science, pp. 350–360. Springer, Berlin (2010)
29. D'Ambrosio, C., Lodi, A.: Mixed integer nonlinear programming tools: a practical overview. 4OR: Q. J. Operat. Res. **9**, 329–349 (2011)
30. D'Ambrosio, C., Frangioni, A., Liberti, L., Lodi, A.: A storm of feasibility pumps for nonconvex MINLP. Math. Program. **136**, 375–402 (2012)
31. Danna, E., Rothberg, E., Pape, C.L.: Exploring relaxation induced neighborhoods to improve MIP solutions. Math. Program. A **102**(1), 71–90 (2004)
32. Fischetti, M., Monaci, M.: Proximity search for 0–1 mixed-integer convex programming. Technical report, DEI, University of Padova (2012)
33. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming B **98**(1–3), 23–47 (2003)
34. Fischetti, M., Glover, F., Lodi, A.: The feasibility pump. Math. Program. A **104**(1), 91–104 (2005)
35. Fischetti, M., Salvagnin, D.: Feasibility pump 2.0. Math. Program. C **1**, 201–222 (2009)
36. Ghosh, S.: DINS, a MIP improvement heuristic. In: Fischetti, M., Williamson, D.P. (eds.) Integer Programming and Combinatorial Optimization (IPCO 2007), volume 4513 of LNCS, pp. 310–323 (2007)
37. Glover, F., Løkketangen, A., Woodruff, D.L.: Scatter search to generate diverse MIP solutions. In: Laguna, M., González-Velarde, J.L. (eds.) OR Computing Tools for Modeling, Optimization and Simulation: Interfaces in Computer Science and Operations Research, pp. 299–317. Kluwer Academic Publishers, Dordrecht (2000)
38. Glover, F., Laguna, M.: General purpose heuristics for integer programming—part I. J. Heuristics **2**(4), 343–358 (1997)

39. Glover, F., Laguna, M.: General purpose heuristics for integer programming—part II. J. Heuristics **3**(2), 161–179 (1997)
40. Hammer, P.L., Johnson, E.L., Peled, U.N.: Facets of regular 0–1 polytopes. Math. Program. **8**, 179–206 (1975)
41. Hansen, P., Mladenović, N., Urošević, D.: Variable neighborhood search and local branching. Comput. Operat. Res. **33**(10), 3034–3045 (2006)
42. Ipopt (Interior Point OPTimizer). http://www.coin-or.org/Ipopt/
43. Jones, E., Oliphant, T., Peterson, P., et al.: SciPy: open source scientific tools for Python (2001)
44. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Math. Program. Comput. **3**(2), 103–163 (2011)
45. Land, A.H., Doig, A.G.: An automatic method of solving discrete programming problems. Econometrica **28**(3), 497–520 (1960)
46. Liberti, L., Mladenović, N., Nannicini, G.: A recipe for finding good solutions to MINLPs. Math. Program. Comput. **3**(4), 349–390 (2011)
47. LindoGlobal. Lindo Systems, Inc. http://www.lindo.com
48. Løkketangen, A.: Heuristics for 0–1 mixed integer programming. In: Pardalos, P.M., Resende, M.G.C. (eds.) Handbook of Applied, Optimization, pp. 474–477. Oxford University Press, New York (2002)
49. Løkketangen, A., Glover, F.: Solving zero/one mixed integer programming problems using tabu search. Eur. J. Operat. Res. **106**, 624–658 (1998)
50. Mittelmann, H.: Decision tree for optimization software: Benchmarks for optimization software. http://plato.asu.edu/bench.html
51. Nannicini, G., Belotti, P., Liberti, L.: A local branching heuristic for MINLPs. ArXiv e-prints (2008)
52. Nannicini, G., Belotti, P.: Rounding-based heuristics for nonconvex MINLPs. Math. Program. Comput. **4**(1), 1–31 (2012)
53. Rothberg, E.: An evolutionary algorithm for polishing mixed integer programming solutions. INFORMS J. Comput. **19**(4), 534–541 (2007)
54. Savelsbergh, M.W.P.: Preprocessing and probing techniques for mixed integer programming problems. ORSA J. Comput. **6**, 445–454 (1994)
55. SBB. ARKI Consulting & Development A/S and GAMS Inc. http://www.gams.com/solvers/solvers.htm#SBB
56. SCIP. Solving Constraint Integer Programs. http://scip.zib.de/
57. SoPlex. An open source LP solver implementing the revised simplex algorithm. http://soplex.zib.de/
58. Tawarmalani, M., Sahinidis, N.: Global optimization of mixed-integer nonlinear programs: A theoretical and computational study. Math. Program. **99**, 563–591 (2004)
59. Vigerske, S.: Decomposition in Multistage Stochastic Programming and a Constraint Integer Programming Approach to Mixed-Integer Nonlinear Programming. PhD thesis, Humboldt-Universität zu Berlin (2013)
60. Wächter, A., Biegler, L.: On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Math. Program. **106**(1), 25–57 (2006)
61. Wallace, C.: ZI round, a MIP rounding heuristic. J. Heuristics **16**(5), 715–722 (2010)
62. Wolsey, L.A.: Faces for a linear inequality in 0–1 variables. Math. Program. **8**, 165–178 (1975)
63. Wunderling, R.: Paralleler und objektorientierter Simplex-Algorithmus. PhD thesis, Technische Universität Berlin (1996)