Primal Heuristics for Mixed Integer Programs with a Staircase Structure

Marco E. Lübbecke and Christian Puchert

Chair of Operations Research, RWTH Aachen University,
Kackertstr. 7, 52072 Aachen, Germany
{marco.luebbecke, christian.puchert}@rwth-aachen.de

Abstract. In the recent years, Mixed Integer Programmms (MIPs) have become a powerful tool to solve problems from a wide range of applications. When solving large-scale MIPs, primal heuristics have gained increasing attention: Ideally, a heuristic finds a good feasible solution in a short amount of time.
In this paper, we present three heuristic approaches for MIPs whose coefficient matrices have a so-called staircase structure. We will present and evaluate a generalization of a the rolling horizon heuristic that has recently been successful applied to several structured MIPs as well as a new approach called fifty/fifty. Furthermore, we will investigate whether so-called diving heuristics can be extended to staircase structures.

Keywords: Mixed Integer Programming, staircase matrices, primal heuristics

1 Introduction

We are solving Mixed Integer Programs of the form

$$\begin{aligned} \min\ & c^T x \\ \text{s.t. } & Ax \leq b \\ & x_j \in \mathbb{Z} \qquad j \in I, \end{aligned} \tag{1}$$

where $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, and $I \subseteq \{1, \ldots, n\}$. There are various problems in industry and economy that can be modeled in this fashion. Such problems include e. g. the resource allocation or lot sizing problems. In such problems, the variables refer to decisions on "nearly independent" time steps, except that there are some linking variables. When modeling this as a MIP, the matrix then has a certain structure:

Definition 1. *A MIP of the form* (1) *is said to have a* staircase structure *if its coefficient matrix A can be written in the form*

$$A = \begin{pmatrix} B_1 & R_1 & & & \\ L_1 & B_2 & R_2 & & \\ & & \ddots & & \\ & & L_{K-2} & B_{K-1} & R_{K-1} \\ & & & L_{K-1} & B_K \end{pmatrix} \tag{2}$$

For $1 \leq k \leq K$, the rows of (L_k, B_k, R_k) are called a stage*, with L_1 and R_K being empty. We will write the variable vector x as*

$$x = ((x^1)^T, (y^1)^T, \ldots, (x^{K-1})^T, (y^{K-1})^T, (x^K)^T)^T,$$

where x^k refers to the columns of B_k and y^k to the columns of R_k and L_k. The x_k variables will be called block variables*, while the y_k variables will be called* linking variables*.*

While in some applications have an "intuitive" staircase structure, one might ask if an arbitrary given MIP has this form. In the recent part, several successful efforts have been made to detect a "hidden" staircase structure in a MIP, possibly after permuting rows and columns [10]. How this works, however, is not the topic of this paper, so we will not go into further detail here. In Fig. 1, a staircase structure in MIPLIB2010 instance `ic97_potential` is shown.

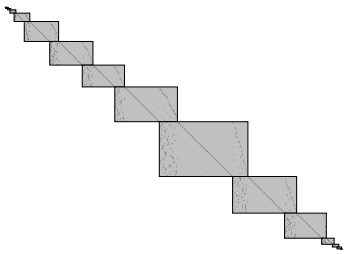

Fig. 1. Staircase structure in MIPLIB2010 instance `ic97_potential`. Each dot corresponds to a matrix coefficient, and each gray box marks a stage

Note that there needs not to be "the" staircase structure; for a given MIP, there may exists multiple such structures. In the remainder of this paper, we will assume that for a given MIP at least one staircase structure is known.

An important aspect of MIP solving are *primal heuristics*. Finding good feasible solutions in a short amount of time may accelerate a branch-and-bound algorithm since a low primal bound means a good chance that a node can be pruned. In addition, they may be the only chance to find a feasible solution at all and deliver what practitioners need most.

Related Literature. In the last years, primal heuristics for generic MIPs have gained increasing attention in the literature. *Rounding heuristics* [6] [26] round fractional variables of an LP feasible solution while trying to preserve LP feasibility. *Diving heuristics* quickly go down the branch-and-bound tree by starting with an LP optimal solution at a node and iteratively perform a branching decision and re-solving the LP relaxation. *Objective diving heuristics* also use a modified objective function—a prominent example is the so-called *feasibility pump* [16]. *Large neighborhood search (LNS) heuristics*—e. g. [14] [18] [24]—search a

neighborhood of a given feasible solution by fixing some of the variables and solving the residual MIP—a so-called *sub-MIP*.

Besides those heuristics—of which none assumes a problem structure as defined above—, there exist heuristics tailored to problems such as the aforementioned lot sizing problem. One of them is the *rolling horizon* approach that successively solves the single stages of a MIP. This approach was successfully applied to problems over a discrete time horizon, as e. g. lot sizing problems [8] [9] [19] [21], but also on maritime inventory routing problems [23].

Our contribution. In this paper, we will present three heuristics tailored to arbitrary MIPs with a staircase structure and give a comparison to existing generic MIP heuristics. We present a generalized rolling horizon heuristic that is not restricted to a particular application but assumes only a staircase structure as defined above, making it applicable to any arbitrary staircase-structured MIP. The second approach is an improvement heuristic performing variable fixing according to the staircase structure. As a third approach, we extend diving heuristics to the context of staircase-structured MIPs.

The rest of the paper is organized as follows. In Sect. 2, we suggest generic MIP heuristics with one or more staircase structures as the only additional information; one of them is a generic rolling horizon heuristic that is not restricted to one particular type of model. In Sect. 3, we will give a comparison between our approaches and those from the literature and investigate in how far the knowledge of a staircase structure helps finding feasible solutions. Sect. 4 concludes this paper.

2 Heuristics Exploiting a Staircase Structure

In this section, we will present the three aforementioned heuristic approaches.

2.1 A Generalized Rolling Horizon Heuristic

The rolling horizon approach was successfully applied to problems over a discrete time horizon. Its main idea is that subproblems containing only parts of the time horizon are solved, proceeding from the first time steps to the last, while variables from earlier time steps are fixed. The hope is that solving these smaller subproblems is easier than solving the whole MIP at once.

In our general setting, each time step refers to a stage in the coefficient matrix (2). A general rolling horizon heuristic would therefore solve K subproblems—one for each stage—in successive order, while the variables from previous stages are fixed.

For our exact definition of the subproblems, we will write the right hand sides b of the rows (L_k, B_k, R_k) as b_k. The objective function is written as $c^T x = \sum_{k=1}^{K} (c^k)^T x^k + (\hat{c}^k)^T y^k$; we use an analogous notation for all other constant vectors in $\mathbb{R}^n$, for the index sets I^k and $\hat{I}^k$ of the integer variables and for the dimensions n^k and $\hat{n}^k$ of x^k and y^k, respectively. Note that in this notation, we

always assume y^K to be an empty vector and $\hat{n}^K = 0$. Furthermore, we use the following abbreviation:

$$x^{[k,l]} = ((x^k)^T, (y^k)^T, \ldots, (x^l)^T, (y^l)^T)^T$$

In order to reduce the risk of a subproblem to be infeasible, we do not only include the current stage but also $s \in \{0, \ldots, K\}$ further stages. Furthermore, we include $t \in \{0, \ldots, K\}$ stages thereafter with relaxed integrality constraints. With this "lookahead"—in the literature also called *forecast periods*—the risk of running into infeasibility is reduced at the cost of a larger subproblem. Note that, however, we fix only the variables from the current stage when the subproblem has been solved.

Moreover, when already performed fixings are incompatible with the current subproblem, one may ask if a feasible solution would exist if the solutions from, say p, previous stages were altered by a small amount. If $\bar{x}^{[k-p],k-1]}$ is the solution from the p previous subproblems, it may be of profit to allow to alter it by a percentage of $r \in [0, 1]$ of the number of variables, i. e. to accept values $x^{[k-p],k-1]}$ with

$$||x^{[k-p],k-1]} - \bar{x}^{[k-p],k-1]}||_1 \leq r \cdot \sum_{l=k-p}^{k-1} (n^l + \hat{n}^l).$$

Since $||\cdot||_1$ is a nonlinear term, it needs to be linearized. If l_j and u_j are lower and upper bounds of x_j, respectively, then deviation from $\bar{x}_j$ is only possible in one direction if the solution is at one of the bounds. If the solution is not at one of the bounds, we still allow deviation in only one direction, depending on where the risk that a linear constraint is violated is minimized. We judge this according to the number of locks which are used in particular by rounding heuristics:

Definition 2. *Consider a MIP in form* (1). *For a variable* x_j, $1 \leq j \leq n$, *define the number of* up-locks *as* $\zeta_j^+ = |\{i : A_{ij} > 0\}|$ *and the number of* down-locks *as* $\zeta_j^- = |\{i : A_{ij} < 0\}|$.

If $\zeta_j^+ > \zeta_j^-$, we then set $\bar{x}_j$ as the new upper bound of x_j; otherwise, we set its lower bound to ζ_j^+. This then yields the linear distance funtion

$$\begin{aligned} \Delta(x, \bar{x}) = &\sum_{j:\bar{x}_j = l_j} (x_j - l_j) + \sum_{j:\bar{x}_j = u_j} (u_j - x_j) \\ &+ \sum_{\substack{j: l_j < \bar{x}_j < u_j \\ \zeta_j^+ > \zeta_j^-}} (\bar{x}_j - x_j) + \sum_{\substack{j: l_j < \bar{x}_j < u_j \\ \zeta_j^+ \leq \zeta_j^-}} (x_j - \bar{x}_j) \end{aligned}$$

Thus, we define the k-th subproblem in the following way:

Definition 3. *For* $1 \leq k \leq K$, *the* k-th subproblem with IP-lookahead s, LP-lookahead t and deviation r in the p previous blocks *of a MIP with a staircase*

structure is defined as

$$
\begin{aligned}
&\min (\hat{c}^{k-p-1})^T y^{k-p-1} + \textstyle\sum_{l=k-p}^{k+s+t}((c^l)^T x^l + (\hat{c}^l)^T y^l) \\
s.\,t. \begin{pmatrix} L_{k-p-1} & B_{k-p} & R_{k-p} & & \\ & & \ddots & & \\ & & L_{k+s+t-1} & B_{k+s+t} & L_{k+s+t} \end{pmatrix} &\begin{pmatrix} y^{k-p-1} \\ x^{[k-p,k+s+t]} \end{pmatrix} \leq \begin{pmatrix} b^{k-p} \\ \vdots \\ b^{k+s+t} \end{pmatrix} \\
\Delta(x^{[k-p,k-1]}, \bar{x}^{[k-p,k-1]}) &\leq r \cdot \textstyle\sum_{l=k-p}^{k-2}(n^l + \hat{n}^l) + n^{k-1} \\
y^{k-p-1} &= \bar{y}^{k-p-1} \\
x_j^l &\leq \bar{x}_j^l && k-p \leq l \leq k-1, \\
& && l_j^l < \bar{x}_j < u_j^l \wedge (\zeta^+)_j^l > (\zeta^-)_j^l \\
y_j^l &\geq \bar{y}_j^l && k-p \leq l \leq k-2, \\
& && \hat{l}_j^l < \bar{y}_j < \hat{u}_j^l \wedge (\zeta^+)_j^l \leq (\zeta^-)_j^l \\
x_j^l &\in \mathbb{Z} && k-p \leq l \leq k+s,\ j \in I^l \\
y_j^l &\in \mathbb{Z} && k-p \leq l \leq k+s,\ j \in \hat{I}^l. \\
& && (\mathrm{MIP}_k^{s,t,p,r})
\end{aligned}
$$

The algorithm is shown in Alg. 1.

Algorithm 1: Rolling Horizon

Input: A MIP of the form (1) with a staircase structure
Output: a feasible solution $\bar{x}$, or 'unsuccessful'
initialize $\bar{x} \leftarrow 0$;
for $k = 1$ **to** K **do**
  solve $(\mathrm{MIP}_k^{s,t,p,r})$;
  if $(\mathrm{MIP}_k^{s,t,p,r})$ *is infeasible* **then**
    return 'unsuccessful' ;
  else
    set $\begin{pmatrix} \bar{y}^{k-p-1} \\ \bar{x}^{[k-p,k+s+t]} \end{pmatrix} \leftarrow$ optimal solution of $(\mathrm{MIP}_k^{s,t,p,r})$;
return $\bar{x}$;

Note that the larger s, t, and p are, the larger are the subproblems. In particular, if s is chosen too large, a subproblem may already include the whole remaining MIP which contradicts the work with smaller and easier to solve subproblems. Further note that in the literature, the special case with $p = 0$, $s = 0$ and $t = \infty$ is also referred as *fix-and-relax*.

Also note that starting with $k = 1$ and solving the subproblems up to $k = K$ is only one possible direction, one may also go "backwards" by going into the other direction. This is identical to appplying the heuristic as described here on another staircase structure which is obtained by substitute a given structure by

$B'_k = B_{K-k+1}$, $L'_k = R_{K-k}$, $R'_k = L_{K-k}$ and analogously for the x, c, and b vectors.

Last, let us mention that while most MIP heuristic assume that a feasible or an LP feasible solution is already known, this heuristic does not need to known any solution and can therefore before a branch-and-bound-algorithm to obtain a first feasible solution.

2.2 Fifty/fifty

In Sect. 1, we mentioned the class of large neighborhood search (LNS) heuristics whose main idea is to obtain a feasible solution by solving a smaller sub-MIP of the original MIP. A sub-MIP is obtained by fixing some of the variables e. g. to the values of an already known feasible solution.

Many of them work with at least two (LP or integer) feasible solutions. If only one feasible solution is known, an approach would be to arbitrarily fix only some of the variables to their solution values and leave the other variables free. If we have a partition of the variables into two subsets, i. e. two sets N_1 and N_2 with $N_1 \cup N_2 = \{1, \ldots, n\}$, $N_1 \cap N_2 = \emptyset$, then a *"fifty/fifty"* approach would be two first fix the variables x_j, $j \in N_1$, solve the sub-MIP, fix the variables from N_2 and solve another sub-MIP where, in turn, the N_1 are free. Thus, we try to obtain a feasible solution by solving two sub-MIPs of ideally "half of the size" in terms of variables of the original MIP. This then yields Alg. 2

Algorithm 2: Fifty/fifty

Input: A MIP of the form (1), a variable partition N_1, N_2 and a feasible solution $\bar{x}$
Output: a new feasible solution $\bar{x}$, or 'unsuccessful'
solve MIP (1) with the additional constraints $x_j = \bar{x}_j$, $j \in N_1$;
if *MIP is infeasible* **then**
  return 'unsuccessful' ;
$\bar{x} \leftarrow$ optimal solution of MIP ;
solve MIP (1) with the additional constraints $x_j = \bar{x}_j$, $j \in N_2$;
if *MIP is infeasible* **then**
  return 'unsuccessful' ;
$\bar{x} \leftarrow$ optimal solution of MIP ;
return $\bar{x}$;

Of course, the same heuristic may then be applied as well with the roles of N_1 and N_2 interchanged. In our setting, we apply the heuristic with the two sets representing the block and linking variables, respectively. Note that if the linking variables are fixed, then the MIP breaks down into independent subproblems.

2.3 Diving Heuristics

When solving a MIP with a branch-and-bound algorithm without any primal heuristics, feasible solutions only come from integral LP relaxation solutions. Having such an LP optimum at a branch-and-bound node may be "lucky coincidence", but it may also be the result of branching and node selection rules that favored such an outcome. Of course, one may in consequence branch and select nodes such as to reach feasibility, but this is not the only objective that guides the search strategy: another one would be, e. g., to improve the lower bound.

Since obtaining a feasible solution is still an important issue, one may wish to perform a tree search outside the branch-and-bound tree. This can be done by a heuristic: Starting from a node in the current tree, one can branch on a variable, solve the resulting LP and repeat this until either a feasible solution is found or the search is aborted. Since this is a depth-first-search that quickly goes down the tree, such a heuristic is called a *diving heuristic*. An outline of a diving heuristic is shown in Alg. 3.

Algorithm 3: Generic MIP Diving Heuristic

Input: A MIP of the form (1) and an LP feasible solution $\tilde{x}$, a maximum iteration parameter maxIter
Output: A feasible solution $\bar{x}$, or 'unsuccessful'
iter $\leftarrow 0$;
repeat
  iter $\leftarrow$ iter $+ 1$;
  choose a variable x_j with $j \in I$, $\tilde{x}_j \notin \mathbb{Z}$;
  add a branching restriction on x_j to the current LP relaxation ;
  solve the LP relaxation ;
  if *the LP relaxation is infeasible* **then**
    return 'unsuccessful' ;
  $\tilde{x} \leftarrow$ an optimal solution of the LP relaxation ;
  if $\tilde{x}$ *is feasible for MIP* (1) **then**
    return $\bar{x} = \tilde{x}$;
until *iter* $<$ *maxIter*;
return 'unsuccessful' ;

The variables to branch on are chosen in such a way that integral LPs are favored. For an overview, see [4].

However, none of these rules explicitly takes a problem structure into account; they solely assess each variable by a score and choose the variable with the best one. If a diving heuristic is applied on a structured problem, it is the question whether a staircase structure is somehow reflected implicitly in the choices of variables, e. g. if there is a diving heuristic that prefers variables from lower stages k. And if there is none, it may still be of interest if it is of any advantage to do so.

We therefore suggest the following modification of the variable selection: Choose the lowest stage k which contains a fractional integer variable; among the variables in this stage, apply the variable selection rule.

3 Computational Results

In the following section, we will evaluate the performance of the above described heuristics in terms of numbers and qualities of the found solutions. First, we describe the computational environment in which we tested the heuristics.

Experimental Setup. We implemented the heuristics in the generic mixed integer programming solver SCIP 3.1.1 [5] developed at the Zuse Institute Berlin; as underlying LP solver, CPLEX 12.6.1 [3] was used.

In addition, we extended SCIP by detector plugins which try to find a staircase structure in a MIP and which are part of the generic branch-and-price solver GCG [17]; GCG itself is an extension of SCIP. The rolling horizon and the fifty/fifty heuristic are part of the current development version of GCG and will be included in its next release. For testing the modified diving stratagies presented in this paper, we modified the diving heuristics of the current SCIP release.

The computational experiments were performed on a cluster of Intel Nehalem EX X7550 CPUs with 2.0 GHz, 18 MB cache and 256 GB of main memory running with a Scientific Linux 6.6 operating system and GCC 4.4.7. We limited the CPU time to half an hour and the number of branch-and-bound nodes to 5000 per instance.

Our test set was compiled from generic MIP test sets Cor@l [1], MIPLIB2010 [20] and NETLIB [2]. Furthermore, we added instances from applications with a time horizon: lot sizing [25], maritime inventory routing [22], network design [7] and temporal knapsack [13]. We restricted the test set to only those instances on which the staircase detectors were able to find a suitable staircase structure. Since there is no measure of the "goodness" of a staircase structure, we selected instances by visual judgment. This yielded a total of 597 instances.

Rolling Horizon. We applied our general rolling horizon heuristic before the start of the branch-and-bound algorithm. For each instance, we report the number of found solutions and the gap of its best solution against the best dual bound after hitting the time or node limits (or after having solved the instance, if it could be solved to optimality).

Furthermore, we compare the heuristic against RENS (relaxation enforced neighborhood search) [12]. Similarly to rolling horizon, it is a constructive sub-MIP based heuristic that is applied early in the branch-and-bound algorithm and tries to determine an optimal rounding of an LP relaxation; by default, it is called after having solved the root node relaxation.

Figure 2 shows a comparison in terms of found solutions and solution qualities: For each heuristic, a point represents the best solution the heuristic could

find on an instance so that a point close to the x-axis means that the heuristic was able to find a solution close to the dual bound; note that a y-value of 100 means a gap of 100 percent *or higher*. If the heuristic did not find a solution on an instance, no point is shown. For each heuristic, we sorted the points in non-descending order by their gaps, and we truncated the picture on the right side. Thus, the longer the curve, the more solutions a heuristic did find, and the closer it is to the x-axis, the better the quality of the found solutions.

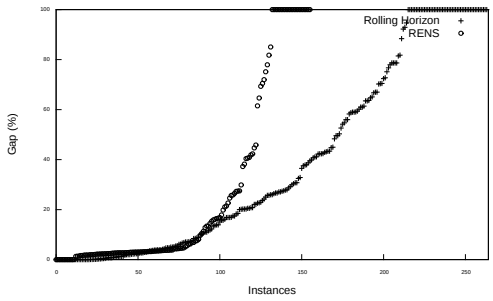

Fig. 2. Comparison of Rolling Horizon to RENS

In 579 calls, rolling horizon was able to produce 427 solutions, while RENS was called 273 times and found 159 solutions. This is reflected by the longer curve of rolling horizon in Fig. 2. While the curves are close to each other for 100 instances, they then start to diverge from each other. For 116 instances, rolling horizon finds a solution with a gap of 40 percent or less.

Fifty/fifty. Since fifty/fifty is an improvement heuristic, we compare it against other improvement heuristics which are implemented in SCIP; these are DINS [18], RINS [14], Crossover [11] [24], Mutation [24], Local Branching [15], One-opt and Two-opt [6]. We called each of these heuristic at each 10-th depth of the branch-and-bound tree; furthermore, they were only called if no new incumbent solution had been found in the last 50 nodes.

Figure 3 shows a comparison; we summarize SCIP's improvement heuristics by a single curve. The figure shows that fifty/fifty performs worse than SCIP's improvement heuristics both in terms of numbers of solutions and solution qualities.

Diving heuristics. For evaluating the diving heuristics, we switched off all other heuristics. We then ran SCIP with Coefficient, Fractionality, Guided, Linesearch, Pseudocost and Vectorlength Diving on each instance, once with the preferation of early stages and once without.

For each of these two settings, Fig. 4 shows a summarized curve for all of these diving heuristics. As the figure shows, the diving heuristics behave quite

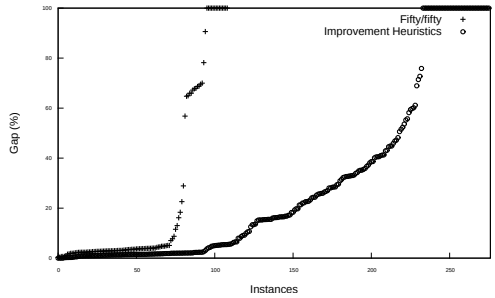

Fig. 3. Comparison of Fifty/fifty to other improvement heuristics

identically, with our modified strategy being slightly worse for the first 100 instances.

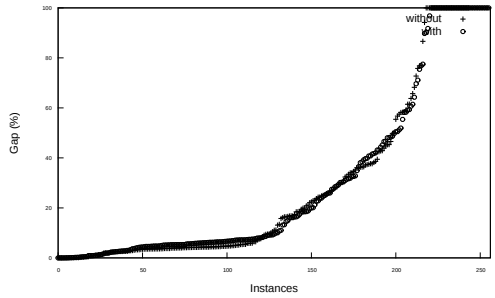

Fig. 4. Comparison of Diving with and without preferation of low stages

On the other hand, preferring low stages seems to influence the variable selection. Fig. 5 shows two exemplary runs of Coefficient Diving on a temporal knapsack instance, without (left) and with (right) preferation of low stages; it can be seen that while the picture on the left hand side looks quite "chaotic" in the sense that variables are really selected from arbitrary blocks, the picture on the right hand side shows an almost non-descending curve.

4 Conclusion

In this paper, we presented three heuristic approaches to general mixed integer programs with a staircase structure and evaluated their performance against existing MIP heuristics.

It turned out that a generalized rolling horizon heuristic is suitable for a wide range of applications in that it was able to produce good feasible solutions for a high number of instances of different types. The fact that it does not require any

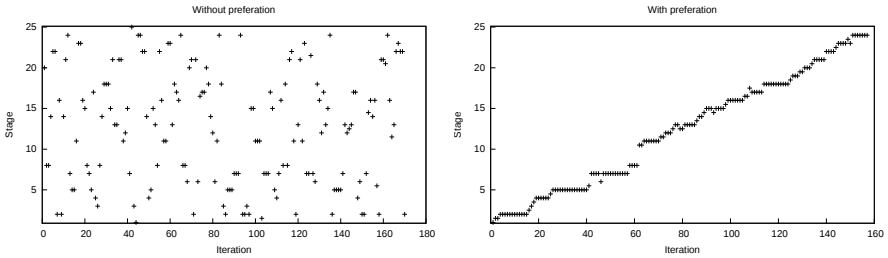

Fig. 5. Variable selection of Coefficient Diving in temporal knapsack instance `new1_1`

integer feasible nor LP feasible solution makes it a good start heuristic that can be applied either standalone or right at the beginning of a branch-and-bound algorithm.

Fifty/fifty, on the other hand, also produced solutions but was not able to beat the existing improvement heuristics. Note that we tested only one variable partitioning approach; there remains the question whether there exist variable partitions with which the heuristic would perform better.

For the diving heuristics, we could not observe any improvements by taking a staircase structure into account.

Acknowledgments. We thank Timo Berthold for devising the fifty/fifty heuristic and providing us with a first SCIP implementation.

References

1. COR@L MIP instances., http://coral.ie.lehigh.edu/data-sets/mixed-integer-instances/, http://coral.ie.lehigh.edu/data-sets/mixed-integer-instances/
2. The NETLIB LP test problem set, http://www.numerical.rl.ac.uk/cute/netlib.html, http://www.numerical.rl.ac.uk/cute/netlib.html
3. IBM ILOG CPLEX Optimizer, version 12.6.1 (2014), www-03.ibm.com/software/products/de/ibmilogcpleoptistud/
4. Achterberg, T.: Constraint Integer Programming. Ph.D. thesis, Technische Universität Berlin (2007), http://opus.kobv.de/zib/volltexte/2009/1153/
5. Achterberg, T.: SCIP: Solving constraint integer programs. Mathematical Programming Computation 1(1), 1–41 (2009)
6. Achterberg, T., Berthold, T., Hendel, G.: Rounding and propagation heuristics for Mixed Integer Programming. ZIB-Report 11-29, Zuse Institute Berlin (2011)
7. Atamtürk, A., Rajan, D.: On splittable and unsplittable flow capacitated network design arc–set polyhedra. Mathematical Programming 92(2), 315–333 (2002)
8. Baker, K.R.: An experimental study of the effectiveness of rolling schedules in production planning. Decision Sciences 8(1), 19–27 (1977)
9. Beraldi, P., Ghiani, G., Grieco, A., Guerriero, E.: Rolling-horizon and fix-and-relax heuristics for the parallel machine lot-sizing and scheduling problem with sequence-dependent set-up costs. Computers & Operations Research 35(11), 3644–3656 (2008)

10. Bergner, M., Caprara, A., Furini, F., Lübbecke, M.E., Malaguti, E., Traversi, E.: Partial convexification of general MIPs by Dantzig-Wolfe reformulation. In: Günlük, O., Woeginger, G. (eds.) Integer Programming and Combinatorial Optimization. LNCS, vol. 6655, pp. 29–51. Springer Berlin / Heidelberg (2011)
11. Berthold, T.: Primal Heuristics for Mixed Integer Programs. Master's thesis, Technische Universität Berlin (2006), http://opus.kobv.de/zib/volltexte/2007/1054/
12. Berthold, T.: RENS – the optimal rounding. ZIB-Report 12–17, Zuse Institute Berlin (2012), http://opus4.kobv.de/opus4-zib/frontdoor/index/index/docId/1520/
13. Caprara, A., Furini, F., Malaguti, E.: Exact algorithms for the temporal knapsack problem. Technical report OR-10-7, DEIS, University of Bologna (2010)
14. Danna, E., Rothberg, E., Pape, C.L.: Exploring relaxation induced neighborhoods to improve MIP solutions. Mathematical Programming 102, 71–90 (2005)
15. Fischetti, M., Lodi, A.: Local branching. Mathematical Programming 98, 23–47 (2003)
16. Fischetti, M., Salvagnin, D.: Feasibility pump 2.0. Mathematical Programming Computation 1, 201–222 (2009)
17. Gamrath, G., Lübbecke, M.E.: Experiments with a Generic Dantzig-Wolfe Decomposition for Integer Programs. In: Festa, P. (ed.) Experimental Algorithms, Lecture Notes in Computer Science, vol. 6049, pp. 239–252. Springer Berlin / Heidelberg (2010)
18. Ghosh, S.: DINS, a MIP improvement heuristic. In: Fischetti, M., Williamson, D. (eds.) Integer Programming and Combinatorial Optimization, LNCS, vol. 4513, pp. 310–323. Springer Berlin / Heidelberg (2007)
19. Guimarães, L., Klabjan, D., Almada-Lobo, B.: Pricing, relaxing and fixing under lot sizing and scheduling. European Journal of Operational Research 230(2), 399–411 (2013)
20. 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. Mathematical Programming Computation 3(2), 103–163 (2011), http://mpc.zib.de/index.php/MPC/article/view/56/28
21. Mercé, C., Fontan, G.: MIP-based heuristics for capacitated lotsizing problems. International Journal of Production Economics 85(1), 97–111 (2003)
22. Papageorgiou, D.J., Nemhauser, G.L., Sokol, J., Cheon, M.S., Keha, A.B.: MIRPLib – A library of maritime inventory routing problem instances: Survey, core model, and benchmark results. European Journal of Operational Research 235(2), 350–366 (2014), maritime Logistics
23. Rakke, J.G., Stålhane, M., Moe, C.R., Christiansen, M., Andersson, H., Fagerholt, K., Norstad, I.: A rolling horizon heuristic for creating a liquefied natural gas annual delivery program. Transportation Research Part C: Emerging Technologies 19(5), 896–911 (2011)
24. Rothberg, E.: An evolutionary algorithm for polishing Mixed Integer Programming solutions. Tech. rep., ILOG Inc. (2005)
25. Tempelmeier, H., Derstroff, M.: A Lagrangean-based heuristic for dynamic multilevel multiitem constrained lotsizing with setup times. Management Science 42(5), 738–757 (1996)
26. Wallace, C.: ZI round, a MIP rounding heuristic. Journal of Heuristics 16(5), 715–722 (2010)